UNIVERSIDADE ESTADUAL DE CAMPINAS
FACULDADE DE ENGENHARIA ELÉTRICA E DE COMPUTAÇÃO
DEPARTAMENTO DE COMUNICAÇÕES

RECONHECIMENTO DE CARACTERES NUMÉRICOS MANUSCRITOS QUE CONSIDERA AS CARACTERÍSTICAS INTRÍNSECAS E COMUNS DE CADA DÍGITO

Este exemplar corresponde à redação final da tese defendida por Edna Lúcia Flôres
e aprovada pela Comissão Julgadora em 04/02/97
Orientador

Autora: Edna Lúcia Flôres
Orientador: Prof. Dr. João Baptista Tadanobu Yabu-uti

1º volume da tese apresentada à Faculdade de Engenharia Elétrica e de Computação da Universidade Estadual de Campinas, como parte dos requisitos exigidos para a obtenção do título de **Doutora em Engenharia Elétrica**

97.08021

Fevereiro - 1997

UNICAMP
BIBLIOTECA CENTRAL

UNIDADE BC
N.º CHAMADA: T/UNICAMP F663r
V. 01 Ex.
TOMBO BC/ 30967
PROC. 281/97
C [ ] D [x]
PREÇO R$ 11,00
DATA 05/07/97
Nº CPD 0.000 99252-4

FICHA CATALOGRÁFICA ELABORADA PELA
BIBLIOTECA DA ÁREA DE ENGENHARIA - BAE - UNICAMP

F663r

Flôres, Edna Lúcia
Reconhecimento de caracteres numéricos manuscritos que considera as características intrínsecas e comuns de cada dígito / Edna Lúcia Flôres.--Campinas, SP: [s.n.], 1997
2 v.

Orientador: João Baptista Tadanobu Yabu-uti.
Tese (doutorado) - Universidade Estadual de Campinas Faculdade de Engenharia Elétrica e de Computação.

1. Reconhecimento de padrões. 2.Processamento de imagens. I. Yabu-uti, João Baptista Tadanobu. II. Universidade Estadual de Campinas. Faculdade de Engenharia Elétrica e de Computação. III. Título.

RECONHECIMENTO DE CARACTERES NUMÉRICOS MANUSCRITOS QUE CONSIDERA AS CARACTERÍSTICAS INTRÍNSECAS E COMUNS DE CADA DÍGITO

Autora: Edna Lúcia Flôres

1º volume da Tese apresentada à Faculdade de Engenharia Elétrica e de Computação da Universidade Estadual de Campinas, como parte dos requisitos exigidos para a obtenção do título de **Doutora em Engenharia Elétrica**

Banca Examinadora:

Prof. Dr. João Baptista Tadanobu Yabu-uti

Prof. Dr. Gilberto Arantes Carrijo

Prof. Dr. Yuzo Iano

Prof. Dr. Edson Moschim

Prof. Dr. Lee Luan Ling

UNICAMP, 04 de Fevereiro de 1997

Aos meus pais, Pedro e Odette, agradeço o amor, o apoio e os ensinamentos que me guiaram e permitiram esta e tantas outras realizações.
Ao meu pai, especialmente, agradeço a boa e feliz lembrança que deixou, para continuar a me guiar.
E à minha mãe, agradeço a perseverança e a garra para continuarmos lutando juntas nesta caminhada.

Agradecimentos

Gostaria de agradecer a todos os que de alguma forma tenham me ajudado na realização deste trabalho, seja através de discussões, críticas, sugestões ou colocações pessoais.

Ao meu orientador Prof. Dr. João Baptista Tadanobu Yabu-uti, pela orientação e apoio recebidos na elaboração deste trabalho.

Ao Prof. Dr. Gilberto Arantes Carrijo pelas discussões valiosas, apoio, incentivo e amizade dispensados, sem os quais esta tarefa tornar-se-ia bastante árdua.

A Maria Neusa Costa Moraes pela grande sujestão dada para o desenvolvimento do software.

A Eder Naves Rezende pela grande colaboração prestada no desenvolvimento do software.

Ao colega Dr. José Roberto Camacho (Phd) e a Osmar Alves de Souza Filho por suas valiosas colaborações e paciência às nossas solicitações.

A Capes pelo apoio financeiro e ao Departamento de Engenharia Elétrica da Universidade Federal de Uberlândia pela infraestrutura, sem os quais o desenvolvimento deste trabalho seria impossível.

Acima de tudo, quero agradecer a Deus, que nos momentos difíceis de nossas vidas, sabe nos dar conforto e paz para enfrentá-los.

RESUMO

Este trabalho descreve o desenvolvimento de um software em linguagem C para o reconhecimento de caracteres numéricos manuscritos. Estes caracteres podem estar isolados, sobrepostos ou conectados e apresentar ao longo do seu contorno trechos contínuos ou fragmentados.

No desenvolvimento deste trabalho, o reconhecimento dos caracteres foi realizado inicialmente utilizando-se redes neuronais. Os resultados obtidos com esse processo não são satisfatórios quando ocorrem diferenças expressivas entre os padrões utilizados no treinamento e os padrões utilizados no reconhecimento. As diferenças que podem ocorrer entre os padrões utilizados são mudança de escala, translação e rotação. A mudança de escala ocorre quando as dimensões do padrão utilizado no treinamento são diferentes das dimensões do padrão utilizado no reconhecimento. A translação ocorre quando o padrão utilizado no treinamento encontra-se localizado em uma posição distinta à posição do padrão utilizado no reconhecimento. A rotação ocorre quando os padrões utilizados no treinamento e no reconhecimento apresentam inclinações diferentes. Com o objetivo de minimizar essas diferenças, os dígitos utilizados antes de serem treinados e reconhecidos pela rede neuronal são normalizados, tornando-se insensíveis a mudança de escala, translação e rotação.

Vários testes foram realizados utilizando-se padrões normalizados. Os resultados obtidos nesses testes foram satisfatórios quando na fase de treinamento os padrões utilizados foram escritos pelas mesmas pessoas que escreveram os padrões utilizados na fase de reconhecimento. Porém, para que 90% ou mais dos padrões testados fossem reconhecidos, foi necessário aumentar a quantidade de padrões utilizados no processo de treinamento da rede neuronal. Esse aumento resultou em uma elevação considerável no tempo de processamento da rede neuronal na fase de treinamento. Ao serem utilizados na fase de reconhecimento,

padrões escritos por pessoas diferentes das pessoas que escreveram os padrões utilizados na fase de treinamento, a taxa de reconhecimento apresentou expressiva redução; sendo que em alguns casos essa taxa chegou a ser igual a 57,75%. Devido a este baixo percentual de acerto e aos problemas encontrados na fase de reconhecimento quando ocorrem diferenças expressivas entre o padrão utilizado na fase de treinamento e o padrão utilizado na fase de reconhecimento, resolveu-se desenvolver um novo algoritmo de reconhecimento de caracteres manuscritos que é totalmente diferente do algoritmo de reconhecimento utilizado na rede neuronal.

O processo utilizado pelo novo algoritmo de reconhecimento de caracteres manuscritos que é a base deste trabalho tem como princípio a verificação das características intrínsecas de cada dígito e a verificação das características que os dígitos têm em comum. Ao analisar estas características, esse algoritmo procurou aproximar-se do processo de reconhecimento utilizado pelo ser humano. Com o objetivo de facilitar a definição das características intrínsecas e comuns dos dígitos, o esqueleto de cada um destes dígitos é obtido utilizando-se o algoritmo rápido de afilamento (ARA) desenvolvido neste trabalho.

O algoritmo ARA possui dois passos principais, que são repetidos até que a imagem obtida aproxime-se do eixo médio da imagem original. No primeiro passo é marcado o contorno da imagem e no segundo passo o contorno marcado é analisado visando verificar quais os "pixels-on" pertencentes a esse contorno devem ser apagados.

Com o objetivo de verificar a eficiência do algoritmo ARA foram realizados vários testes. Através da análise dos resultados obtidos nesses testes, observa-se que o algoritmo ARA mantém a conectividade e preserva os pontos finais da imagem original, é praticamente imune ao ruído e, de extrema importância, têm rápido tempo de execução. Os resultados obtidos pelo algoritmo ARA foram comparados com os resultados obtidos pelos algoritmos One-Pass

Thinning Algorithm (OPTA) e Paralled-Symmetric Thinning Algorithm (PSTA). Nestes testes foi observado que o algoritmo ARA é aproximadamente 3,0 vezes mais rápido do que o algoritmo PSTA, e 3,3 a 5,3 vezes mais rápido do que o algoritmo OPTA.

O algoritmo de reconhecimento de caracteres numéricos manuscritos desenvolvido neste trabalho, parte do topo da imagem de cada dígito e agrupa os "pixels-on" em linhas horizontais e linhas verticais. É a partir das coordenadas iniciais e finais dessas linhas que são definidas as características intrínsecas e comuns de cada um dos dígitos.

Com o objetivo de testar a eficiência do algoritmo de reconhecimento de caracteres numéricos manuscritos, foram realizados testes utilizando-se dígitos isolados. Os resultados obtidos nesses testes apresentaram uma porcentagem de acerto de 99,67%.

Em nossa vida diária existem várias aplicações para o reconhecimento de caracteres numéricos manuscritos, uma dessas aplicações é o reconhecimento da quantia numérica de cheques bancários. Neste trabalho, com o objetivo de comprovar a eficiência do algoritmo de reconhecimento, além de serem realizados testes com dígitos isolados, foram realizados testes em que a quantia numérica de cheques bancários era reconhecida. Para que estes testes pudessem ser realizados, foi necessário adicionar ao algoritmo de reconhecimento, algoritmos que realizassem a segmentação de caracteres sobrepostos e caracteres conectados, eliminassem vírgulas, pontos, cerquilhas, 'R$' impressos, 'R$' manuscritos e campo que envolve a quantia numérica.

Nos testes realizados com cheques bancários foi observado que o processo utilizado na segmentação de dígitos sobrepostos e conectados possibilitou que o algoritmo de reconhecimento apresentasse uma taxa de reconhecimento de 99,92%. Esta taxa é um pouco superior as taxas de reconhecimento de sistemas desenvolvidos atualmente, que estão normalmente entre 93% a 99,5%.

As principais características que o algoritmo de reconhecimento de caracteres manuscritos desenvolvido neste trabalho possui são:

a) o reconhecimento dos caracteres manuscritos independe da pessoa que os escreveu;

b) o reconhecimento dos caracteres independe do seu tamanho e da posição em que o caractere foi escrito;

c) desempenho satisfatório, mesmo na presença de ruído;

d) taxa de acerto elevada.

Palavras Chaves: - Reconhecimento, caracteres numéricos manuscritos, dígitos, esqueleto, caracteres numéricos conectados, caracteres numéricos sobrepostos, padrões.

ABSTRACT

This work describes the development of software in C language for the recognition of handwritten numerical characters. These characters may be isolated, overlapped or connected and present continuous or fragmented parts along their boundaries.

In the development of this work, the recognition of characters was realized initially using neural networks. The results obtained from using this process are not satisfactory when expressive differences occur between the training and recognition patterns used. Differences that can occur between the used patterns are scale change, translation and rotation. The scale change occurs when the dimensions of the training pattern are different from the dimensions of the recognition pattern. The translation occurs when the training pattern is located in a distinctly different position from the recognition pattern. The rotation occurs when the patterns used in training and recognition present different inclinations. With the objective of minimizing the differences, digits used before being trained and recognized by the neural network are normalized, becoming insensitive to scale change, translation and rotation.

Various tests were carried out using normalized patterns. The results obtained from these tests were satisfactory when during the training phase the patterns used were written by the same people who wrote the patterns during the recognition phase.However, in order that 90% or more of the tested patterns to be recognized, it was necessary to increase the quantity of patterns used in the training process of the neural network. This increase resulted in a considerable elevation of processing time of the neural network in the training phase. When used in the recognition phase, patterns written by different people from the people who wrote the patterns of the training phase, the rate of recognition presented a expressive reduction; in some cases being equal to 57,75%. Due to this low

accuracy and the problems found in the recognition phase when there are expressive differences between the pattern used in the training phase and the pattern used in the recognition phase, it was decided that a new algorithm of recognition of handwritten characters be developed, which is totally different from the algorithm used in the neural network.

The process used by the new recognition algorithm of handwritten characters which is the basis of this work has as its basis the intrinsic verification of the characteristics of each digit and the verification of the characteristics that these digits have in common. Analysing these characteristics, this algorithm tried to come close to the recognition process used by the human being. With the objective of making the definition of the intrinsic and common characteristics easier, the skeleton of each one of these digits is obtained using the Fast Thinning Algorithm (FTA) developed in this work.

The algorithm FTA has two main steps, that are repeated until the obtained image approaches the medium axis of the original image. In the first step the contour of the image is marked and in the second step the contour marked is analysed to verify which pixels-on belong to this contour and which should be deleted.

With the objective of verifying the efficiency of the FTA algorithm various tests were carried out. Through the analysis of the results obtained in these tests, it was observed that the FTA algorithm maintains the connectivity and preserves the end points of the original image, it is practically immune to noise and, of extreme importance, has lower execution time. The results obtained by the FTA algorithm were compared with the results obtained by the One-Pass Thinning Algorithm (OPTA) and Paralleled-Symmetric Thinning Algorithm (PSTA). In these tests it was observed that the algorithm FTA is approximately 3,0 times quicker than the PSTA algorithm and 3,3 to 5,3 times quicker than the OPTA algorithm.

The recognition algorithm of handwritten numerical characters developed in this work starts at the top of the image of each digit and groups the pixels-on in horizontal and vertical lines. It is at the start of the initial and final coordinates of these lines that the intrinsic and common characteristics of each of the digits are defined.

With the objective of testing the efficiency of the recognition algorithm of handwritten numerical characters, tests were carried out using isolated digits. The results obtained in these tests presented an accuracy equal to 99,67%.

In our daily life there are various applications for the recognition of handwritten numerical characters, one of these applications is the recognition of the numerical amount of bank checks. In this work, with the objective of confirming the efficiency of the algorithm of recognition, besides carrying out tests with isolated digits, tests were carried out in which the numerical amount of bank checks was recognized. In order for these tests to be carried out, it was necessary to add to the recognition algorithm, other algorithms that carry out the segmentation of overlapped and connected characters, elimination of commas, periods, nonnumerical characters (#), printed 'R$', handwritten 'R$' and the surrounding field of the numerical amount.

In the tests carried out with bank checks it was observed that the process used in segmentation of overlapped and connected digits made it possible for the algorithm of recognition to present a rate of recognition of 99,92%. This rate is a little superior to the rates of recognition systems developed nowadays, that are normally between 93% to 99,5%.

The principal characteristics of the recognition algorithm of handwritten numerical characters developed in this work are:

a) the recognition of the handwritten characters is independent of who wrote them;

b) the recognition of characters is independent of its size and the position in which it was written;

c) satisfactory performance, also in the presence of noise;

d) higher rate of accuracy.

Keywords: Recognition, handwritten numerical characters, digits, skeleton, connected numerical characters, overlapped numerical characters, patterns.

SUMÁRIO DO 1º VOLUME

CAPÍTULO 1 : ESTADO DA ARTE.....1

1.1 - INTRODUÇÃO.....1

1.2 - APLICAÇÕES DA TECNOLOGIA DE RECONHECIMENTO DE CARACTERES.....5

1.3 - IMPLEMENTAÇÃO DO OCR DESENVOLVIDO NESTE TRABALHO.....7

1.4 - SEGMENTAÇÃO.....9

1.5 - PRÉ-PROCESSAMENTO.....11

1.6 - METODOLOGIAS UTILIZADAS NO RECONHECIMENTO DE CARACTERES.....11

1.6.1- Abordagens Utilizadas no Reconhecimento de Caracteres.....12

1.6.2 - Esquemas Baseados na Natureza das Aplicações.....13

1.6.3 - Classificação Baseada nas Características Utilizadas.....14

1.6.3.1 - Casamento de padrões.....14

1.6.3.2 - Análise das características e técnicas de casamento.....15

1.7 - ALGUNS DOS TRABALHOS EM RECONHECIMENTO DE CARACTERES.....17

1.8 - ALGUMAS DAS PESQUISAS REALIZADAS NA ÁREA DE RECONHECIMENTO DE CARACTERES NO INÍCIO DOS ANOS OITENTA.....18

1.9 - ALGUNS DOS TRABALHOS DE PESQUISAS ATUAIS EM RECONHECIMENTO DE CARACTERES.....20

1.10 - ALGUNS DOS OCR's PRÁTICOS.....22

1.11 - CONCLUSÕES.....23

1.12 - RESUMO DOS CAPÍTULOS DESENVOLVIDOS NESTE TRABALHO.....24

CAPÍTULO 2 : RECONHECIMENTO DE DÍGITOS MANUSCRITOS ISOLADOS INSENSÍVEIS A MUDANÇA DE ESCALA, TRANSLAÇÃO E ROTAÇÃO UTILIZANDO-SE REDES NEURONAIS.....26

2.1 - INTRODUÇÃO.....26
2.2 - MUDANÇA DE ESCALA, TRANSLAÇÃO E ROTAÇÃO.....28
2.2.1 - Invariância a Mudança de Escala.....29
2.2.2 - Invariância a Translação.....31
2.2.3 - Invariância a Rotação.....33
2.3 - REDES NEURONAIS.....37
2.4 - RESULTADOS OBTIDOS.....41
2.5 - CONCLUSÕES.....47

CAPÍTULO 3 : SEGMENTAÇÃO E UM ALGORITMO DE AFILAMENTO.....50

3.1- INTRODUÇÃO.....50
3.2 - LEITURA DO ARQUIVO ONDE ESTÁ GRAVADA A IMAGEM DIGITALIZADA.....51
3.3 - SEGMENTAÇÃO DO TEXTO.....56
3.4 - DESCRIÇÃO DOS ALGORITMOS OPTA E PSTA.....58
3.4.1 - Descrição do "One-Pass Thinning Algorithm" (OPTA).....59
3.4.2 - Descrição do "Parallel-Symmetric Thinning Algorithm" (PSTA)....60
3.5 - DESCRIÇÃO DE UM ALGORITMO RÁPIDO DE

AFILAMENTO (ARA)....64
3.6 - RESULTADOS OBTIDOS....78
3.7 - CONCLUSÕES....81

CAPÍTULO 4 : UM ALGORITMO PARA RECONHECIMENTO DE CARACTERES NUMÉRICOS MANUSCRITOS ISOLADOS QUE CONSIDERA AS CARACTERÍSTICAS INTRÍNSECAS E COMUNS DE CADA DÍGITO....82

4.1 - INTRODUÇÃO....82
4.2 - DESCRIÇÃO DO ALGORITMO PARA RECONHECIMENTO DE CARACTERES NUMÉRICOS MANUSCRITOS ISOLADOS QUE CONSIDERA AS CARACTERÍSTICAS INTRÍNSECAS E COMUNS DE CADA DÍGITO....85
4.3 - RESULTADOS OBTIDOS....93
4.4 - CONCLUSÕES....193

CAPÍTULO 5 : RECONHECIMENTO DE CARACTERES MANUSCRITOS DE CHEQUES BANCÁRIOS....195

5.1 - INTRODUÇÃO....195
5.2 - ELIMINAÇÃO DA VÍRGULA E DO PONTO....198
5.2.1 - Eliminação da Vírgula....198
5.2.2 - Eliminação do Ponto e/ou Ruído....207
5.3 - SEGMENTAÇÃO DE CARACTERES SOBREPOSTOS....210
5.4 - VERIFICAÇÃO SE O CARACTERE É CONECTADO

OU ISOLADO, CLASSIFICAÇÃO E DESCONEXÃO
DOS CARACTERES CONECTADOS.....217
5.4.1 - Verificação se o Caractere é Conectado ou Isolado.....217
5.4.2 - Classificação e Segmentação dos Caracteres
Conectados.....246
5.5 - ELIMINAÇÃO DO CAMPO QUE ENVOLVE A
QUANTIA NUMÉRICA DE CHEQUES BANCÁRIOS.....255
5.6 - ELIMINAÇÃO DA CERQUILHA.....280
5.7 - ELIMINAÇÃO DO 'R' E 'CIFRÃO' MANUSCRITOS
E DO 'R' E 'CIFRÃO' IMPRESSOS NA ÁREA
RESERVADA À ESCRITA DA QUANTIA
NUMÉRICA DE CHEQUES BANCÁRIOS.....294
5.7.1 - Eliminação do 'R' e 'Cifrão' Manuscritos.....294
5.7.2 - Eliminação do 'R$' Impresso na Área
Reservada à Escrita da Quantia Numérica
de Cheques Bancários.....300
5.8 - JANELAS DE ENTRADA E SAÍDA DO PROGRAMA.....301
5.8.1 - Janela de Apresentação do Programa.....302
5.8.2 - Janela de Entrada do Programa.....303
5.8.3 - Janela de Saída do Programa.....303
5.9 - RESULTADOS OBTIDOS.....308
5.10 - CONCLUSÕES.....312

CAPÍTULO 6 : CONCLUSÕES, CONTRIBUIÇÕES DO TRABALHO E SUGESTÕES PARA FUTUROS TRABALHOS.....314

6.1 - CONCLUSÕES.....314

6.2 - CONTRIBUIÇÕES DO TRABALHO.....319
6.3 - SUGESTÕES PARA FUTUROS TRABALHOS.....319

REFERÊNCIAS BIBLIOGRÁFICAS.....322

LISTA DE FIGURAS

Figura 1.1 - Diagrama de blocos de um OCR.....7

Figura 2.1 - Rede neuronal com 3 camadas: entradas s_i, camada escondida h_k e unidades de saída u_j. Pesos v_{ik} conectam as entradas às unidades escondidas, e pesos w_{kj} conectam a camada escondida às unidades de saída.....38

Figura 2.2 - Dígitos de uma pessoa antes de serem submetidos aos processos de mudança de escala, translação e rotação.....43

Figura 2.3 - Dígitos de uma pessoa depois de serem submetidos aos processos de mudança de escala, translação e rotação.....43

Figura 2.4 - Gráfico da porcentagem de entradas reconhecidas x número de arquivos de uma pessoas, utilizados na fase de treinamento.....43

Figura 2.5 - Gráfico da porcentagem de entradas reconhecidas x número de arquivos de 4 pessoas distintas, utilizados na fase de treinamento.....45

Figura 3.1 - Cabeçalho de um arquivo no formato tiff.....54

Figura 3.2 - "Pixel" testado e "pixels" vizinhos.....59

Figura 3.3 - Padrões de afilamento (máscaras).....59

Figura 3.4 - Máscaras de restituição.....60

Figura 3.5 - Máscaras utilizadas no algoritmo PSTA.....61

Figura 3.6 - Varredura na imagem da esquerda para a direita
usando a "janela" de afilamento (máscara 4 x 4).....62

Figura 3.7 - Varredura na imagem da direita para a esquerda
usando a "janela" de afilamento (máscara 4 x 4).....62

Figura 3.8 - Obtenção do esqueleto final da
imagem utilizando o algoritmo PSTA.....63

Figura 3.10 - Código de cadeia de 8 direções.....65

Figura 3.11 - a) Imagem original; b) Delimitação inicial do
contorno da imagem; c) Delimitação do contorno
da imagem com espessura de 1 "pixel-on" e ruído;
d) Delimitação do contorno da imagem
no último passo do algoritmo ARA;
e) Esqueleto final da imagem.....67

Figura 3.12 - Máscaras utilizadas na verificação da disposição
do "pixel-on" marcado com o valor 2 e
dos "pixels-on" vizinhos a este "pixel-on".....68

Figura 3.13 - Máscaras utilizadas na verificação da disposição
do "pixel-on" marcado com o valor 2 e

dos "pixels-on" vizinhos a esse "pixel-on".....68

Figura 3.16 - "Pixel-on" marcado com o
valor 2 e seus 2 "pixels-on" vizinhos adjacentes.....72

Figura 3.19 - Análises realizadas pela função verifica-desvio.....77

Figura 3.20 - Exemplo de um caso em que o "pixel-on" marcado
com o valor 2 é movido para a posição encontrada.....78

Figura 3.21 - a) Imagem original do número 2;
b) Esqueleto obtido pelo algoritmo OPTA;
c) Esqueleto obtido pelo algoritmo PSTA;
d) Esqueleto obtido pelo algoritmo ARA.....79

Figura 3.22 - a) Imagem original do número 8;
b) Esqueleto obtido pelo algoritmo OPTA;
c) Esqueleto obtido pelo algoritmo PSTA;
d) Esqueleto obtido pelo algoritmo ARA.....79

Figura 3.23 - a) Imagem original do número 9;
b) Esqueleto obtido pelo algoritmo OPTA;
c) Esqueleto obtido pelo algoritmo PSTA;
d) Esqueleto obtido pelo algoritmo ARA.....80

Figura 3.24 - a) Imagem original do caractere alfabético A;
b) Esqueleto obtido pelo algoritmo OPTA;
c) Esqueleto obtido pelo algoritmo PSTA;

d) Esqueleto obtido pelo algoritmo ARA.....80

Figura 3.25 - a) Imagem original do caractere alfabético B;
b) Esqueleto obtido pelo algoritmo OPTA;
c) Esqueleto obtido pelo algoritmo PSTA;
d) Esqueleto obtido pelo algoritmo ARA.....80

Figura 3.26 - a) Imagem original do caractere alfabético L;
b) Esqueleto obtido pelo algoritmo OPTA;
c) Esqueleto obtido pelo algoritmo PSTA;
d) Esqueleto obtido pelo algoritmo ARA.....80

Figura 4.2 - Linha horizontal superior delimitada da
esquerda para a direita em 2 números manuscritos.....93

Figura 4.5 - Linha horizontal inferior delimitada da
esquerda para a direita em 2 números manuscritos.....104

Figura 4.8 - Linha vertical esquerda em 2 números manuscritos.....114

Figura 4.9 - Vários casos existentes do primeiro tipo
de ponto de cruzamento considerado pelo programa.....115

Figura 4.10 - Os casos em que mais ocorrem o segundo tipo
de ponto de cruzamento considerado pelo programa.....115

Figura 4.11 - Ponto de cruzamento em 2 números manuscritos.....116

Figura 4.13 - Números manuscritos varridos pela função vertical.....123

Figura 4.15 - Números manuscritos varridos pela função horizontal.....131

Figura 4.18 - Alguns números reconhecidos pela função verif-1-3-5-6-0.....144

Figura 4.20 - Alguns números reconhecidos
pela função verif-0-8-6-3-5-1-2-4.....158

Figura 4.22 - Alguns números reconhecidos
pela função verif-2-3-5-7-9-8-0.....178

Figura 4.24 - Alguns números reconhecidos
pela função verif-7-9-4-1-2-0-6-8-3.....188

Figura 4.26 - Alguns números reconhecidos
pela função verif-1-4-5-6-7.....192

Figura 5.5 - Números sobrepostos segmentados e reconhecidos.....216

Figura 5.14 - Alguns dos números conectados que foram
desconectados e reconhecidos pelo algoritmo.....255

Figura 5.15 - Tipos de campos encontrados na área
reservada à escrita da quantia numérica.....256

Figura 5.20 - Área reservada à escrita da quantia
numérica do cheque bancário em que

esta quantia é delimitada por 2 cerquilhas.....280

Figura 5.22 - Área reservada à escrita da quantia
numérica de um cheque bancário,
preenchida e delimitada pelo 'R$' manuscrito.....294

Figura 5.24 - Janela de apresentação.....302

Figura 5.25 - Janela que solicita ao usuário o nome do arquivo
onde está gravada a imagem da área reservada
à escrita da quantia numérica a ser reconhecida.....303

Figura 5.26 - Janela que informa ao usuário que o arquivo
onde está gravada a imagem da área reservada
à escrita da quantia numérica a ser reconhecida
não foi encontrado.....304

Figura 5.27 - Janela apresentada ao usuário informando que os
caracteres estão sendo reconhecidos pelo programa.....305

Figura 5.28 - Janela apresentada ao usuário informando
que o sistema não possui memória suficiente.....306

Figura 5.29 - Janela apresentada ao usuário informando
que a dimensão da pilha é insuficiente.....307

Figura 5.30 - Janela que apresenta os resultados
obtidos no processo de reconhecimento.....308

LISTA DE TABELAS

Tabela 2.1 - Porcentagem de entradas reconhecidas
ao serem utilizados arquivos de uma pessoa.....44

Tabela 2.2 - Porcentagem de entradas reconhecidas
ao serem utilizados arquivos de 4 pessoas distintas.....45

Tabela 3.1 - Tempo de execução dos algoritmos: OPTA,
PSTA e ARA, utilizando-se um microcomputador
Pentium - 100 Mhz com 16 Mbytes de memória RAM.....81

Tabela 5.1 - Tempo de execução do programa para o reconhecimento
da quantia numérica de cheques bancários, quando
são encontrados apenas caracteres numéricos isolados.....310

Tabela 5.2 - Tempo de execução do programa para o reconhecimento
da quantia numérica de cheques bancários, quando são
encontrados caracteres numéricos
isolados, sobrepostos e conectados.....310

Tabela 5.3 - Tempo de execução do programa para o reconhecimento
de caracteres numéricos escritos na área reservada à escrita
da quantia numérica de cheques bancários, quando nesta
área existem tarjas verticais e/ou tarjas horizontais.....311

Tabela 5.4 - Tempo de execução do programa para o reconhecimento de 5 caracteres numéricos isolados escritos na área reservada à escrita da quantia numérica de cheques bancários, quando nesta área existem as tarjas verticais, a tarja horizontal inferior, cerquilhas, 'R$' impresso e 'R$' manuscrito.....312

CAPÍTULO 1

ESTADO DA ARTE

1.1 - INTRODUÇÃO

O reconhecimento automático de caracteres manuscritos tem sido objeto de intensas pesquisas por mais de três décadas. Existem atualmente disponíveis no mercado, sistemas manuais, de mesa, de médio e de grande porte utilizados em várias aplicações com preços que podem ser de até meio milhão de dólares. Contudo, o principal objetivo de se desenvolver uma máquina de leitura com as mesmas capacidades do ser humano ainda permanece inatingido.

Existe ainda, uma grande diferença entre a capacidade de leitura humana e a da máquina, porém um grande esforço tem sido dispendido para diminuir tal diferença.

O processo de reconhecimento de caracteres é também conhecido como reconhecimento óptico de caracteres (OCR) pois este processo trabalha com o reconhecimento de caracteres processados opticamente [1].

O reconhecimento de caracteres surgiu por volta do ano de 1870,

inicialmente como uma ajuda aos deficientes visuais. A primeira tentativa que obtive sucesso, foi realizada pelo cientista russo Tyurin em 1900. A versão moderna do OCR surgiu em meados dos anos 40 com o desenvolvimento dos computadores digitais [1]. Desde então, foi idealizada como uma abordagem de processamento de dados com aplicação no mundo dos negócios. A principal motivação para o desenvolvimento de sistemas OCR é a necessidade de se lidar com um grande fluxo de informações tais como cheques bancários, formulários comerciais, fichas governamentais, impressos de cartões de crédito, etc.

Máquinas OCR estão disponíveis comercialmente desde meados dos anos 50 [1]. Desde então, um grande número de pesquisas e artigos técnicos foram desenvolvidos e publicados. Diversos livros sobre reconhecimento óptico de caracteres foram publicados e também, várias discussões e relatórios sobre o tópico apareceram repetidamente nos anais das "International Joint Conferences on Pattern Recognition" e das "International System, Man and Cybernetics Conferences". Resultados de pesquisas também foram publicados em várias outras conferências tais como "British Conferences on Pattern Recognition" e "The Scandinavian Conferences on Image Analysis". Govindan e Shivaprasad [1] citam em seu artigo que os trabalhos que ilustram o estado da arte sobre pesquisas em reconhecimento de caracteres foram apresentados por Nagy, Harmon, Stallings, Suen, Berthold e Mori; Mori, Yamamoto e Yasuda; Mantas; Davis e Yall; e Chaterji.

Atualmente, as metodologias de reconhecimento de caracteres avançaram desde técnicas primitivas para o reconhecimento de números impressos e um número limitado de letras inglesas (latinas) até a aplicação de sofisticadas técnicas para o reconhecimento de uma grande variedade de caracteres manuais complexos, símbolos e palavras incluindo caracteres chineses e japoneses. O reconhecimento de caracteres chineses utilizando o computador foi considerado um problema muito difícil e deixado como um dos objetivos finais da pesquisa

em reconhecimento de caracteres. Hoje, um grande número de organizações de pesquisa e estabelecimentos comerciais no Japão e na China estão ativamente engajados em introduzir inovações e sofisticações nos sistemas de reconhecimento, com o objetivo de atingir melhores performances nas leitoras de texto e caracteres chineses. Dispositivos de leitura de caracteres chineses e japoneses estão agora disponíveis. Por exemplo, o leitor de caractere CLL-2000 [1] pode ler caracteres chineses assim como caracteres do japonês Hirakana e Katakana. Os japoneses utilizam cerca de 2000 caracteres em sua vida diária, que são compostos dos alfabetos Kanji, Hirakana, Katakana, Romano e números árabes. Os caracteres que compõem o alfabeto Kanji são quase os mesmos caracteres que compõem o alfabeto chinês. Cerca de aproximadamente 3000 caracteres Kanji, são comumente usados no Japão. Caracteres chineses são ideogramas em que a maioria dos traços são verticais e horizontais. Uma representação adequada de um ideograma requer uma matriz de "pixels" cerca de 10 vezes maior do que a matriz de "pixels" necessária para representar uma letra romana. Isto demonstra a complexidade e sofisticação necessárias para desenvolver uma máquina leitora de caracteres chineses que apresente desempenho satisfatório.

A maioria dos reconhecedores citados na literatura e aqueles comercialmente disponíveis são dedicados unicamente ao reconhecimento de caracteres de um conjunto de alfabetos. Porém, como o japonês usa o Kanji, o Hirakana, o Katakana, etc., alguns de seus OCR's podem ler mais de um alfabeto, como no caso do CLL-2000.

Os trabalhos de pesquisas divulgados até agora incluem o desenvolvimento de reconhecedores para o Inglês (Latin), Japonês, Chinês, Indiano, Árabe e Coreano. Alguns trabalhos referem-se a reconhecedores para o Cirílico (Russo), Hebreu (linguagem semita), Thai, Grego e Bérbere.

A importância de qualquer trabalho no campo de reconhecimento de

caracteres pode ser vista pela variedade de aplicações práticas.

As necessidades típicas de um sistema de reconhecimento numérico (por exemplo leitura do código de endereçamento postal) podem ser resumidas como descritas a seguir [2]:

(a) o sistema de reconhecimento deve ser capaz de reconhecer os caracteres de qualquer pessoa;

(b) o sistema de reconhecimento deve ser capaz de processar caracteres de tamanhos arbitrários;

(c) o sistema de reconhecimento deve ser robusto na presença de ruído;

(d) o sistema de reconhecimento deve ter taxas de reconhecimento bastante elevadas;

(e) nas aplicações comerciais o tempo de execução dos sistemas de reconhecimentos deve ser pequeno.

Nas máquinas de reconhecimento de caracteres manuscritos, os resultados obtidos dependem da qualidade e da variabilidade dos dados utilizados. Pesquisas recentes em reconhecimento automático de numerais manuscritos têm obtido resultados com taxas de reconhecimento que variam de 93% a 99,5% para diferentes tipos de dados de teste [3].

Este capítulo trata das aplicações da tecnologia de reconhecimento de caracteres e da implementação do OCR desenvolvido neste trabalho. Também são realizados comentários sobre segmentação, pré-processamento, metodologias utilizadas em reconhecimento de caracteres; são citados alguns dos trabalhos em

reconhecimento de caracteres, algumas das pesquisas realizadas na área de reconhecimento de caracteres no início dos anos oitenta, alguns dos trabalhos de pesquisas atuais em reconhecimento de caracteres e alguns dos OCR's práticos. São realizadas conclusões a respeito dos OCR's existentes e finalmente é feito um resumo sobre os capítulos subseqüentes desenvolvidos neste trabalho.

1.2 - APLICAÇÕES DA TECNOLOGIA DE RECONHECIMENTO DE CARACTERES

A tecnologia de reconhecimento de caracteres tem várias aplicações práticas. Algumas das literaturas conhecidas a respeito deste tópico foram escritas em outras línguas que não o Inglês, tais como Alemão, Japonês, etc. A seguir são descritas algumas das aplicações nas quais os OCR's têm sido utilizados ou têm sido sugeridos por pesquisadores [1].

- Utilização por deficientes visuais como auxiliares de leitura utilizando fotosensores e simuladores de tato e como auxiliar sensorial com saída sonora. Também são utilizados para leitura e reprodução de originais em braile.

- Utilização como auxiliar de telecomunicações para surdos.

- Utilização em departamentos postais para leitura do endereçamento postal e como leitor de códigos postais impressos e manuscritos.

- Medida e análise da qualidade de caracteres impressos, leitura de documentos e ordenação; reservas de linhas aéreas e departamentos

de veículos automotores; como leitor automático do número de placas e gravador para controle do tráfego nas estradas.

- Uso em editoras, como leitor para terminal de comunicação de dados.

- Sistemas de transferência bancária, leitura dos documentos, ordenação e anotações fiscais e leitura das ordens de transferência.

- Processamento direto de documentos, como leitor de documentos de vários tipos no processamento de dados em larga escala; como leitor de microfilme para entrada de sistemas de dados, para entrada de dados em alta velocidade; como leitor eletrônico no manuseio de grandes volumes de correspondência.

- Tarifação do cliente no sistema telefônico; digitais; inspeção de máscaras de CI's e detecção de defeitos em microcircuitos.

- Leitura de código de barra digital e como analisador de escrita manual; no reconhecimento automático do escritor e verificação de assinaturas.

- Leitura mecânica de documentos em empresas têxteis e de manufatura de roupas, toques automáticos em telégrafos industriais, processamento de dados em empresas de alimentação e leitura de preços, códigos e nome do produto no varejo.

- Taxação de exames e na avaliação de fichas de atendimento.

- Cartografia automatizada, indústria metalúrgica, sistemas de linguagem forense auxiliados por computador, correio eletrônico, unidades de informação e bibliotecas e para fax-símile.

- Indústrias eletrônicas e leitura de caracteres estampados em partes metálicas.

1.3 - IMPLEMENTAÇÃO DO OCR DESENVOLVIDO NESTE TRABALHO

Existem várias maneiras que podem ser utilizadas na implementação de um OCR. Conforme a aplicação a que o OCR se destina, é utilizada uma determinada forma de implementação. No entanto, todas as formas de implementação possuem intrínsicamente uma filosofia básica que pode ser esquematizada como mostrado na figura 1.1.

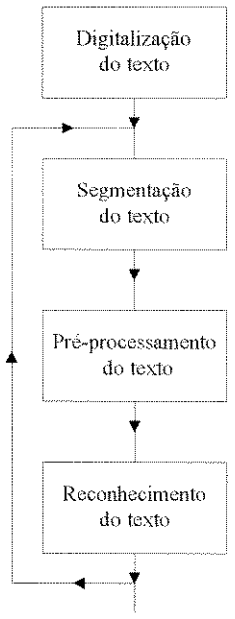


Figura 1.1 - Diagrama de blocos de um OCR.

O nível de complexidade de cada um dos blocos mostrados na figura 1.1 varia conforme o tipo de texto que se quer reconhecer. O OCR idealizado neste trabalho, possui as seguintes características:

i) A digitalização do texto é realizada utilizando-se um "scanner" de mesa e o sistema Deskscan, que funciona no ambiente Windows. A imagem digitalizada é gravada em um arquivo em disco. A imagem gravada, é lida por um programa que a transfere para uma matriz na memória do computador.

ii) A matriz obtida no item (i) contém a imagem do texto que será segmentado em dois componentes básicos. Estes componentes são as linhas da imagem do texto e a imagem dos caracteres que compõem estas linhas.

iii) Segmentada a linha da imagem do texto e em seguida cada uma das imagens de seus caracteres, é realizado o pré-processamento que tem como objetivo a [4]: filtragem, restauração, realce e afilamento da imagem do caractere que foi segmentado.

iv) Reconhecimento da imagem dos caracteres segmentados. Neste processo se a imagem dos caracteres a serem reconhecidos estão sobrepostos ou conectados, são realizados alguns procedimentos que possibilitam a separação e o reconhecimento de cada imagem desses caracteres.

A seguir, são realizados comentários sobre os itens descritos acima, nestes comentários o item (i) não foi incluído, pois este item é analisado mais detalhadamente no capítulo 3.

1.4 - SEGMENTAÇÃO

Segmentação [5] é o processo de identificação da localização de linhas, campos e caracteres individuais que formam os objetos a serem reconhecidos.

No processo de segmentação os procedimentos utilizados são mais heurísticos que os procedimentos utilizados no processo de reconhecimento e também suas taxas de sucesso são mais difíceis de serem quantificadas. Talvez seja por estas razões, que o processo de segmentação tenha sido objeto de um número menor de pesquisas que o processo de reconhecimento.

Como citado por Westall e Narasimha [5], a abordagem da segmentação de números conectados é fornecida por Dunn e Wang.

Dois artigos que tratam especificamente de segmentação de números manuscritos foram publicados por Shridhar e Badreldin nos anos 80. Em seu primeiro artigo [6] eles descrevem uma abordagem baseada no rastreamento dos perfis superior e inferior das imagens dos números a serem segmentados. No segundo artigo Shridhar e Badreldin [7] apresentam uma abordagem na qual um ponto central do caminho de partição ("fronteira de decisão") é localizado através da análise dos pontos de cruzamento feitos por linhas horizontais centralmente localizadas. O restante do caminho de partição é então construído ao se seguir a fronteira ou a linha média dos dois caracteres.

Como citado por Govidan e Shivaprasad [1], Tampi e Chetlur propuseram a segmentação de caracteres conectados manuscritos através da utilização de vocabulário de descrição de imagem. Este vocabulário consiste de palavras com características inerentes que fornecem as características essenciais para a segmentação. Outro trabalho que trata da segmentação de caracteres conectados citado por Govindan e Shivaprasad [1] é o trabalho desenvolvido por Ting e Ward.

Como citado por Westall e Narasimha [5], um refinamento da abordagem

por análise de perfil desenvolvido por Shridhar e Badreldin foi proposto por Kimura e Shridhar. Este refinamento é baseado em um procedimento que determina a direção de máxima separabilidade de um par de numerais conectados. O caminho de partição é uma linha reta cuja localização é derivada a partir de análise estatística dos perfis superior e inferior.

Como citado por Westall e Narasimha [5], Cheriet, Huang e Suen propuseram um método baseado na análise de regiões de "pixels" de fundo. Eles tentavam casar regiões de "pixels" de fundo a partir da parte inferior dos numerais com regiões opostas a partir do topo. Quando as regiões casadas são identificadas, um corte é construído conectando essas regiões e particionando os numerais. Se nenhuma região casada é identificada, um corte vertical é construído tanto para cima a partir da região de "face up" quanto para baixo a partir da região de "face down".

Como citado por Westall e Narasimha [5], Fujisawa, Yakano, Komatsu, Kadota e Kurino propuseram um sistema de segmentação baseado no processo de reconhecimento. Este sistema necessita que um reconhecedor identifique corretamente todos os pares numéricos no estado conectado. Foi também requerido que o reconhecedor seja apto a identificar certos casos de fragmentos numéricos (por exemplo o numeral "5" com a barra horizontal superior removida). Em um artigo recente, Fujisawa, Nakano e Kurino [8] propuseram uma abordagem mais tradicional baseada em uma análise de contorno orientado a padrão. Neste sistema, o reconhecimento é utilizado para discriminar entre os vários caminhos de partição possíveis identificados no estágio de análise.

Como citado por Westall e Narasimha [5], Bayer, Kressel e Hammelsbeck descreveram duas abordagens para a segmentação de caracteres alfabéticos conectados, um problema que é significativamente mais difícil que a segmentação de numerais conectados. Um classificador estatístico é usado para determinar prováveis localizações para os cortes verticais. Um classificador de árvore que

incorpora reconhecimento decide então quais dos possíveis cortes devem realmente ser efetuados.

Westall e Narasimha [5] propuseram o algoritmo chamado Segmentação Direcionada por Vértice. Neste algoritmo os numerais conectados são particionados por cortes baseados em vértices selecionados e então extendidos às partes superior e inferior dos numerais.

1.5 - PRÉ-PROCESSAMENTO

As técnicas de filtragem para remoção dos ruídos adquiridos pela imagem quando ela é digitalizada, restaurada e realçada são utilizadas para melhorar a qualidade da imagem.

Neste trabalho a remoção de ruídos é realizada nos processos de afilamento e reconhecimento dos caracteres. Devido a qualidade dos dados utilizados no reconhecimento não houve necessidade de que a técnica de restauração fosse utilizada, já que esses dados ao serem varridos pelo "scanner" ficaram bem visíveis. O realce foi feito no próprio sistema Deskscan que funciona no ambiente Windows.

1.6 - METODOLOGIAS UTILIZADAS NO RECONHECIMENTO DE CARACTERES

As metodologias utilizadas no reconhecimento de caracteres podem ser consideradas de várias maneiras. As três principais maneiras [1] são baseadas nas:

(1) abordagens utilizadas;

(2) natureza das aplicações, e

(3) características utilizadas.

1.6.1 - Abordagens Utilizadas no Reconhecimento de Caracteres

Existiam inicialmente duas abordagens utilizadas no reconhecimento de padrões. Estas abordagens são: abordagem estatística/decisão-teórica e abordagem sintática/lingüística/gramatical/estrutural. Cada uma dessas abordagens possui seus méritos e deméritos. A informação estrutural sobre as interconexões nos padrões complexos não podem ser muito bem tratadas por técnicas de reconhecimento de padrões do tipo estatístico. Por outro lado, o uso de modelos teóricos de linguagem formal para se representar padrões é a principal desvantagem da abordagem sintática. Padrões são entidades naturais que não podem estritamente obedecer ao conjunto de restrições matemáticas imposto pela teoria da linguagem formal. Impor uma regra rigorosa na estrutura do padrão não é particularmente aplicável ao reconhecimento de caracteres, onde as variações entre as classes são infinitas. A abordagem lingüística nos informa pouco sobre as limitações do extrator de características. Portanto, um modelo híbrido é a única solução para a resolução de problemas práticos de reconhecimento de caracteres. Citando Fu [4], "... a dicotomia das abordagens sintática e de decisão teórica parece ser conveniente apenas sob o ponto de vista do estudo teórico. Em outras palavras, a divisão entre as duas abordagens, não é às vezes bem definida particularmente em termos de aplicações práticas."

Para o reconhecimento de caracteres necessita-se de técnicas que descrevam um grande número de estruturas similares que pertençam a mesma categoria e que permitam descrições distintas entre padrões de diferentes categorias. As pesquisas sobre reconhecimento de caracteres têm como principal

objetivo desenvolver máquinas que com a mesma capacidade de reconhecimento do ser humano, possam ler textos manuscritos ou impressos, porém, num rítmo mais rápido. Tem-se a expectativa que se as características que as pessoas usam para reconhecer caracteres forem descritas corretamente e utilizadas em um algoritmo de reconhecimento de caracteres, este algoritmo seria tão eficiente quanto o ser humano. Esta é a motivação para a utilização da abordagem descritiva na pesquisa sobre reconhecimento de caracteres. A descrição de um caractere envolve características (detalhes estruturais) e as regras sob as quais essas características compõem um caractere. Para se alcançar uma descrição eficiente, as características utilizadas devem ser independentes. Isto é, a presença de nova(s) característica(s) ou a falta de característica(s) antiga(s) não devem afetar a descrição das características remanescentes. Isto fornece alguma imunidade às limitações dos extratores de características.

1.6.2 - Esquemas Baseados na Natureza das Aplicações

Conforme a natureza das aplicações, é possível agrupar os trabalhos de reconhecimento de caracteres em dois esquemas principais. Estes esquemas são: reconhecimento de caracteres "off-line" e "on-line" [1]. Nos sistemas "off-line", o reconhecimento dos caracteres não é realizado ao mesmo tempo em que os caracteres são escritos. No reconhecimento "on-line", o reconhecimento é realizado enquanto os caracteres são escritos, e portanto a informação sobre o tempo dispendido na escrita de cada caractere está também disponível juntamente com a imagem do caractere.

Com base na capacidade e complexidade, pode-se ainda classificar os esquemas "off-line" como [1]:

(1) Reconhecimento de caracteres de fonte fixa, lidam com o reconhecimento de

um tipo específico de fonte escrita como OCR-A; OCR-B; Pica, Elite, etc.

(2) Reconhecimento de caracteres multifonte, reconhecem mais de uma fonte.

(3) Reconhecimento de caracteres "omni-font", reconhecem qualquer tipo de fonte.

(4) Reconhecimento de caracteres manuscritos, reconhecem caracteres manuscritos normais não conectados.

(5) Reconhecimento de letras, lidam com o reconhecimeto de caracteres manuscritos livres que podem estar conectados ou isolados.

1.6.3 - Classificação Baseada nas Características Utilizadas

Em termos das características utilizadas, as técnicas de reconhecimento de caracteres podem ser classsificadas como [1]:

(1) casamento de padrões, e

(2) análise das características e técnicas de casamento.

1.6.3.1 - Casamento de padrões

O casamento de padrões compara diretamente o caractere de entrada com um conjunto de padrões previamente guardados. Deste conjunto de padrões, o caractere de entrada é reconhecido como sendo o padrão a que ele mais se assemelha.

Do ponto de vista de Inteligência Artificial, o casamento de padrões não

deve ser incluído como explicação para o desempenho humano.

1.6.3.2 - Análise das características e técnicas de casamento

Estas são as técnicas mais frequentemente utilizadas no reconhecimento de caracteres. Nestes métodos, características significativas do caractere são extraídas e comparadas com as características dos caracteres ideais. Das características dos caracteres ideais a que mais se assemelha com as características significativas do caractere, fornece o reconhecimento deste caractere. O desempenho humano é melhor descrito através das técnicas de análise de características do que através das técnicas de casamento de padrões.

Muitas técnicas de análise de características foram desenvolvidas e aplicadas no reconhecimento de caracteres. A maioria dessas técnicas são exemplos dos métodos tradicionais de reconhecimento de caracteres, e são usualmente aplicadas em domínios restritos. Como citado por Govindan e Shivaprasad [1]; Suen, Berthod e Mori proporcionaram uma visão muito útil das várias técnicas de casamento de características. Os detalhes descritos a seguir são baseados no trabalho de Suen, Berthod e Mori.

Baseado no tipo de técnica de extração de características utilizada, as técnicas de análise de características são agrupadas em [1]:

(1) Transformação global e expansão em série;

(2) Características derivadas da distribuição estatística dos pontos, e

(3) Características geométricas e topológicas.

As técnicas de transformação global e expansão em série ajudam a reduzir

a dimensão do vetor de características e fornecem características invariantes a algumas deformações globais como translação e rotação. Nestas técnicas, os pesquisadores utilizaram as transformadas de Fourier, Walsh, Haar, Hadamard, expansão em séries, expansão de Karhunen-Loève, transformada de Hough, transformada de projeção, transformada da cadeia de código e transformada do eixo principal. Através da utilização dessas características, os processos de extração e preparação da máscara tornam-se bastante simples. Contudo, tais técnicas de extração de características demandam alto trabalho computacional.

As características derivadas da distribuição estatística dos pontos incluem Momentos de Zoning, n-uplas, Características dos Lugares, Cruzamentos e Distâncias. Estas características são tolerantes a distorção e previnem em um certo grau a variação de estilos, fornecem alta velocidade e baixa complexidade de implementação. Porém, em geral, a preparação da máscara é difícil para estes tipos de características.

O método de análise das características geométricas e topológicas é a técnica mais popular investigada por pesquisadores. As características podem representar propriedades globais e locais dos caracteres. Estas características incluem traços, seções em várias direções, delimitações, intersecções de segmentos de linhas, "loops", relações de traço, propriedades angulares e saliências agudas. Estas características têm alta tolerância a distorções, variações de estilo e também toleram um certo grau de translação e rotação. Elas auxiliam no processamento de caracteres em altas velocidades. Porém, os processos de extração são geralmente muito complexos e a geração de máscaras para estes tipos de características são geralmente difíceis.

1.7 - ALGUNS DOS TRABALHOS EM RECONHECIMENTO DE CARACTERES

Como citado por Govindan e Shivaprasad [1], um esforço notável realizado na área de reconhecimento de caracteres foi feito por Grimsdale, Summer, Tunis e Kilburn em 1958. No seu método, o padrão do caractere de entrada obtido por um "scanner" é descrito em termos do comprimento e da inclinação dos segmentos de reta e do comprimento e curvatura dos segmentos curvos. A descrição do padrão do caractere de entrada é comparada com a descrição do protótipo armazenado no computador com o objetivo de encontrar a decisão certa sobre a identidade do caractere desconhecido.

Como citado por Govindan e Shivaprasad [1], outro trabalho importante realizado é o método de análise por síntese sugerido por Eden do M.I.T. Ele propôs que todos os caracteres latinos podem ser formados por 18 traços, que por sua vez, podem ser formados por subconjuntos de 4 traços, chamados "hump" (concavidade), "bar" (barra), "hook" (gancho) e "loop" (forma circular). Como citado por Govindan e Shivaprasad [1], alguns dos exemplos de trabalhos nesta direção são de Blesser, Shillman, Kuklinski, Cox, Eden e Ventura; Cox, Blesser e Eden; Shillman, Kuklinski e Blesser; Yoshida e Eden; e Berthod. Blesser e autores propuseram uma abordagem teórica baseada em atributos fenomenológicos. Cox e autores apresentaram dois grupos principais de regras gramaticais para se lidar com a variabilidade de tipos de fonte. Três técnicas experimentais utilizados no estudo de caracteres ambíguos e na investigação da relação entre atributos físicos e funcionais foram sugeridas por Shillman e autores. Yoshida e Eden propuseram um sistema de reconhecimento de caracteres que emprega um processo gerador para extrair a seqüência de traços a partir do padrão de entrada e um dicionário de traços utilizado para o reconhecimento. Berthod utilizou as primitivas de Eden para análise de letras manuscritas.

Como citado por Govindan e Shivaprasad [1], Narasimhan sugeriu um esquema de rótulos para a descrição sintática de figuras e uma interpretação de classes de figuras direcionada a sintaxe. Depois, Narasimhan e Reddy apresentaram um esquema de reconhecimento auxiliado por sintaxe, onde eles incorporam na regra de decisão alguma flexibilidade necessária ao desempenho satisfatório do sistema de decisão. Os autores expressaram que as regras atualmente em uso devem ser refinadas, modificadas e aumentadas continuamente com base em experimentos e outros conhecimentos relevantes adquiridos.

Como citado por Govindan e Shivaprasad [1], Pavlidis e Ali utilizaram um algoritmo "split and merge" para a aproximação poligonal de caracteres no reconhecimento de numerais. Uma técnica de geração de característica para o reconhecimento do padrão sintático pela aproximação dos limites do caractere por polígonos e a decomposição baseada na concavidade foi sugerida por Feng e Pavlidis em 1974.

1.8 - ALGUMAS DAS PESQUISAS REALIZADAS NA ÁREA DE RECONHECIMENTO DE CARACTERES NO INÍCIO DOS ANOS OITENTA

Como citado por Govindan e Shivaprasad [1], algumas das importantes pesquisas sobre reconhecimento de caracteres no início dos anos oitenta foram realizadas por Tanaka, Hirakawa e Kaneku; Sarvarayudu e Sethi; Sato, Isshiki, Ohoka e Yoshida; Evangelisti e Shridhar; e Badreldin [9, 10]. Uma breve descrição dessas pesquisas é realizada a seguir.

Tanaka e autores apresentaram um novo sistema de reconhecimento de padrões distorcidos utilizando o algoritmo de Viterbi e uma treliça modificada incorporando a estatística relacionada aos padrões modificados. A treliça inicialmente elimina todas as classes de padrões irrelevantes, deixando apenas os

padrões mais prováveis para a decisão final. O método é utilizado para o reconhecimento de caracteres manuscritos escritos em inglês e japonês Katakana.

Os trabalhos desenvolvidos por Sarvarayudu e Sethi, Shridhar e Badreldin tem como objetivo o reconhecimento de caracteres numéricos. Sarvarayudu e Sethi utilizaram para caracterizar as fronteiras do padrão os descritores de Walsh e apresentaram uma técnica que para a reconstrução da fronteira do padrão utiliza também esses descritores. Shridhar e Badreldin apresentaram inicialmente um algoritmo de reconhecimento de caracteres de dois estágios que utilizava descritores de Fourier e descritores topológicos para se atingir uma alta precisão no reconhecimento de caracteres numéricos. Com o objetivo de melhorar a velocidade do reconhecimento, prejudicada pelo alto consumo computacional dos descritores de Fourier, Shridhar e Badreldin utilizaram um novo conjunto de características topológicas derivadas de uma descrição global do caractere. O sistema de reconhecimento consiste de um classificador sintático que analisa a estrutura topológica do padrão.

Para funcionar como um dispositivo econômico de entrada para um sistema de processamento de dados distribuídos, Sato e autores sugeriram um "scanner" manual do tipo OCR de baixo custo que pode ler caracteres impressos. Esse "scanner" utiliza um sensor de imagem unidimensional e o processo de varrer a imagem é realizado manualmente na direção horizontal.

Um método de avaliar um "scanner" para o reconhecimento de caracteres antes de ser projetada a lógica do reconhecimento foi proposto por Evangelisti. O "scanner" é avaliado comparando o padrão por ele produzido com padrões standard selecionados pelo computador.

1.9 - ALGUNS DOS TRABALHOS DE PESQUISAS ATUAIS EM RECONHECIMENTO DE CARACTERES

Como citado por Govindan e Shivaprasad [1], uma grande quantidade de trabalhos de pesquisa foram desenvolvidos a partir de meados dos anos 80. Esses trabalhos incluem processamento pós-contextual desenvolvido por Nagy, Sethi e Einspahr; e Sinha; reconhecimento de palavra/letra desenvolvidos por Almuallim e Yamaguchi; El-sheikh e Guindi; Hull, Aoki e Yamaya; Wong e Fallside; e Shrihari e Bozinovic; separação de caracteres conectados desenvolvidos por Tampi e Chetlur; e Ting e Ward; reconhecimento de numerais desenvolvidos por Lam e Suen [3]; Baptista e Kulkarni; aprendizagem multifonte desenvolvido por Cannat, Kodratoff e Moscatelli; aprendizagem por experiência desenvolvido por Malyan e Sunthankar; reconhecimento de estenografia de Pitman desenvolvido por Leedham e Downton; descrição de padrões e técnicas de geração desenvolvido por Nagahashi e Nakatsuyama; reconhecimento por auxílio de descrição desenvolvido por Harjinder, técnicas de transformada de código de cadeia desenvolvido por Cheng e Leung e pré-classificação e reconhecimento utilizando transformada de Walsh desenvolvido por Huang e Lung.

Como citado por Govindan e Shivaprasad [1], Nagy, Sethi e Einspahr demonstraram um algoritmo heurístico que atribui identidade alfabética para símbolos num contexto textual. Essa atribuição baseia-se num pequeno vocabulário de palavras freqüentes do inglês, necessitando com isso, de poucos recursos de memória e computação. Um pós-processador contextual baseado em regras para o reconhecimento Devanagari foi sugerido por Sinha. Esse pós-processador consiste de um verificador de composição de sintaxe na forma de uma máquina de estados finitos. As regras de substituição estão na forma de pares ação-condição dando flexibilidade ao sistema para cada alteração. Cada regra de substituição tem uma penalidade associada e o valor acumulado de

penalidades para uma dada palavra é uma medida do seu nível de confiança.

Como citado por Govindan e Shivaprasad [1], a abordagem de percepção multinível para a leitura de letras manuscritas foi proposta por Shrihari e Bozinovic.

Lam e Suen [3] descreveram um sistema de classificação que associa a relaxação dos caracteres manuscritos do código de endereçamento postal. Esse sistema inclui um extrator de características que decompõe o esqueleto do caractere em primitivas geométricas e dois algoritmos de classificação. Um destes algoritmos é um classificador estrutural rápido que identifica a maioria das amostras e o outro é um algoritmo robusto de relaxação que classifica o resto dos dados.

Como citado por Govindan e Shivaprasad [1], Baptist e Kulkarni empregaram uma abordagem multinível no processamento da informação visual pelo cérebro humano, o que resultou em uma alta precisão no reconhecimento de caracteres manuscritos.

Como citado por Govindan e Shivaprasad [1], um método de geração e descrição de caracteres estruturais é relatado por Nagahashi e Nakatsuyama. Neste método, qualquer caractere é considerado como um padrão composto construído por vários sub-padrões simples, e é descrito em termos desses sub-padrões ao se introduzir 3 tipos de relações posionais entre eles.

Como citado por Govindan e Shivaprasad [1], Cheng e Leung sugeriram um novo método de transformação de parâmetros, chamado transformada do código de cadeia, próprio para o reconhecimento de padrões contendo linhas retas. O método da transformada do código de cadeia essencialmente mapeia as linhas de um caractere num espaço de parâmetros bi-dimensional semelhante ao utilizado na transformada de Hough. A técnica foi empregada em um experimento preliminar de reconhecimento com caracteres chineses e obteve uma melhor taxa de reconhecimento quando foi comparada com o método de projeção de perfil,

transformada de Fourier e transformada de Hough.

1.10 - ALGUNS DOS OCR's PRÁTICOS

Entre os vários OCR's existentes na atualidade, Govindan e Shivaprasad [1] citam alguns dos OCR's descritos a seguir.

Pequenos OCR's manuais, fáceis de manusear chegam a custar US$ 600,00 enquanto OCR's maiores, como os OCR's de mesa chegam a custar US$ 2.500,00. OCR's de médio e grande porte são encontrados a preços bem maiores.

A Kurzweil Corporation produz leitores de caracteres de tamanho médio que chegam a custar US$ 15.000,00. Esses leitores têm como vantagem a capacidade de reconhecimento de uma grande variedade de fontes.

Os Estados Unidos e outros países instalaram grandes máquinas de leitura de endereços postais que chegam a custar meio milhão de dólares. Essas máquinas possibilitam que sejam obtidas performances bem mais rigorosas que as performances obtidas pela maioria das outras leitoras.

O CLL-200 é um OCR de mesa que pode reconhecer com precisão cerca de 2400 caracteres manuscritos, incluindo entre estes caracteres, caracteres chineses, Hirakana e Katakana. Este OCR portátil possui 21 microprocessadores de 16 bits montados em 4 placas de circuito de tamanho A4 e pode ser interfaceado com sistemas de processamento de palavras japonesas.

Vossen descreveu o Formscan TXL4 Workless Station que pode ler até 1000 páginas por dia em um sistema de texto. Este OCR foi desenvolvido por Dest dos Estados Unidos para lidar com grandes quantidades de correspondências.

Um exemplo de um OCR de propósito geral é o OCR N376G desenvolvido pela NEC utilizando novas idéias e tecnologias para que fosse obtida uma alta vazão de processamento.

O CSL 2610 é um OCR desenvolvido para aplicações em correspondências de negócios. Este OCR tem a capacidade de processar formulários com o endereço do cliente pré-impresso e informações manuscritas adicionais sobre o cliente.

Um sistema autônomo para leitura de documentos manuscritos ou datilografados é o leitor de caracteres ópticos Siemens SLS9691.

Um exemplo de OCR de alta performance é o OCR TO-3000 desenvolvido pela ETL (Electrotechnical Laboratory). Este OCR pode reconhecer tanto caracteres manuscritos quanto caracteres impressos, através da utilização da técnica do ponto mais externo de Yamamoto.

1.11 - CONCLUSÕES

Com citado por Govindan e Shivaprasad [1], a performance dos sistemas disponíveis comercialmente apresentam várias limitações, que quando ultrapassadas podem causar uma grande deteriorização nas performances desses sistemas. Algumas das restrições comumente impostas em algumas ou em todas as máquinas são:

- Caracteres individuais não devem se tocar e o texto deve apresentar boa impressão, onde os caracteres devem ser impressos com tinta escura num fundo colorido.

- Os caracteres devem estar localizados em limites especificados.

- A capacidade multifonte é atingida somente se o operador treinar a máquina com novas fontes.

A partir da análise destas limitações e das baixas performances obtidas, pode-se concluir que a capacidade das máquinas em simular a habilidade humana na leitura de textos permanece como um objetivo inatingido, apesar dos grandes esforços já realizados neste assunto. Observa-se, deste modo, que ainda existe uma grande diferença entre a capacidade de leitura humana e a capacidade de leitura obtida pelas máquina. Grandes esforços devem ainda serem dispendidos, com o objetivo de que tais diferenças sejam minimizadas.

1.12 - RESUMO DOS CAPÍTULOS DESENVOLVIDOS NESTE TRABALHO

O **capítulo 2** foi dedicado ao reconhecimento de dígitos manuscritos isolados insensíveis a mudança de escala, translação e rotação, utilizando redes neuronais. Neste capítulo é descrita a técnica utilizada para tornar os dígitos isolados invariantes a mudança de escala, translação e rotação, são realizados comentários a respeito do algoritmo da rede neuronal utilizada, são apresentados os resultados e conclusões obtidas nos testes realizados com essa rede.

No **capítulo 3** são explicados os procedimentos utilizados na leitura e segmentação da imagem digitalizada, são descritos os algoritmos de afilamento OPTA, PSTA e o algoritmo rápido de afilamento (ARA) desenvolvido neste trabalho, são apresentados os resultados e as conclusões obtidas nos testes realizados com estes algoritmos.

No **capítulo 4** é descrito o algoritmo desenvolvido neste trabalho para o reconhecimento de caracteres numéricos manuscritos isolados, são apresentados os resultados e as conclusões obtidas nos testes realizados com este algoritmo.

No **capítulo 5** são explicadas as modificações realizadas no algoritmo desenvolvido no capítulo 4 para que esse algoritmo reconhecesse a quantia numérica de cheques bancários. São apresentadas as janelas de entrada e saída de

dados do programa, os resultados e as conclusões obtidas após as modificações realizadas no algoritmo desenvolvido no capítulo 4.

No **capítulo 6** são apresentadas as conclusões dos algoritmos desenvolvidos, as contribuições deste trabalho e as sugestões para que futuros trabalhos possam ser desenvolvidos.

CAPÍTULO 2

RECONHECIMENTO DE DÍGITOS MANUSCRITOS ISOLADOS INSENSÍVEIS A MUDANÇA DE ESCALA, TRANSLAÇÃO E ROTAÇÃO UTILIZANDO-SE REDES NEURONAIS

2.1 - INTRODUÇÃO

Este capítulo trata do reconhecimento de dígitos manuscritos isolados, utilizando-se redes neuronais. Com o objetivo de obter resultados satisfatórios, os dígitos antes de serem treinados e reconhecidos pela rede são normalizados, tornando-se insensíveis a mudança de escala, translação e rotação.

Recentemente, a utilização de redes neuronais no reconhecimento de caracteres e tipos diferentes de padrões tem ressurgido [11].

Os métodos de reconhecimento de padrões convencionais requerem que os dados de entrada apresentem posição, orientação e tamanho similares a um

determinado padrão. Na prática, porém, os dados de entrada podem apresentar diferenças de tamanho, posição ou rotação em relação ao padrão pré-determinado. Além disso, os dados de entrada podem ser corrompidos por ruído. Os métodos convencionais de reconhecimento de padrões (o casamento de padrões, por exemplo) não trabalham bem sob essas condições.

Como citado por Fukumi e autores [12], na área de reconhecimento de padrões, foram apresentados vários sistemas que utilizam redes neuronais e são insensíveis a transformações na entrada. Dentre esses sistemas pode-se citar: o "Neocognitron" apresentado por Fukushima, que é invariante à translação e a deformação do padrão de entrada; um sistema auto organizante proposto por Carpenter, que classifica padrões pela teoria da ressonância adaptativa; um sistema de reconhecimento de padrões composto por redes ADALINE proposto por Widrow, que é insensível à translação e rotação de 90° do padrão de entrada; Koch também descreveu um método similar ao de Widrow; um sistema tolerante a ruído que é uma versão extendida do trabalho de Widrow foi apresentado por Fukumi e autores, nesse sistema, unidades neuronais sigmoidais são adotadas e treinadas pelo algoritmo Back-propagation e um sistema que é invariante a translação, rotação e a mudança de escala do padrão de entrada utilizando como pré-processador uma rede neuronal de mais alta ordem foi construído por Reid. Entretanto, isto, resultou numa explosão combinatorial de unidades. Apesar de vários outros sistemas de reconhecimento de padrões terem sido propostos, ainda não há um sistema útil, insensível a rotação, translação e a mudança de escala.

Outra abordagem para o reconhecimento invariante de padrões [12] é o uso do critério de similaridade definido em termos de certas funções de transformações matemáticas, por exemplo, a transformada de Fourier, a transformada de Hough e a invariância do momento utilizada como pré-processador. O vetor resultante da aplicação da transformada pode servir de

entrada para as redes neuronais. Entretanto, se um padrão de entrada é uma imagem com muitos níveis de tons de cinza, a detecção de bordas seria necessária como pré-processador em vários casos. Ainda, em casos onde funções como a transformada de Fourier são usadas como pré-processadores, seguidos pela alimentação da rede, uma rede em grande escala seria necessária e isto resultaria em um aumento considerável no tempo de treinamento [12].

Neste capítulo, descreve-se a técnica utilizada para tornar os dígitos isolados invariantes a mudança de escala, translação e rotação. Comenta-se a respeito do algoritmo utilizado para a implementação da rede neuronal e sobre os dados utilizados para testar este algoritmo. Apresenta-se os resultados obtidos e as conclusões obtidas a respeito destes resultados.

2.2 - MUDANÇA DE ESCALA, TRANSLAÇÃO E ROTAÇÃO

Os resultados obtidos com a utilização de redes neuronais não são satisfatórios quando ocorrem diferenças expressivas entre os padrões utilizados no treinamento e os padrões utilizados no reconhecimento. As diferenças que podem ocorrer entre esses padrões são: mudança de escala, translação e rotação. A mudança de escala ocorre quando as dimensões do padrão utilizado no treinamento são diferentes das dimensões do padrão utilizado no reconhecimento. A translação ocorre quando o padrão utilizado no treinamento encontra-se localizado em uma posição distinta à posição do padrão utilizado no reconhecimento. A rotação ocorre quando os padrões utilizados no treinamento e no reconhecimento apresentam inclinações diferentes. Com o objetivo de minimizar essas diferenças, os dígitos utilizados antes de serem treinados e reconhecidos pela rede neuronal são normalizados, tornando-se insensíveis a mudança de escala, translação e rotação.

Na normalização de um dígito são realizados os processos de mudança de escala, translação e rotação. O algoritmo desenvolvido neste trabalho [13, 14] que torna o dígito invariante a mudança de escala, translação e rotação é descrito nas próximas sessões.

2.2.1 - Invariância a Mudança de Escala

Os "pixels-on" ("pixel" com valor igual a 1) e os "pixels-off" ("pixel" com valor igual a 0) que constituem a imagem do dígito são armazenados em uma matriz, denominada no programa de matriz-dígito, de dimensões n_l x n_c onde:

n_l - número de linhas da matriz-dígito.
n_c - número de colunas da matriz-dígito.

Para que o algoritmo[13, 14] realize a ampliação ou redução da imagem, são fornecidas as dimensões n_{ln} e n_{cn}, onde:

n_{ln} - número de linhas da imagem na nova escala.
n_{cn} - número de colunas da imagem na nova escala.

Com n_l, n_c, n_{ln} e n_{cn} disponíveis, calcula-se as seguintes razões:

$$e_l = \frac{n_{ln}}{n_l} \qquad (1)$$

$$e_c = \frac{n_{cn}}{n_c} \qquad (2)$$

onde:

e_l - fator de escala das linhas da imagem.

e_c - fator de escala das colunas da imagem.

O fator de escala final (e_f) que a imagem do dígito fica submetida é calculado do seguinte modo:

- Se o valor absoluto da diferença entre e_l e e_c for menor do que 0.4, o e_f será igual a média entre e_l e e_c.

- Se e_l e e_c forem maiores do que 1.0, o e_f será o menor dos dois termos.

- Se e_l e e_c forem menores ou iguais a 1.0, o e_f será o menor dos dois termos.

- Se o e_l for maior ou igual a 2.0, o e_f será igual a e_c Caso contrário, o e_f será igual a e_l.

A imagem do dígito invariante a mudança de escala será:

$$f_E(x_i, y_j) = f(e_f.x_i, e_f.y_j) \qquad (3)$$

onde:

$f(x_i, y_j)$ - fornece o valor do "pixel" ("pixel-on" ou "pixel-off") da coordenada (x_i, y_j).

$f_E(x_i, y_j)$ - fornece o valor do "pixel" ("pixel-on" ou "pixel-off") da coordenada (x_i, y_j) invariante a mudança de escala.

Os "pixels-on" e os "pixels-off" que constituem a imagem do dígito insensível a mudança de escala são armazenados na matriz denominada no programa de matriz-escala.

2.2.2 - Invariância a Translação

O processo de translação é realizado utilizando-se a matriz-escala que armazena a imagem do dígito insensível a mudança de escala.

Para que a imagem do dígito torne-se insensível à translação é calculado o centro de gravidade da imagem, utilizando-se a matriz-escala. O centro de gravidade é deslocado para a origem do novo sistema de coordenadas, que corresponde ao centro de gravidade da matriz-dígito [13, 14].

O centro de gravidade da imagem insensível a mudança de escala é calculado pela média das coordenadas x e y dos "pixels-on", de acordo com as fórmulas abaixo:

$$P = \sum_{i=1}^{N} \sum_{j=1}^{N} f_E(x_i, y_j) \qquad (4)$$

onde:

P - a soma de todos os "pixels-on" da matriz-escala.

N - o número de "pixels-on" da imagem insensível a mudança de escala.

O centro de gravidade da imagem insensível a mudança de escala é:

$$x_{cg} = \frac{1}{P} \sum_{i=1}^{N} \sum_{j=1}^{N} f_E(x_i, y_j).x_i \quad (5)$$

$$y_{cg} = \frac{1}{P} \sum_{i=1}^{N} \sum_{j=1}^{N} f_E(x_i, y_j).y_j \quad (6)$$

A imagem do dígito invariante a mudança de escala é deslocada para o centro de gravidade da matriz-dígito. Esse deslocamento é calculado da seguinte maneira:

$$x_{cgm} = x_{cg} - x_o \quad (7)$$

$$y_{cgm} = y_{cg} - y_o \quad (8)$$

onde:

x_o - abscissa do centro de gravidade da matriz-dígito.

y_o - ordenada do centro de gravidade da matriz-dígito.

x_{cgm} - abscissa do centro de gravidade da matriz-escala-translação (esta matriz será definida adiante).

y_{cgm} - ordenada do centro de gravidade da matriz-escala-translação.

A imagem do dígito invariante a mudança de escala e a translação será:

$$f_{ET}(x_i, y_j) = f_E(x_i + x_{cgm}, y_j + y_{cgm}) \quad (9)$$

onde:

$f_{ET}(x_i, y_j)$ fornece o valor do "pixel" ("pixel-on" ou "pixel-off") da coordenada (x_i, y_j) invariante a mudança de escala e a translação.

Os "pixels-on" e os "pixels-off" que constituem a imagem do dígito insensíveis a mudança de escala e a translação são armazenados na matriz denominada no programa de matriz-escala-translação.

2.2.3 - Invariância a Rotação

A rotação é realizada utilizando-se a matriz-escala-translação que armazena os "pixels-on" e os "pixels-off" que constituem a imagem do dígito insensível a mudança de escala e translação.

Ao rotacionar a imagem do dígito na direção de máxima variância, que coincide com o eixo x, a imagem torna-se invariante a rotação. A derivação da função que torna a imagem invariante a rotação é baseada na transformação de Karhunen-Loève, que tem sido usada em algumas aplicações [15, 16]. Essa transformação é explicada do seguinte modo: dado um conjunto de vetores, o autovetor que corresponde ao maior autovalor da matriz de covariância, calculada desse conjunto de vetores, aponta na direção da máxima variância [15, 16]. Essa propriedade pode ser usada para manter a invariância a rotação, visto que a detecção da direção de variância máxima revela o ângulo de rotação. A solução geral para qualquer tamanho de vetores é impraticável, porém, para vetores de duas dimensões, formados pelas coordenadas x e y dos "pixels-on", os autovalores são fáceis de calcular e uma fórmula para o autovetor que corresponde ao maior autovalor pode ser derivada da matriz de covariância 2 x 2.

Os parâmetros da rotação são derivados como segue:

Define-se:

$$\begin{bmatrix} m_x \\ m_y \end{bmatrix} = \frac{1}{\sum_{i=1}^{N}\sum_{j=1}^{N} f_{ET}(x_i, y_j)} \sum_{i=1}^{N}\sum_{j=1}^{N} f_{ET}(x_i, y_j).\begin{bmatrix} x_i \\ y_j \end{bmatrix} \tag{10}$$

$$P = \sum_{i=1}^{N}\sum_{j=1}^{N} f_{ET}(x_i, y_j) \tag{11}$$

$$T_{xx} = \sum_{i=1}^{N}\sum_{j=1}^{N} f_{ET}(x_i, y_j).x_i^2 \tag{12}$$

$$T_{yy} = \sum_{i=1}^{N}\sum_{j=1}^{N} f_{ET}(x_i, y_j).y_j^2 \tag{13}$$

$$T_{xy} = \sum_{i=1}^{N}\sum_{j=1}^{N} f_{ET}(x_i, y_j).x_i.y_j \tag{14}$$

A matriz de covariância é definida como:

$$C = \frac{1}{P}\left(\begin{bmatrix} x \\ y \end{bmatrix} - \begin{bmatrix} m_x \\ m_y \end{bmatrix}\right).\left(\begin{bmatrix} x \\ y \end{bmatrix} - \begin{bmatrix} m_x \\ m_y \end{bmatrix}\right)^T \tag{15}$$

Que pode ser simplificada para:

$$C = \left(\frac{1}{\sum_{i=1}^{N}\sum_{j=1}^{N} f_{ET}(x_i, y_j)} \sum_{i=1}^{N}\sum_{j=1}^{N} f_{ET}(x_i, y_j) \begin{bmatrix} x_i \\ y_j \end{bmatrix} \cdot \begin{bmatrix} x_i \\ y_j \end{bmatrix}^T \right) - \left(\begin{bmatrix} m_x \\ m_y \end{bmatrix} \cdot \begin{bmatrix} m_x \\ m_y \end{bmatrix}^T \right) \tag{16}$$

Desde que a invariância a translação tenha sido mantida, as médias m_x e m_y serão zero. Além disso, o termo médio na frente da matriz pode ser eliminado desde que ele não mude a direção dos autovetores. Portanto, a matriz de covariância torna-se:

$$C = \begin{bmatrix} T_{xx} & T_{xy} \\ T_{xy} & T_{yy} \end{bmatrix} \tag{17}$$

Os autovalores (λ) dessa matriz são calculados de:

$$(\lambda I - C) = \begin{bmatrix} (\lambda - T_{xx}) & -T_{xy} \\ -T_{xy} & (\lambda - T_{yy}) \end{bmatrix} = 0 \tag{18}$$

$$(\lambda - T_{xx}).(\lambda - T_{yy}) - T_{xy}^2 = 0 \tag{19}$$

$$\lambda_{1,2} = \frac{T_{yy} + T_{xx} \pm \sqrt{((T_{yy} + T_{xx})^2 - 4.(T_{xx}.T_{yy} - T_{xy}^2))}}{2} \tag{20}$$

e os autovetores correspondentes podem então ser derivados de:

$$\begin{bmatrix} T_{xx} & T_{xy} \\ T_{xy} & T_{yy} \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix} = \lambda \cdot \begin{bmatrix} x \\ y \end{bmatrix} \tag{21}$$

$$T_{xx}.x + T_{xy}.y = \lambda.x \tag{22}$$

$$T_{xy}.x + T_{yy}.y = \lambda.y \tag{23}$$

$$\frac{y}{x} = \frac{\lambda - T_{xx}}{T_{xy}} \tag{24}$$

Substituindo o maior autovalor, obtém-se a inclinação do autovetor, segundo:

$$\frac{y}{x} = \frac{T_{yy} - T_{xx} + \sqrt{(T_{yy} - T_{xx})^2 + 4.T_{xy}^2}}{2.T_{xy}} \tag{25}$$

Portanto, o seno e o cosseno para esta inclinação podem ser derivados de:

$$sen\theta = \frac{y}{\sqrt{(x^2 + y^2)}} \tag{26}$$

$$sen\theta = \frac{(T_{yy} - T_{xx}) + \sqrt{((T_{yy} - T_{xx})^2 + 4.T_{xy}^2}}{\sqrt{(8.T_{xy}^2 + 2.(T_{yy} - T_{xx})^2 + 2.(T_{yy} - T_{xx})\sqrt{((T_{yy} - T_{xx})^2 + 4.T_{xy}^2)}}} \tag{27}$$

$$\cos\theta = \frac{x}{\sqrt{(x^2 + y^2)}} \tag{28}$$

$$\cos\theta = \frac{2.T_{xy}}{\sqrt{(8.T_{xy}^2 + 2.(T_{yy} - T_{xx})^2 + 2.(T_{yy} - T_{xx})\sqrt{((T_{yy} - T_{xx})^2 + 4.T_{xy}^2)}}} \tag{29}$$

A imagem invariante a mudança de escala, translação e rotação será:

$$f_{ETR}(x_i, y_j) = f_{ET}(\cos\theta.x_i - sen\theta.y_j, sen\theta.x_i + \cos\theta.y_j) \tag{30}$$

onde $f_{ETR}(x_i, y_j)$ fornece o valor do "pixel" ("pixel-on" ou "pixel-off") da coordenada (x_i, y_j) invariante a mudança de escala, translação e rotação.

Os "pixels-on" e os "pixels-off" que constituem a imagem do dígito insensível a mudança de escala, translação e rotação são armazenados na matriz denominada no programa de matriz-escala-translação-rotação.

2.3 - REDES NEURONAIS

Redes neuronais artificiais são modelos computacionais inspirados na estrutura do cérebro humano [15]. Uma rede neuronal consiste de várias camadas de neurônios distintos. A camada de entrada, utiliza neurônios sensores s_i para monitorar sinais externos, enquanto a camada de saída u_j transmite sinais para o mundo externo.

Uma rede pode conter uma ou mais camadas escondidas. Unidades

escondidas h_k são usadas pela rede, para representar o domínio de conhecimento e para resolver problemas. A resolução de problemas, pode ser vista como uma manipulação de símbolos ou como um mapeamento espacial, dependendo para isso se a ligação ou as propriedades analógicas dos neurônios serão enfatizadas [17]. A figura 2.1 ilustra uma rede neuronal básica com uma única camada escondida.

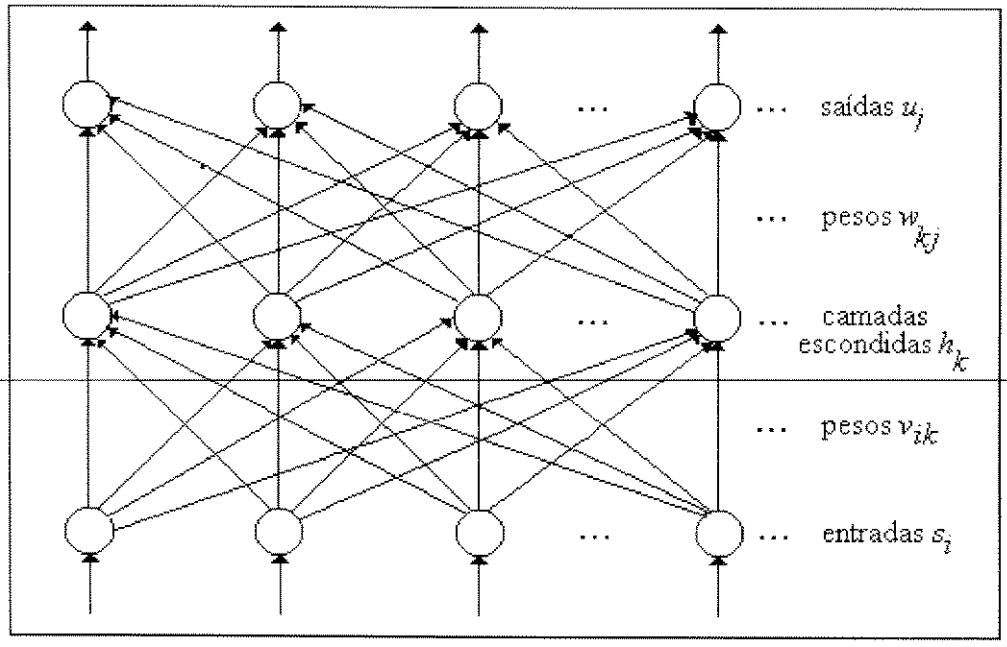


Figura 2.1 - Rede neuronal com 3 camadas: entradas s_i, camada escondida h_k e unidades de saída u_j Pesos v_{ik} conectam as entradas às unidades escondidas, e pesos w_{kj} conectam a camada escondida às unidades de saída.

Os neurônios se comunicam por sinais analógicos através das ligações sinápticas ilustradas na figura 2.1. Em geral, todos os neurônios em uma camada são completamente interconectados aos neurônios em camadas adjacentes. A informação flue unidirecionalmente da entrada, através da camada escondida, para as camadas de saída. Porém, durante a fase de treinamento, a informação flue na direção oposta. A cada sinapse está associado um peso v_{ik} que conecta o neurônio de entrada i ao neurônio escondido k e um peso w_{kj} que conecta o neurônio escondido k ao neurônio de saída j. Estes pesos medem o grau de

correlação entre níveis de atividade dos neurônios a eles conectados [17].

Cada célula do neurônio recebe um sinal da rede que é a soma linear ponderada de todas as suas entradas. Uma ativação logística ou função de saída $1/(1 + e^{-x})$ converte a soma linear ponderada para uma aproximação suave do clássico neurônio de passo de McCulloch e Pitts [17]. Outras funções monotônicas similares a logísticas podem também ser usadas. Portanto, o algoritmo de aprendizagem requer que elas sejam diferenciáveis. A saída h_k do neurônio escondido k é fornecida por:

$$h_k = \frac{1}{1 + e^{-\sum_i v_{ik} s_i}} \tag{31}$$

Similarmente, a camada de saída u_j do neurônio de saída j é fornecida por:

$$u_j = \frac{1}{1 + e^{-\sum_k w_{kj} h_k}} \tag{32}$$

O algoritmo utilizado para o reconhecimento de dígitos manuscritos isolados foi o Back-propagation [17]. O algoritmo Back- propagation é um processo de treinamento iterativo no qual o sinal erro de saída é propagado no sentido das entradas através da rede, e é usado para modificar os valores dos pesos [17]. Desde que os pesos da rede são inicialmente indeterminados, um processo de treinamento é necessário para estabelecer seus valores.

O sinal erro de saída ε do algoritmo Back-propagation é definido por:

$$\varepsilon = \frac{1}{2}\sum_j (t_j - u_j)^2 \tag{33}$$

onde tj é o valor desejado ou valor alvo da saída u_j para um dado padrão de entrada e a somatória é calculada sobre todos os nós de saída j. A derivada parcial do erro com relação ao peso da rede define uma regra de atribuição de valores que é usada para modificar o peso. Ela atribui valor, proporcionalmente, aqueles responsáveis pelo erro, implementando um gradiente descendente no espaço de pesos. Pode ser visto como uma aplicação direta do aprendizado de Hebb ou uma generalização da regra delta perceptron [17].

O treinamento inicia-se apresentando uma amostra padrão para os sensores de entrada de uma rede primária com pesos iniciais aleatórios (no programa deste trabalho foram gerados, inicialmente, pesos aleatórios de -1 a 1).

Pesos w_{kj} são alterados de acordo com a equação 34 :

$$\Delta w_{kj}(n) = \eta \delta_j h_k + \alpha \Delta w_{kj}(n-1) \tag{34}$$

onde: $\delta_j = u_j(1-u_j).(t_j - u_j)$ η é a taxa de aprendizagem e α é um fator de suavização que reduz mudanças agudas no espaço de pesos. Uma fração α da mudança de peso da iteração anterior, é somada a estimativa atual.

Após calcular δ_j, na camada de saída, valores δ_k^* da camada escondida podem ser obtidos. Para um neurônio escondido, a regra muda de acordo com a equação 35:

$$\delta_k^* = h_k(1-h_k)\sum_j \delta_j w_{kj} \tag{35}$$

onde h_k é a ativação de um neurônio escondido k e a somatória é sobre todos os neurônios j da camada de saída. A correção de peso para v_{ik} é, similarmente, igual a equação 34:

$$\Delta v_{ik}(n) = \eta \delta_k^* s_i + \alpha \Delta v_{ik}(n-1) \quad (36)$$

Iterações múltiplas de padrões de ativação e erro de realimentação são realizados com novos padrões de treinamento, até que o erro seja reduzido a um valor inferior ao limite pré-estabelecido [17] (no programa deste trabalho estabeleceu-se que este limite seria de 10^{-4}). Novos padrões, diferentes do conjunto de padrões utilizados no treinamento, podem ser empregados na fase de reconhecimento com o objetivo de testar a rede treinada.

2.4 - RESULTADOS OBTIDOS

O algoritmo explicado no item 2.2 foi testado com dígitos escritos por várias pessoas. Foi solicitado a cada uma destas pessoas que escrevesse em uma folha branca de papel sulfite e em apenas uma linha os dígitos de 0 a 9, separados uns dos outros (essa linha é considerada pelo algoritmo de segmentação, que será explicado no capítulo 3 deste trabalho, como uma linha de texto). Esse pedido foi feito a cada uma das pessoas, afim de evitar que os dígitos se sobrepusessem (dígitos sobrepostos são dígitos isolados que apesar de não se tocarem apresentam "pixels-on" na mesma coluna) ou se conectassem (dígitos conectados são dígitos que apresentam "pixels-on" em comum. Isto é, um ou mais "pixels-on" pertencem a dois dígitos distintos); definições mais detalhadas a respeito desses tipos de dígitos serão apresentadas no capítulo 5 deste trabalho. Os dígitos de 0 a 9 escritos em uma linha, são digitalizados e armazenados em um mesmo arquivo, que é gravado no formato tiff. Várias pessoas ao escreverem os dígitos de 0 a 9 utilizaram caneta azul, outras utilizaram caneta preta, ou mesmo caneta vermelha, etc. Independente da cor da caneta utilizada todos os dígitos foram digitalizados, utilizando-se um "scanner" monocromático.

Cada dígito utilizado na rede neuronal é normalizado utilizando-se a técnica apresentada no item 2.2 deste capítulo. Esse processo torna os dígitos insensíveis a mudança de escala, translação e rotação. Os "pixels-on" e os "pixels-off" que constituem a imagem do dígito normalizado são armazenados na matriz-escala-translação-rotação. E esta matriz é gravada em um arquivo. Ao terminar o processo de normalização e armazenamento da matriz-escala-translação-rotação, o arquivo conterá a matriz-escala-translação-rotação de cada dígito uma após a outra. O conjunto formado pelos "pixels-on" e pelos "pixels-off" da matriz-escala-translação-rotação de cada um dos 10 dígitos do arquivo constituem os neurônios de entrada da rede neuronal.

Nos primeiros testes realizados com a rede neuronal, a imagem de cada dígito foi armazenada em uma matriz-escala-translação-rotação de dimensões 30 x 30; como existem 10 dígitos em cada arquivo, o número total de neurônios de entrada a serem utilizados na rede é de 9000. Na fase de treinamento são utilizados os dígitos de 0 a 9 de cada arquivo, que constitui então um arquivo de treinamento. Como foram utilizados 10 dígitos no processo de treinamento, o número de neurônios de saída da rede é 10. Foi estabelecido que o número de neurônios da camada escondida a serem utilizados seria igual a 20. Todos os testes foram realizados utilizando-se uma estação de trabalho Risc System/6000 da IBM.

A figura 2.2 mostra 10 dígitos de uma pessoa, antes de serem submetidos aos processos de mudança de escala, translação e rotação e a figura 2.3 mostra esses dígitos, após serem submetidos aos processos de mudança de escala, translação e rotação. Esses dígitos serão utilizados na fase de treinamento ou na fase de reconhecimento.

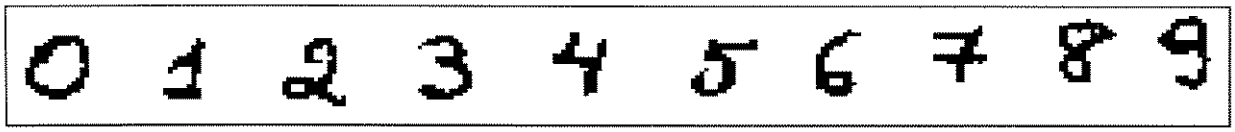


Figura 2.2 - Dígitos de uma pessoa antes de serem submetidos aos processos de mudança de escala, translação e rotação.

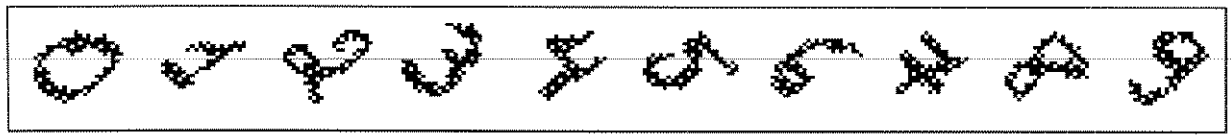


Figura 2.3 - Dígitos de uma pessoa depois de serem submetidos aos processos de mudança de escala, translação e rotação.

Foram realizados 3 tipos de testes com a imagem de cada dígito armazenada em uma matriz-escala-translação-rotação de dimensões 30 x 30. O primeiro teste foi realizado com os dados escritos por apenas uma pessoa. Foram utilizados no treinamento 10 arquivos, onde cada um destes arquivos continham 10 dígitos dessa pessoa. Trinta outros arquivos da mesma pessoa, porém, diferentes dos arquivos utilizados no treinamento foram utilizados no reconhecimento. A figura 2.4 e a tabela 2.1 apresentam os resultados obtidos neste teste, para um total de 300 dígitos utilizados no reconhecimento.

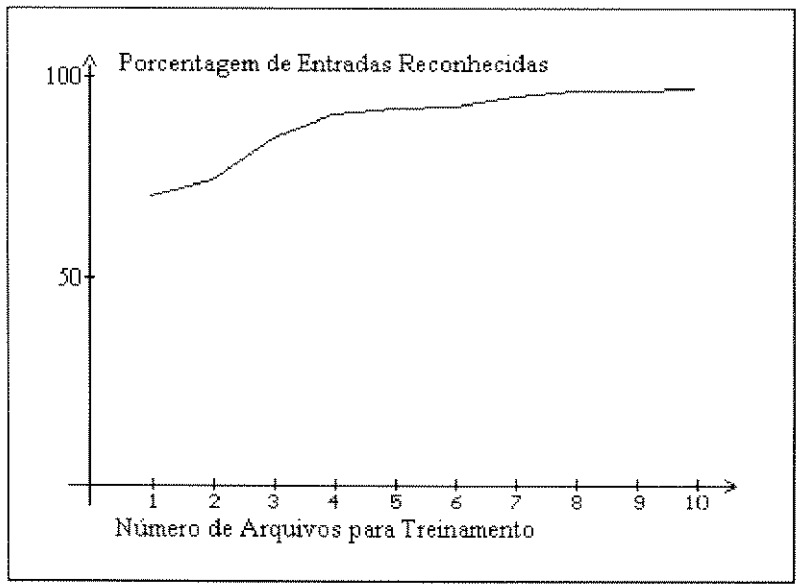


Figura 2.4 - Gráfico da porcentagem de entradas reconhecidas x número de arquivos de uma pessoa, utilizados na fase de treinamento.

Tabela 2.1 - Porcentagem de entradas reconhecidas ao serem utilizados arquivos de uma pessoa.

Número de Arquivos para Treinamento	Número de Entradas Reconhecidas	Porcentagem de Entradas Reconhecidas
1	208	69,30%
2	219	73,00%
3	265	83,30%
4	266	88,87%
5	271	90,33%
6	272	90,67%
7	280	93,33%
8	284	94,67%
9	283	94,33%
10	285	95,00%

Ao analisar os resultados da figura 2.4 e da tabela 2.1 pode-se verificar que quanto maior o número de arquivos utilizados na fase de treinamento, maior é a porcentagem de entradas reconhecidas.

O segundo teste foi realizado com os arquivos de 4 pessoas distintas. Inicialmente, na fase de treinamento foi utilizado um arquivo de cada pessoa, contendo 10 dígitos de 0 a 9. Na fase de reconhecimento foram utilizados outros 10 arquivos das mesmas pessoas, o que resultou num total de 400 dígitos a serem reconhecidos. Para este caso a porcentagem de reconhecimento foi de 56,25%. O mesmo teste foi realizado utilizando-se no treinamento 2, 3 e 4 arquivos de cada uma das 4 pessoas acima mencionadas. A porcentagem de dígitos reconhecidos aumentou gradativamente atingindo um máximo de 71,00% de acerto, quando

foram utilizados, na fase de treinamento, 4 arquivos de cada pessoa. Em todos os testes realizados, foram utilizados na fase de reconhecimento 10 arquivos, contendo cada um 10 dígitos de 0 a 9 de cada uma das 4 pessoas. Os resultados obtidos são mostrados na figura 2.5 e na tabela 2.2.

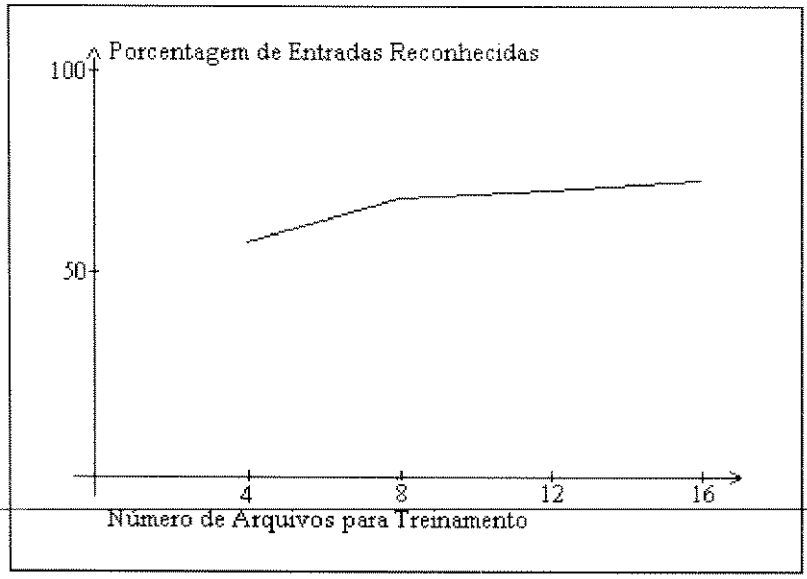


Figura 2.5 - Gráfico da porcentagem de entradas reconhecidas x número de arquivos de 4 pessoas distintas, utilizados na fase de treinamento.

Tabela 2.2 - Porcentagem de entradas reconhecidas ao serem utilizados arquivos de 4 pessoas distintas.

Número de Arquivos para Treinamento	Número de Entradas Reconhecidas	Porcentagem de Entradas Reconhecidas
4	225	56,25%
8	267	66,75%
12	276	69,00%
16	284	71,00%

O terceiro teste foi realizado treinando-se a rede neuronal com 4 arquivos, onde cada um desses arquivos continha 10 dígitos de 0 a 9, de 4 pessoas

distintas. Na fase de reconhecimento foram utilizados um total de 40 arquivos, onde os dígitos contidos nesses arquivos foram escritos por pessoas diferentes das pessoas que escreveram os dígitos utilizados na fase de treinamento. Estes testes foram realizados, com o intuito de se obter um sistema de reconhecimento que fosse capaz de reconhecer o dígito de uma pessoa que não havia participado da fase de treinamento da rede neuronal. Nesse terceiro teste, a porcentagem de dígitos reconhecidos caiu bastante, atingindo um valor igual a 57,75%. Em todos os testes citados anteriormente, foram utilizados dígitos, insensíveis a mudança de escala, translação e rotação; armazenados em matrizes-escala-translação-rotação de dimensões 30 x 30.

Outros testes foram realizados, utilizando-se 5 arquivos de uma única pessoa na fase de treinamento e 10 arquivos da mesma pessoa na fase de reconhecimento; porém, esses arquivos eram distintos dos arquivos utilizados no treinamento. Cada arquivo continha 10 dígitos de 0 a 9, armazenados em matrizes-escala-translação-rotação de dimensões 20 x 20. Ao serem realizados estes testes, percebeu-se uma diminuição na porcentagem de dígitos reconhecidos. Essa porcentagem diminuiu de 95,00% (quando foram utilizados 10 arquivos para treinamento de uma única pessoa e 30 arquivos para reconhecimento da mesma pessoa, onde a imagem de cada dígito foi armazenada na matriz-escala-translação-rotação de dimensões 30x30) para 88,00% nesse caso. Para as mesmas condições citadas acima (5 arquivos de uma única pessoa na fase de treinamento e 10 arquivos da mesma pessoa na fase de reconhecimento), porém, a imagem de cada dígito sendo armazenada em uma matriz-escala-translação-rotação de dimensões 30 x 30, a porcentagem de acerto caiu para 90,33%. Quando a imagem de cada dígito foi armazenada em uma matriz-escala-translação-rotação de dimensões 40 x 40, realizando-se os mesmos testes anteriores (5 arquivos de uma única pessoa na fase de treinamento e 10

arquivos da mesma pessoa, porém distintos dos arquivos utilizados no treinamento, na fase de reconhecimento), a porcentagem de acerto foi de 93,00%.

2.5 - CONCLUSÕES

A rede neuronal apresentou uma porcentagem de dígitos reconhecidos, igual a 95,00% quando foram utilizados arquivos de uma única pessoa nas fases de treinamento e reconhecimento. Como era de se esperar quanto maior o número de arquivos utilizados na fase de treinamento, melhores são os resultados obtidos na fase de reconhecimento. Pode-se observar que ao serem utilizados arquivos de 4 pessoas distintas, também são obtidos melhores resultados quando é aumentado o número de arquivos utilizados na fase de treinamento. Porém, quando foram utilizados na fase de reconhecimento 40 arquivos com dígitos escritos por pessoas que não eram as pessoas que escreveram os dígitos utilizados na fase de treinamento, a porcentagem de reconhecimento caiu bastante, ou seja, de 71,00% para 57,75%. Na utilização de dígitos cuja a imagem foi armazenada na matriz-escala-translação-rotação de dimensões 20 x 20 a porcentagem de reconhecimento diminuiu em relação aos dígitos cuja imagem foi armazenada na matriz-escala-translação-rotação de dimensões 30 x 30. Como era de se esperar, isso ocorreu devido à menor quantidade de "pixels-on" e "pixels-off" utilizados na imagem, ou seja, menor quantidade de infomações disponíveis na matriz-escala-translação-rotação na fase de treinamento. Quando as dimensões da imagem de cada dígito armazenado na matriz-escala-translação-rotação foi aumentada para 40 x 40, pode-se verificar que a porcentagem de dígitos reconhecidos aumentou, como era de se esperar. Isto ocorreu, porque na fase de treinamento foram utilizados mais "pixels-on" e mais "pixels-off" na matriz-escala-translação-rotação.

O aumento de variáveis de entrada ("pixels-on" e "pixels-off", no caso de imagens) nem sempre fornece melhor desempenho, especialmente no que se refere à habilidade de generalizar. A escolha das variáveis de entrada é, sim, decisiva. Com a expansão da imagem (uso de mais "pixels-on" e mais "pixels-off"), uma topologia que utiliza todas as variáveis de entrada tem mais chances de encontrar entradas significativas. Portanto, não se deve esperar a prevalência deste tipo de resultado em outros cenários. Os resultados são, contudo, interessantes e elucidadores para o caso em questão.

Na utilização de redes neuronais para o reconhecimento de dígitos manuscritos isolados insensíveis a mudança de escala, translação e rotação, os resultados mostraram-se satisfatórios, mas para que a taxa de reconhecimento estivesse acima de 90% foi necessário aumentar a quantidade de arquivos utilizados na fase de treinamento da rede. Esse aumento resultou em uma elevação considerável no tempo de processamento da rede neuronal na fase de treinamento. Ao serem utilizados na fase de reconhecimento padrões escritos por pessoas diferentes das pessoas que escreveram os padrões utilizados na fase de treinamento, a taxa de reconhecimento apresentou expressiva redução; sendo que em alguns casos essa taxa chegou a ser igual a 57,75%. Devido a esse baixo percentual de acerto e aos problemas encontrados na fase de reconhecimento quando ocorrem diferenças expressivas entre o padrão utilizado na fase de treinamento e o padrão utilizado na fase de reconhecimento, resolveu-se desenvolver um novo algoritmo de reconhecimento de caracteres manuscritos que é totalmente diferente do algoritmo de reconhecimento utilizado na rede neuronal.

O processo utilizado pelo novo algoritmo de reconhecimento de caracteres manuscritos que é a base deste trabalho tem como princípio a verificação das características intrínsecas de cada dígito e a verificação das características que os dígitos têm em comum. Ao analisar estas características, esse algoritmo procura

aproximar-se do processo de reconhecimento utilizado pelo ser humano. Esse algoritmo será explicado nos capítulos 4 e 5, deste trabalho. Devido a essa resolução, de desenvolver um algoritmo que tem como base a verificação das características intrínsecas de cada dígito e a verificação das características que os dígitos têm em comum, não foram realizados testes com a rede neuronal em que os dígitos encontram-se sobrepostos e conectados.

CAPÍTULO 3

SEGMENTAÇÃO E UM ALGORITMO DE AFILAMENTO

3.1- INTRODUÇÃO

No primeiro capítulo deste trabalho foi comentado que existem várias formas de se implementar um OCR. O diagrama de blocos do OCR implementado neste trabalho foi mostrado na figura 1.1 do capítulo 1. Este OCR é composto dos seguintes blocos: Digitalização, Segmentação, Pré-processamento e Reconhecimento do texto.

No primeiro bloco do OCR, utilizando-se um "scanner" e o sistema Deskscan, que funciona no ambiente Windows, a imagem é digitalizada e gravada em um arquivo.

A imagem anteriormente gravada é lida por um programa que armazena os seus "pixels-on" e "pixels-off" em uma matriz na memória do computador.

No segundo bloco do OCR, a imagem que foi armazenada na matriz é

segmentada em 2 componentes básicos: linhas do texto e os caracteres que compõem cada uma destas linhas.

No terceiro bloco, o pré-processamento realiza [4] a: filtragem, restauração, realce e afilamento da imagem. Neste trabalho, foi desenvolvido "Um algoritmo rápido de afilamento" (ARA) [18]. Este algoritmo realiza o afilamento da imagem e a remoção de ruídos que podem ter ocorrido quando a imagem foi digitalizada. Não houve a necessidade de realizar a restauração das imagens utilizadas, pois as imagens digitalizadas pelo "scanner" apresentaram qualidade satisfatória. O realce da imagem foi obtido utilizando-se os recursos existentes no próprio sistema Deskscan.

O quarto bloco do OCR realiza o reconhecimento dos caracteres. O algoritmo de reconhecimento utilizado neste bloco foi desenvolvido neste trabalho e será explicado no capítulo 4 e no capítulo 5.

Neste capítulo, são descritos os procedimentos adotados para a leitura do arquivo onde está gravada a imagem digitalizada e a segmentação do texto. São também descritos os algoritmos de afilamento OPTA e PSTA que foram implementados com o objetivo de comparar os resultados obtidos pelo algoritmo ARA. São realizados alguns comentários a respeito dos dados utilizados para testar o algoritmo de afilamento desenvolvido. São apresentados alguns dos resultados obtidos nos testes com os algoritmos de afilamento OPTA, PSTA e ARA e, finalmente, são feitas as conclusões a respeito dos resultados obtidos com estes algoritmos de afilamento.

3.2 - LEITURA DO ARQUIVO ONDE ESTÁ GRAVADA A IMAGEM DIGITALIZADA

Para a digitalização e posterior gravação da imagem do texto, neste trabalho foi utilizado o "scanner" de mesa HP modelo Scanjet II p

monocromático e o programa Deskscan for Windows.

No processo de gravação do arquivo da imagem, o programa Deskscan pode gerar arquivos nos formatos tiff, pcx, bmp, etc. Dentre estes formatos, o utilizado neste trabalho foi o formato tiff. O motivo da utilização deste formato deve-se ao fato que na Universidade encontra-se disponível o manual [19], contendo informações referentes a este formato.

Arquivos no formato tiff apresentam no cabeçalho ("header" do arquivo) os bytes dispostos na seguinte ordem:

i) Bytes 0-1 indicam o formato no qual o cabeçalho e o IFD ("Image File Directory") do arquivo foram gravados. Se estes 2 bytes formam o número hexadecimal 4949, cada byte do cabeçalho e do IFD é gravado na ordem do byte menos significativo para o byte mais significativo. Porém, se estes 2 bytes formam o número hexadecimal 4D4D, cada byte é gravado na ordem do byte mais significativo para o byte menos significativo. Estes 2 critérios são estabelecidos para inteiros de 16 ou 32 bits, e não influem na ordem em que os bytes da imagem foram gravados.

ii) Bytes 2-3 indicam qual versão do formato tiff o arquivo foi gravado.

iii) Bytes 4-7 contém a posição (em bytes) do IFD. O IFD é um bloco de dados que contém informações referentes a imagem que está sendo gravada. Estas informações são: número de linhas, número de colunas, se a imagem foi gravada utilizando-se processos de compressão, padrão de cores utilizadas (que podem ser : preto e branco, tons de cinza ou colorido), palheta de cores utilizadas, etc. Cada uma dessas informações é gravada em campos individuais de tamanho igual a 12 bytes e dispostos de forma seqüencial. O inteiro formado pelos 2 primeiros bytes do IFD indica o número de campos que foram

gravados no IFD.

Os campos gravados no IFD possuem o seguinte formato:

- Bytes 0-1 contém o "TAG". O "TAG" é o código que indica qual informação está sendo gravada no presente campo. Este código é um número fornecido pelo manual [19].

- Bytes 2-3 informam se o valor contido nos bytes de 8-11 possuem o seguinte formato:

  1 = BYTE, número de 8 bits.

  2 = ASCII, número de 8 bits que armazena o código ASCII.

  3 = SHORT, número inteiro sem sinal de 16 bits.

  4 = LONG, número inteiro sem sinal de 32 bits.

  5 = RATIONAL (2 LONGs), o primeiro LONG representa o numerador e o segundo o denominador da fração.

- Bytes 4-7 podem ser considerados como um "offset", que indica se a informação está armazenada no presente campo, ou se é necessário a utilização de novos espaços para que a informação possa ser convenientemente guardada.

- Bytes 8-11 contém informações referentes a imagem, como por exemplo: número de linhas, número de colunas, padrões de cores, etc. São nestes bytes

que se verifica o "offset" (bytes de 4-7). Se o "offset" for igual a 0001, a informação está realmente contida nos bytes 8-11. Se o "offset" for um valor diferente de 0001, estes bytes contém o endereço da posição inicial onde a informação foi gravada.

A figura 3.1 ilustra o cabeçalho de um arquivo no formato tiff.

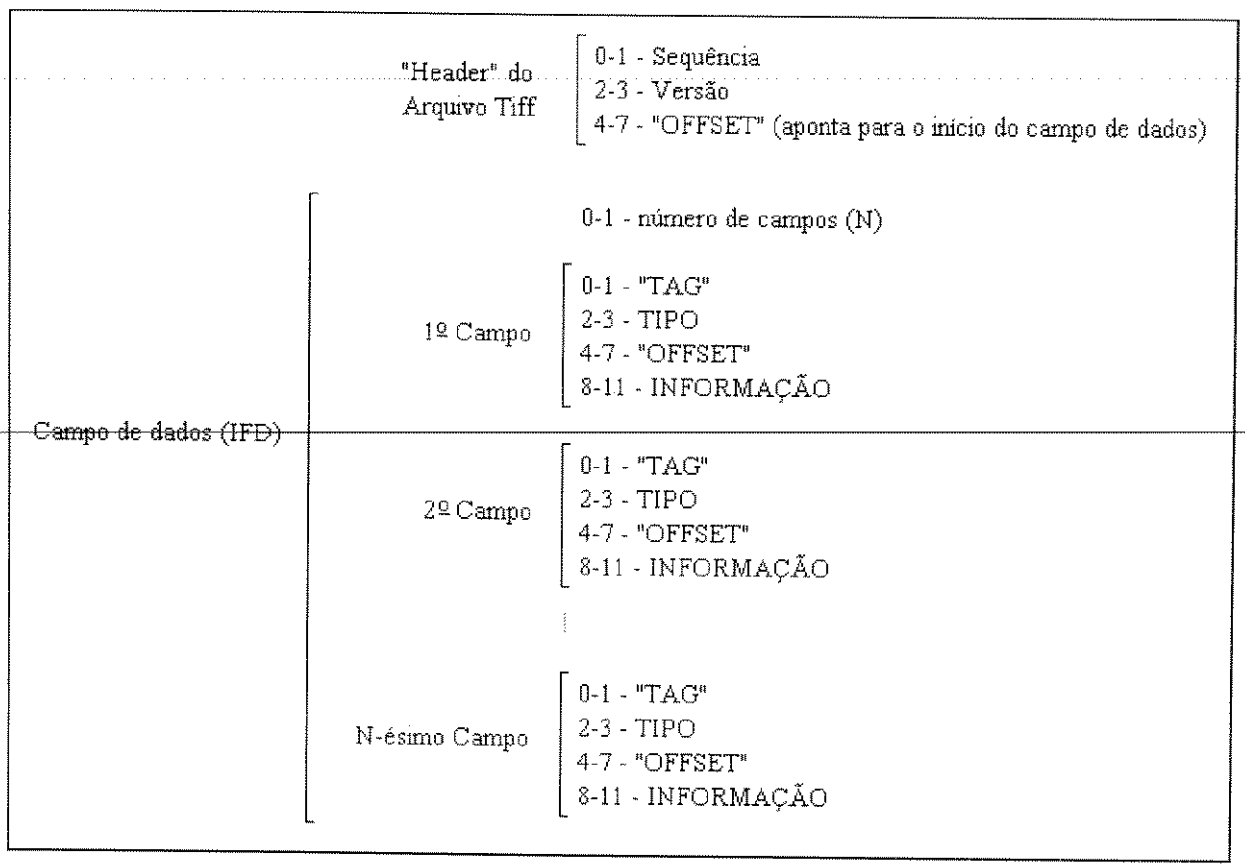


Figura 3.1 - Cabeçalho de um arquivo no formato tiff.

A seguir é descrito o algoritmo utilizado neste trabalho, que realiza a leitura de arquivos gravados no formato tiff.

O algoritmo lê os bytes 0-1 do "header", com o objetivo de identificar a seqüência em que os bytes foram gravados no "header" e no IFD. Um exemplo dessa identificação é o "TAG", que corresponde aos bytes 0-1 de cada campo. Se os bytes 0-1 do "header" formam o número hexadecimal 4949, o byte mais significativo do valor gravado no "TAG" foi armazenado no byte 0 do "TAG" e o

byte menos significativo deste valor foi armazenado no byte 1 do "TAG". Se os bytes 0-1 do "header" formam o número hexadecimal 4D4D o valor armazenado no "TAG" foi gravado de forma contrária ao valor armazenado no "TAG" quando os bytes 0-1 do "header" formam o número hexadecimal 4949.

Após identificar a seqüência em que os bytes do "header" e do IFD foram gravados, é realizada a leitura do "offset", com o objetivo de encontrar a posição inicial em que foi gravado o campo de dados (IFD). No campo de dados (IFD), o programa lê o valor armazenado nos bytes 0-1. Esse valor indica o número de campos que foram gravados no IFD.

Conhecendo o número de campos que foram gravados no IFD, o programa realiza a leitura desses campos e os armazena em estruturas adequadas. Estas estruturas são elaboradas com o objetivo de armazenar os campos "TAG", TIPO, "OFFSET" e INFORMAÇÃO, ilustrados na figura 3.1 deste capítulo.

Entre os diversos campos gravados no IFD, os principais campos a serem acessados pelo programa são:

- Campo com "TAG" = 256 contém o número de colunas da imagem (n_c).

- Campo com "TAG" = 257 contém o número de linhas da imagem (n_l).

- Campo com "TAG" = 273 contém a posição inicial da imagem.

Neste trabalho as imagens digitalizadas foram sempre gravadas em preto e branco. Por este motivo, o programa não acessa o campo com "TAG" = 258, que informa se a imagem digitalizada foi gravada preta e branca, colorida ou tons de cinza.

O manual informa que se o arquivo foi gravado no formato tiff, os bytes 2-3 do "header" deste arquivo, devem conter o número 42. O programa prevê que

se os bytes 2-3 não contiverem o número 42, a execução deve ser terminada imediatamente, pois como cita o manual este não será um arquivo no formato tiff.

Após obter o número de linhas (n_l), o número de colunas (n_c) e a posição do arquivo onde está o início da imagem, o programa armazena em uma matriz de dimensões n_l x n_c os "pixels-on" e os "pixels-off" que constituem a imagem. Em seguida é iniciado o processo de segmentação do texto, que é descrito a seguir. O "pixel-on" possui o valor 1 e o "pixel-off" possui o valor 0.

3.3 - SEGMENTAÇÃO DO TEXTO

Com o objetivo de reconhecer os caracteres que compõem o texto, o programa inicialmente realiza a segmentação das linhas que compõem o texto e posteriormente para cada linha segmentada, o programa segmenta seus caracteres e realiza o processo de reconhecimento.

O processo de segmentação das linhas de texto é realizado varrendo-se a imagem de cima para baixo e armazenando em uma matriz denominada de matriz-imagem os "pixels-on" e "pixels-off" que são encontrados. Este armazenamento é realizado de tal forma, que uma linha e uma coluna da matriz-imagem seja equivalente a uma linha e uma coluna da imagem.

Cada vez que uma linha de "pixels-on" e "pixels-off" é lida, o programa verifica a existência de "pixels-on" na mesma. No caso de existirem "pixels-on" na linha da imagem que está sendo varrida, o programa continua o processo de leitura e armazenamento da imagem. Se na linha da imagem que está sendo varrida não existirem "pixels-on", o programa considera que foi encontrado um espaço em branco (linha de "pixels-off") existente entre duas linhas consecutivas de texto. Neste caso tem-se uma linha de texto segmentada e armazenada na matriz-imagem.

Segmentada a linha de texto, o programa inicia o processo de segmentação

dos caracteres desta linha. É interessante observar que o processo de leitura e armazenamento das linhas de texto é interrompido temporariamente, enquanto a linha de texto armazenada na matriz-imagem é processada.

No processo de segmentação dos caracteres a matriz-imagem é varrida da esquerda para a direita. Neste processo, o programa utiliza a variável **brkcost,** que armazena o número de "pixels-on" de cada coluna e a variável **ptrcol**, que indica qual é a coluna do caractere que está sendo analisada. Enquanto o programa varre a matriz-imagem da esquerda para direita, os "pixels-on" e os "pixels-off" varridos, são armazenados em uma matriz denominada de matriz-original. Este processo é repetido, até que a variável **brkcost** possua valor igual a 0, isto é, seja encontrada uma coluna constituída apenas por "pixels-off". Quando o valor armazenado na variável **brkcost** for igual a 0, o programa considera que foi encontrado um espaço em branco entre 2 caracteres adjacentes e que os "pixels-on" e os "pixels-off" armazenados na matriz-original constituem a imagem de um caractere que foi segmentado.

Após ser realizada a segmentação de um caractere, são realizados os processos de afilamento e reconhecimento desse caractere. O processo de afilamento é realizado utilizando-se o algoritmo de afilamento ARA, explicado no item 3.5 deste capítulo. O processo de reconhecimento do caractere afilado é realizado utilizando-se o algoritmo de reconhecimento desenvolvido neste trabalho e descrito nos capítulos 4 e 5.

Os processos de segmentação, afilamento e reconhecimento dos caracteres, são repetidos, até que não existam mais "pixels-on" na matriz-imagem.

A segmentação de caracteres descrita acima é eficiente apenas quando, entre os caracteres existem espaços em branco ("pixels-off"), ou seja, quando os caracteres são caracteres isolados. Neste ponto do trabalho ainda não foram analisados os casos em que os caracteres estão conectados ou sobrepostos. Caracteres sobrepostos, são caracteres que apresentam "pixels-on" na mesma

coluna, porém não se tocam. Caracteres conectados são caracteres distintos que possuem "pixels-on" em comum. A análise detalhada da segmentação dos caracteres sobrepostos e conectados será realizada no capítulo 5 deste trabalho.

Este trabalho tem como objetivo realizar o reconhecimento da quantia numérica de cheques bancários. Devido a isto, a região do cheque bancário que é digitalizada e gravada, é apenas a região que contém a área reservada à escrita da quantia numérica. Neste caso, considera-se que esta área, representa apenas uma linha de texto.

3.4 - DESCRIÇÃO DOS ALGORITMOS OPTA E PSTA

Afilamento é um importante passo do pré-processamento para muitas operações de análise de imagens [20]. Entre estas análises, pode-se citar: identificação de impressão digital, reconhecimento de caracteres ópticos e processamento de documentos, podendo também ser utilizado em sistemas biomédicos [21].

Afilamento é um processo no qual os "pixels-on" da fronteira da imagem são apagados, sem que ocorra a desconexão de "pixels-on" vizinhos. Este processo, é realizado até que todas as linhas ou curvas tenham largura de um "pixel-on". A imagem resultante do processo de afilamento é denominada de esqueleto da imagem original. O esqueleto resultante deve aproximar-se do eixo médio da imagem original.

Dentre os vários algoritmos de afilamento existentes pode-se citar: PSTA [22], OPTA[22], Suzuki's [23], KBTA [24], incluindo geração de contorno [21], etc.

O algoritmo ARA [18] possui dois passos principais, que são repetidos até que a imagem obtida aproxime-se do eixo médio da imagem original. No primeiro passo é marcado o contorno da imagem e no segundo passo o contorno marcado

é analisado visando verificar quais os "pixels-on" pertencentes a este contorno devem ser apagados. As principais características que o algoritmo ARA possui são: i) mantém a conectividade; ii) preserva os pontos finais; iii) o esqueleto resultante aproxima-se do eixo médio da imagem original; iv) é praticamente imune ao ruído; e de extrema importância v) o tempo de execução é muito rápido.

3.4.1 - Descrição do "One-Pass Thinning Algorithm" (OPTA)

No algoritmo OPTA [22], cada "pixel" da imagem é varrido seqüencialmente, começando do extremo superior da imagem no primeiro "pixel" esquerdo, até o último "pixel" no extremo inferior direito. Cada "pixel" é extraído da imagem com seus 10 "pixels" vizinhos, como mostrado na figura 3.2 e é comparado com 8 máscaras, como ilustrado na figura 3.3.

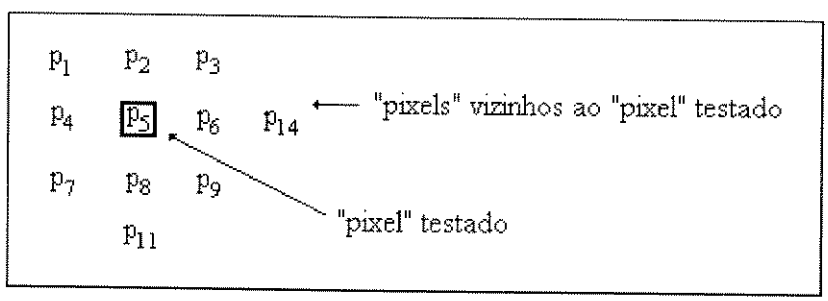


Figura 3.2 - "Pixel" testado e "pixels" vizinhos.

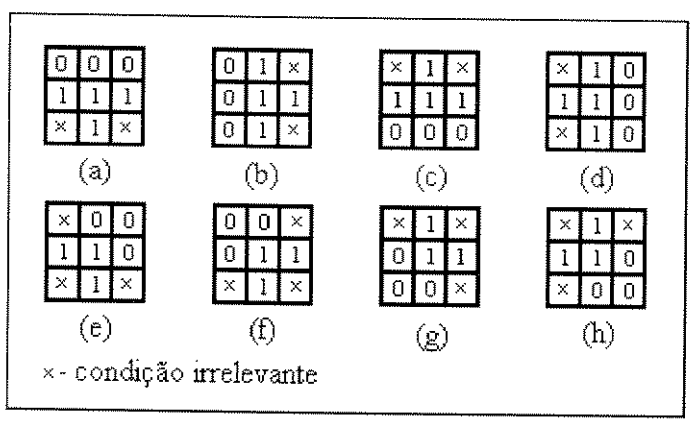


Figura 3.3 - Padrões de afilamento (máscaras).

Em particular, se os 8 "pixels" p_1, p_2, p_3, p_4, p_6, p_7, p_8 e p_9 são igualados com qualquer uma das 8 máscaras de afilamento a, b, c, d, e, f, g e h e o "pixel" $p_5 = 1$, o "pixel" p_5 é removido da imagem. Caso contrário, o "pixel" p_5 permanece na imagem.

De maneira a conservar a conectividade, os "pixels" vizinhos ao "pixel" testado (figura 3.2) são comparados com as máscaras i e j mostradas na figura 3.4.

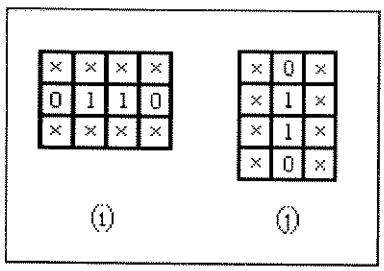


Figura 3.4 - Máscaras de restituição.

Em particular, se a segunda linha dos "pixels" vizinhos p_4, p_5, p_6 e p_{14} ao "pixel" testado é igualada com a segunda linha da máscara i da figura 3.4 ou a segunda coluna dos "pixels" vizinhos p_2, p_5, p_8 e p_{11} ao "pixel" testado é igualada com a segunda coluna da máscara j (figura 3.4), o "pixel" p_5 removido da imagem é restituído. Caso contrário, o "pixel" p_5 deve realmente ser removido da imagem.

O processo descrito acima, corresponde a uma iteração do algoritmo OPTA. Este processo é repetido, até que entre duas iterações consecutivas, não seja possível apagar mais "pixels-on". O número de iterações requerido para a obtenção do esqueleto da imagem, depende exclusivamente da forma e do tamanho da imagem original.

3.4.2 - Descrição do "Parallel-Symmetric Thinning Algorithm" (PSTA)

O algoritmo PSTA [22] utiliza as mesmas máscaras de remoção e

restituição do algoritmo OPTA, como mostrado na figura 3.5. Com apenas uma modificação, no algoritmo PSTA o tamanho das máscaras é 4 x 4. Isto implica que o valor dos "pixels" adicionais p_{10}, p_{12}, p_{13}, p_{15} e p_{16} das máscaras 4 x 4 é sempre "x", isto é, condição irrelevante. A principal diferença entre os algoritmos OPTA e PSTA, está no número de iterações requeridas para que o esqueleto final da imagem seja obtido.

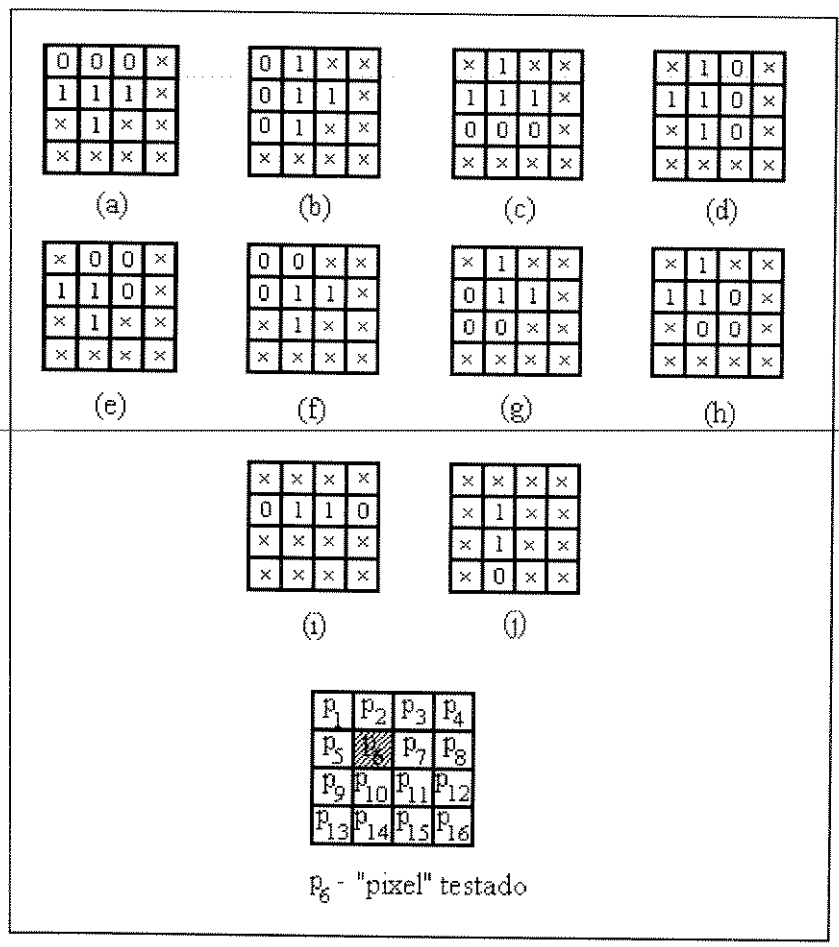


Figura 3.5 - Máscaras utilizadas no algoritmo PSTA.

Para obter o esqueleto final da imagem, o algoritmo PSTA realiza apenas 2 iterações; independente da forma e do tamanho da imagem.

Na primeira iteração, o algoritmo varre a imagem do seu extremo superior esquerdo até o seu extremo inferior direito, utilizando a janela 4 x 4 mostrada na figura 3.6. Os resultados obtidos $P = \{ p_i \}$ são armazenados temporariamente na

memória.

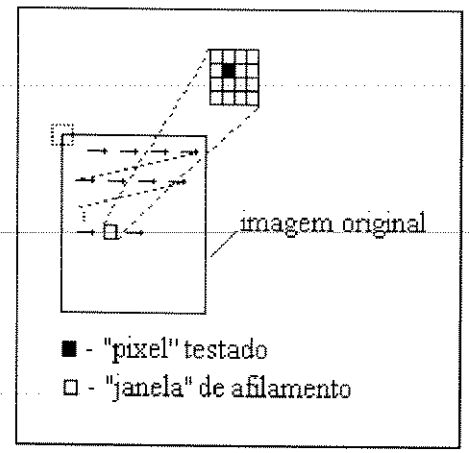


Figura 3.6 - Varredura na imagem da esquerda para a direita usando a "janela" de afilamento (máscara 4 x 4).

Na segunda iteração, os mesmos "pixels" são agora varridos da direita para a esquerda, usando-se a "janela" mostrada na figura 3.7. Os resultados temporários $Q = \{q_i\}$ são também armazenados na memória.

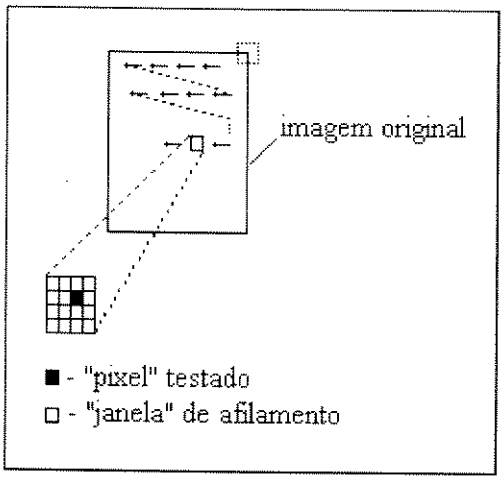


Figura 3.7 - Varredura na imagem da direita para a esquerda usando a "janela" de afilamento (máscara 4 x 4).

O esqueleto da imagem $S = \{s_i\}$ é obtido através da operação lógica AND. Esta operação lógica é realizada entre o resultado P da primeira iteração e o resultado Q da segunda iteração, como ilustrado na figura 3.8.

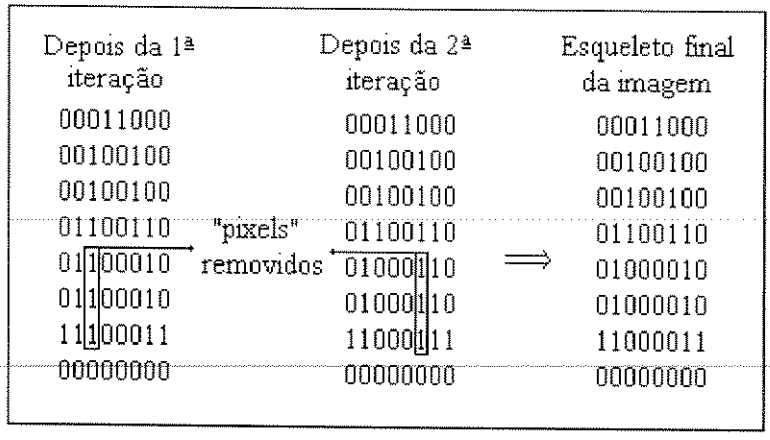


Figura 3.8 - Obtenção do esqueleto final da imagem utilizando o algoritmo PSTA.

Em particular, se $p_i = q_i = 1$, $s_i = [p_i \text{ (AND) } q_i] = 1$. Caso contrário, se $p_i = 0$ ou $q_i = 0$, $s_i = 0$. Onde p_i, q_i representam os "pixels" que são armazenados em P e Q, respectivamente, após a primeira e a segunda iteração. O símbolo s_i representa os "pixels" do esqueleto final, que são armazenados em S.

Após a obtenção do esqueleto final o passo descrito a seguir é realizado:

Em particular, se $E_j \in S$, com $n \geq 3$ e $m \geq 4$ e se as linhas da fronteira E_j (L_t para cima, L_b para baixo) são conectadas ao esqueleto principal S, ($L_t \cap S \neq \varnothing$ e/ou $L_b \cap S \neq \varnothing$), o esquema PSTA salva somente o eixo médio (de 1 ou 2 "pixels") da área E_j e a linha L_t e/ou L_b que é uma "linha-final" do esqueleto S. Caso contrário, se $E_j \cap S = \varnothing$, o algoritmo PSTA salva o eixo médio da área E_j e as linhas L_t e L_b. E_j é uma área sólida de n x m elementos.

O esquema PSTA requer somente 2 iterações e uma comparação em qualquer caso. O tempo de execução é linearmente proporcional ao tamanho da imagem, em contraste com muitos algoritmos de afilamento existentes que são exponencialmente proporcionais ao tamanho da imagem.

3.5 - DESCRIÇÃO DE UM ALGORITMO RÁPIDO DE AFILAMENTO (ARA)

O algoritmo ARA possui dois passos principais, que são repetidos até que a imagem obtida aproxime-se do eixo médio da imagem original. No primeiro passo é marcado o contorno da imagem e no segundo passo o contorno marcado é analisado visando verificar quais os "pixels-on" pertencentes a este contorno devem ser apagados. Após terminar este processo o esqueleto final da imagem estará armazenado na matriz-original. O programa considera como contorno os "pixels-on" da imagem que se encontram nas posições mais internas e externas dessa imagem. A matriz-original armazena os "pixels-on" e os "pixels-off" que constituem o esqueleto da imagem.

Uma observação importante a ser feita a respeito do algoritmo ARA é que este algoritmo pode ser aplicado tanto no processamento da imagem de um texto, quanto no processamento da imagem de uma linha deste texto, ou mesmo, no processamento de alguns dos caracteres que compõem este texto. Embora o algoritmo ARA possa ser utilizado para processar esses 3 tipos de imagens, neste capítulo foram mostrados apenas os resultados obtidos no processamento da imagem de caracteres isolados.

Os processos de delimitação e deleção do contorno da imagem são descritos a seguir.

Delimitação do contorno da imagem

A função que delimita o contorno da imagem no programa é denominada de delimita-contorno. Para facilitar o entendimento do processo utilizado pela função delimita-contorno, optou-se por explicá-lo no texto e esboçar o seu fluxograma. Aconselha-se ao leitor que para sua melhor clareza, leia o texto que

descreve a função delimita-contorno e acompanhe ao mesmo tempo o seu fluxograma. A figura 3.9 que se encontra nas páginas 328 a 331 do 2º volume deste trabalho, ilustra o fluxograma utilizado na função delimita-contorno. Nesta figura alguns comentários que se encontram no texto não constam do fluxograma. Devido à extensão do fluxograma ele foi dividido em 3 partes.

O programa delimita o contorno da imagem, varrendo a matriz-original da esquerda para a direita e de cima para baixo. Para cada "pixel-on" que está sendo varrido, o programa realiza os seguintes passos:

Se (((o "pixel-on" que está sendo varrido possui "pixel-on" vizinho que se encontra na direção 2, segundo o código de cadeia de 8 direções) e (não possui "pixel-on" vizinho que se encontra na direção 6)) ou ((possui "pixel-on" vizinho que se encontra na direção 6) e (não possui "pixel-on" vizinho que se encontra na direção 2))), o "pixel-on" que está sendo varrido é marcado com o valor 2, por ser considerado como pertencente ao contorno da imagem.

A figura 3.10 ilustra o código de cadeia de 8 direções.

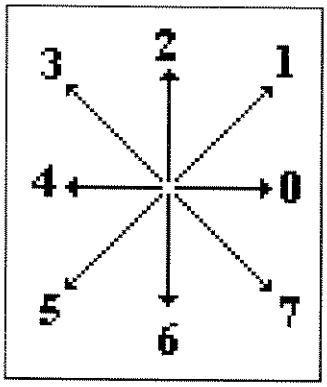


Figura 3.10 - Código de cadeia de 8 direções.

- Se (((o "pixel-on" que está sendo varrido possui "pixel-on" vizinho que se encontra na direção 4) e (não possui "pixel-on" vizinho que se encontra na direção 0)) ou ((possui "pixel-on" vizinho que se encontra na direção 0) e (não

possui "pixel-on" vizinho que se encontra na direção 4))), o "pixel-on" que está sendo varrido é marcado com o valor 2, por ser considerado como pertencente ao contorno da imagem.

Se o "pixel-on" que está sendo varrido não possui nenhum "pixel-on" vizinho que se encontra na direção 0, 2, 4 ou 6, o programa verifica se esse "pixel-on" possui "pixel-on" vizinho em alguma das 8 direções do código de cadeia. Caso o "pixel-on" que está sendo varrido não possua "pixel-on" vizinho, ele é apagado da imagem, por ser considerado pelo programa como um ruído. Ruído são "pixels-on" que podem aparecer ou não em posições aleatórias na imagem, devido, por exemplo, ao tipo de papel utilizado, ao processo de digitalização da imagem, etc. O programa considera como ruído o conjunto de "pixels-on" formado por até 4 "pixels-on".

- O processo de delimitação do contorno da imagem é realizado até que o extremo inferior direito da imagem seja encontrado.

Após finalizar o processo de delimitação do contorno da imagem, o programa executa a função que realiza a deleção do contorno da imagem que foi delimitado.

A figura 3.11 ilustra a imagem original de um caractere, o contorno da imagem que foi delimitado pela função delimita-contorno durante o processo de afilamento e o esqueleto final do caractere.

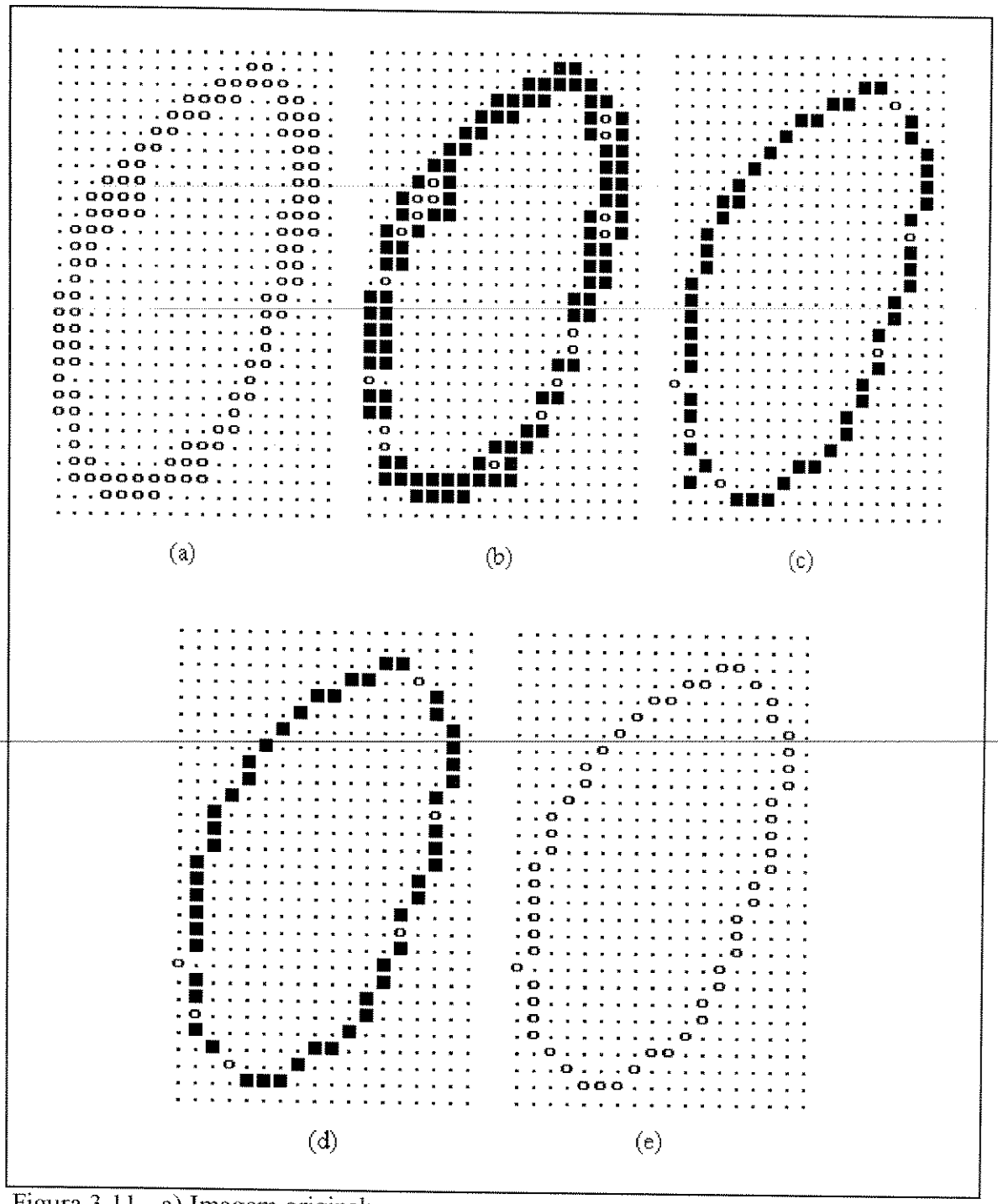


Figura 3.11 - a) Imagem original;

b) Delimitação inicial do contorno da imagem;

c) Delimitação do contorno da imagem com espessura de 1 "pixel-on" e ruído;

d) Delimitação do contorno da imagem no último passo do algoritmo ARA;

e) Esqueleto final da imagem.

Deleção do contorno da imagem

Ao terminar de delimitar o contorno da imagem, o programa varre a matriz-original novamente da esquerda para a direita e de cima para baixo, analisando para cada "pixel-on" marcado com o valor 2 se este "pixel-on" deve ou não ser apagado da imagem, a função que realiza esta análise é denominada pelo programa de apaga-contorno. Nesta análise, é verificado se o "pixel-on" marcado com valor 2 e seus "pixels-on" vizinhos encontram-se dispostos como ilustrado nas figuras 3.12 e 3.13. Esta análise é explicada detalhadamente a seguir.

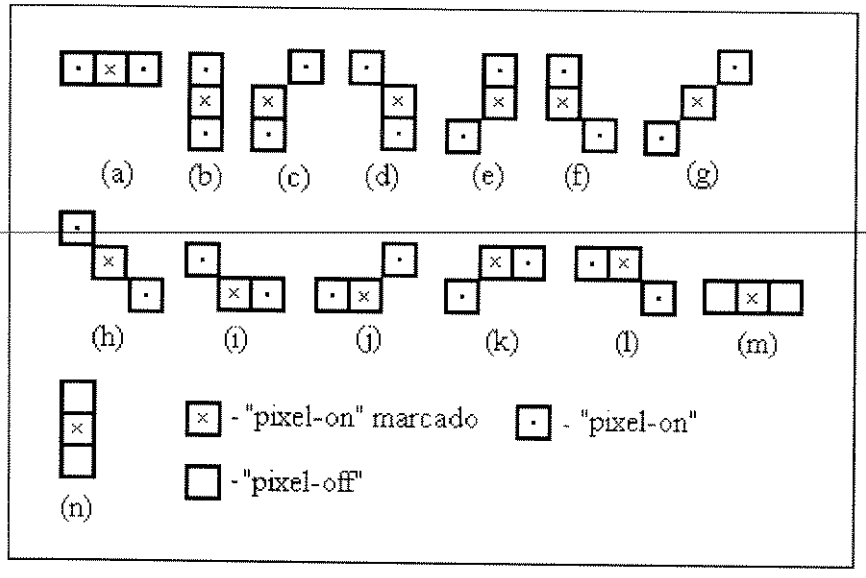


Figura 3.12 - Máscaras utilizadas na verificação da disposição do "pixel-on" marcado com o valor 2 e dos "pixels-on" vizinhos a este "pixel-on".

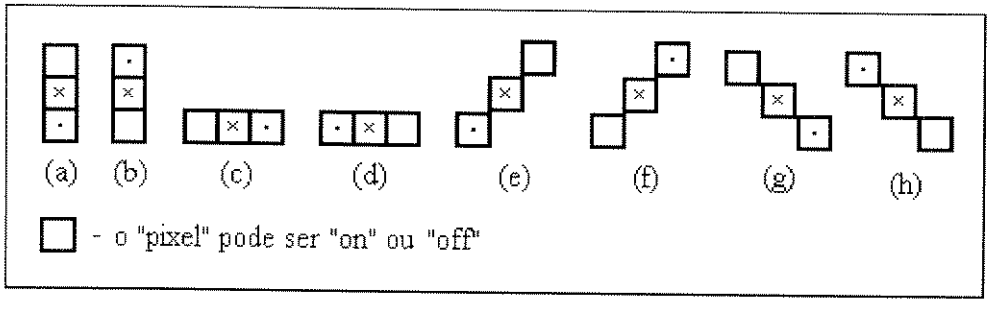


Figura 3.13 - Máscaras utilizadas na verificação da disposição do "pixel-on" marcado com o valor 2 e dos "pixels-on" vizinhos a este "pixel-on".

Para facilitar o entendimento do processo utilizado pela função apaga-contorno, optou-se por explicá-lo no texto e esboçar o seu fluxograma. Aconselha-se ao leitor que para sua melhor clareza, leia o texto que descreve a função apaga-contorno e acompanhe ao mesmo tempo o seu fluxograma. A figura 3.14 que se encontra nas páginas 332 a 338 do 2º volume deste trabalho, ilustra o fluxograma utilizado na função apaga-contorno. Nesta figura alguns comentários que se encontram no texto não constam do fluxograma. Devido à extensão do fluxograma ele foi dividido em 6 partes.

Se a disposição do "pixel-on" marcado com o valor 2 e seus "pixels-on" vizinhos coincide com alguma das máscaras de (a) a (l) da figura 3.12, a variável **sign0** é igualada a 1. Se ((a variável **sign0** foi igualada a 1) e (a disposição do "pixel-on" marcado com o valor 2 e seus "pixels-off" vizinhos coincide com a máscara (m) ou (n) da figura 3.12)), a variável **sign2** é igualada a 1. As variáveis **sign0** e **sign2** têm como objetivo indicar se o "pixel-on" marcado com o valor 2 e seus "pixels-on" vizinhos estão dispostos como indicado nas máscaras da figura 3.12 e/ou 3.13.

Se a disposição do "pixel-on" marcado com o valor 2 e seus "pixels-on" vizinhos não coincide com alguma das máscaras de (a) a (l) da figura 3.12, o programa verifica se a disposição do "pixel-on" marcado com o valor 2 e seus "pixels-on" vizinhos coincide com (((a máscara (a) ou (b) da figura 3.13) e (a máscara (m) da figura 3.12)) ou ((a máscara (c) ou (d) da figura 3.13) e (a máscara (n) da figura 3.12)) ou ((a máscara (e) ou (f) da figura 3.13) e (as máscaras (m) e (n) da figura 3.12)) ou ((a máscara (g) ou (h) da figura 3.13) e (as máscaras (m) e (n) da figura 3.12))). Em caso afirmativo, se o número de "pixels-on" vizinhos ao "pixel-on" marcado com o valor 2 é menor do que 2, o programa executa a função verifica-ruído. Esta função será explicada adiante neste capítulo. Caso contrário, se o número de "pixels-on" vizinhos ao "pixel-on" marcado com o valor 2 é igual a 2 ou maior do que 3, a variável **sign1** é igualada a 1 e se o

número de "pixels-on" vizinhos ao "pixel-on" marcado com o valor 2 é igual a 2, a coordenada do "pixel-on" marcado com o valor 2 é armazenada para que posteriormente seja verificado se este "pixel-on" deve permanecer ou deve ser apagado da imagem, por ser considerado pelo programa como sendo um ruído (este passo será referenciado adiante como PASSO 1). A variável **sign1** tem como objetivo indicar que o "pixel-on" marcado com o valor 2 possui 2 ou 4 "pixels-on" vizinhos. A função verifica-ruído tem como objetivo analisar se o "pixel-on" marcado com o valor 2 representa ou não um ruído na imagem.

Se os valores das variáveis ((**sign0** e **sign2**) ou **sign1**) são iguais a 1, o programa verifica se ((os "pixels-on" vizinhos ao "pixel-on" marcado com o valor 2 encontram-se na direção 1, 3, 5 ou 7) e (o "pixel-on" marcado com o valor 2 possui apenas 2 "pixels-on" vizinhos)). Em caso afirmativo, o programa executa a função verifica-desvio. Esta função será descrita adiante neste capítulo. A função verifica-desvio tem o objetivo de fazer com que ao longo do esqueleto da imagem os "pixels-on" estejam dispostos de forma seqüencial. Ao realizar isto, essa função faz com que a variação da direção entre "pixels-on" adjacentes seja a menor possível.

Se os valores das variáveis ((**sign0** ou **sign2**) e **sign1**) são diferentes de 1, o programa verifica se ((((o "pixel-on" marcado com o valor 2 possui "pixel-on" vizinho que se encontra na direção 1 ou 3) e (não possui "pixel-on" vizinho que se encontra na direção 2)) ou ((o "pixel-on" marcado com o valor 2 não possui "pixels-on" vizinhos que se encontram nas direções 1 e 3) e (possui "pixel-on" vizinho que se encontra na direção 2)) ou ((o "pixel-on" marcado com o valor 2 possui "pixel-on" vizinho que se encontra na direção 5 ou 7) e (não possui "pixel-on" vizinho que se encontra na direção 6)) ou ((o "pixel-on" marcado com o valor 2 não possui "pixels-on" vizinhos que se encontram nas direções 5 e 7) e (possui "pixel-on" vizinho que se encontra na direção 6))) e (o "pixel-on" marcado com o valor 2 possui menos do que 2 "pixels-on" vizinhos que se

encontram na direção 0 ou 4) e (o número de "pixels-on" vizinhos ao "pixel-on" marcado com o valor 2 é diferente de 1)). Em caso afirmativo, o programa executa a função verifica-conectividade. Esta função será explicada adiante neste capítulo. Caso contrário, o "pixel-on" marcado com o valor 2 é apagado da imagem, por ser considerado pelo programa como sendo um ruído. A função verifica-conectividade tem o objetivo de verificar se um "pixel-on" ao ser apagado da imagem não provocará a desconexão das partes que constituirão o esqueleto final dessa imagem.

Após terminar de varrer a matriz-original da esquerda para a direita e de cima para baixo, analisando se os "pixels-on" marcados com o valor 2 pertencem ou não ao contorno da imagem, o programa realiza a análise de todos os "pixels-on" cujas coordenadas foram armazenadas no PASSO 1. Nesta análise, se o "pixel-on' cuja coordenada foi armazenada no PASSO 1 possui 2 "pixels-on" vizinhos, o programa executa a função verifica-ruído. Caso contrário, se o "pixel-on" cuja coordenada foi armazenada no PASSO 1 possui mais do que 2 "pixels-on" vizinhos, o "pixel-on" cuja coordenada foi armazenada no PASSO 1 é apagado da imagem, por ser considerado pelo programa como sendo um ruído.

A seguir descreve-se detalhadamente as funções verifica-ruído, verifica-conectividade e verifica-desvio.

Verifica-ruído

A função verifica-ruído tem como objetivo analisar se o "pixel-on" marcado com o valor 2 representa ou não um ruído na imagem.

Para facilitar o entendimento da função verifica-ruído, optou-se por explicá-la no texto e esboçar o seu fluxograma. Aconselha-se ao leitor que para sua melhor clareza, leia o texto que descreve a função verifica-ruído e acompanhe ao mesmo tempo o seu fluxograma. A figura 3.15 que se encontra nas páginas

339 a 341 do 2º volume deste trabalho, ilustra o fluxograma da função verifica-ruído. Nesta figura alguns comentários que se encontram no texto não constam do fluxograma. Devido à extensão do fluxograma ele foi dividido em 2 partes.

Ao iniciar a função verifica-ruído, o programa verifica se o "pixel-on" marcado com o valor 2 possui 2 ou mais "pixels-on" vizinhos. Em caso afirmativo, o programa verifica se entre os "pixels-on" vizinhos ao "pixel-on" marcado com o valor 2 existem 2 "pixels-on" adjacentes. Se isto ocorrer, a variável **sign1** é igualada a 0 e o "pixel-on" marcado com o valor 2 é apagado da imagem, por ser considerado pelo programa como sendo um ruído. O "pixel-on" marcado com o valor 2, utilizado na função verifica-ruído é o "pixel-on" analisado na função apaga-contorno, que satisfez as condições necessárias para que a função verifica-ruído fosse executada. A figura 3.16 ilustra um caso em que o "pixel-on" marcado com o valor 2 possui 2 "pixels-on" vizinhos adjacentes que se encontram nas direções 2 e 3, segundo o código de cadeia 8 direções.

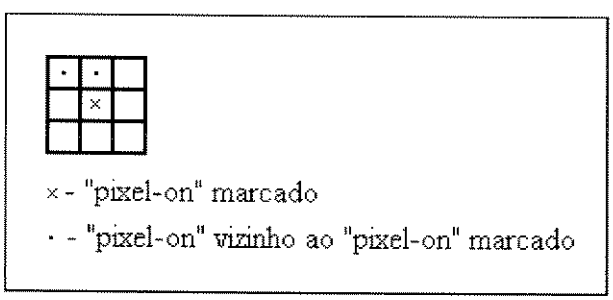


Figura 3.16 - "Pixel-on" marcado com o valor 2 e seus 2 "pixels-on" vizinhos adjacentes

Se o "pixel-on" marcado com o valor 2 possui um "pixel-on" vizinho, o programa denomina este "pixel-on" de *pn* e verifica se (((o "pixel-on" *pn* encontra-se na direção 0 ou 4 em relação ao "pixel-on" marcado com o valor 2 e possui "pixels-on" vizinhos que se encontram nas direções 2 e 6) ou (o "pixel-on" *pn* encontra-se em relação ao "pixel-on" marcado com o valor 2 em uma direção diferente das direções 0 e 4)) e (o "pixel-on" *pn* possui mais de 2 "pixels-on" vizinhos. Em caso afirmativo, o "pixel-on" marcado com o valor 2 é apagado

da imagem, por ser considerado pelo programa como sendo um ruído. Caso contrário, a variável **sign1** é igualada a 1.

Ao terminar a execução da função verifica-ruído, o programa retorna ao ponto da função apaga-contorno em que a função verifica-ruído foi executada.

Verifica-conectividade

A função verifica-conectividade tem o objetivo de verificar se um "pixel-on" ao ser apagado da imagem não provocará a desconexão das partes que constituirão o esqueleto final dessa imagem.

Para facilitar o entendimento da função verifica-conectividade, optou-se por explicá-la no texto e esboçar o seu fluxograma. Aconselha-se ao leitor que para sua melhor clareza, leia o texto que descreve a função verifica-conectividade e acompanhe ao mesmo tempo o seu fluxograma. A figura 3.17 que se encontra nas páginas 342 a 346 do 2º volume deste trabalho, ilustra o fluxograma da função verifica-conectividade. Nesta figura alguns comentários que se encontram no texto não constam do fluxograma. Devido à extensão do fluxograma ele foi dividido em 4 partes.

Ao iniciar a execução da função verifica-conectividade, o programa verifica em quais direções o "pixel-on" marcado com o valor 2 possui "pixels-on" vizinhos. O programa denomina de *pn* cada um dos "pixels-on" vizinhos ao "pixel-on" marcado com o valor 2. O "pixel-on" marcado com o valor 2 analisado na função verifica-conectividade, é o "pixel-on" marcado com o valor 2 analisado na função apaga-contorno que satisfez as condições para que a função verifica-conectividade fosse executada. Para cada "pixel-on" *pn* são realizados os procedimentos descritos a seguir :

- O programa verifica se ((o "pixel-on" *pn* encontra-se na direção 5 em relação

ao "pixel-on" marcado com o valor 2 e não possui "pixels-on" vizinhos que se encontram nas direções 0 e 2) ou (o "pixel-on" *pn* encontra-se na direção 7 em relação ao "pixel-on" marcado com o valor 2 e não possui "pixels-on" vizinhos que se encontram nas direções 2 e 4) ou (o "pixel-on" *pn* encontra-se na direção 6 em relação ao "pixel-on" marcado com o valor 2 e não possui "pixels-on" vizinhos que se encontram nas direções 1, 2 e 3) ou (o "pixel-on" *pn* possui apenas um "pixel-on" vizinho que se encontra na direção 2)). Em caso afirmativo, o "pixel-on" marcado com o valor 2 deve permanecer na imagem. Devido a isto este "pixel-on" é desmarcado.

- O programa verifica se ((o "pixel-on" *pn* encontra-se na direção 1 em relação ao "pixel-on" marcado com o valor 2 e não possui "pixels-on" vizinhos que se encontram nas direções 4 e 6) ou (o "pixel-on" *pn* encontra-se na direção 3 em relação ao "pixel-on" marcado com o valor 2 e não possui "pixels-on" vizinhos que se encontram nas direções 0 e 6) ou (o "pixel-on" *pn* encontra-se na direção 2 em relação ao "pixel-on" marcado com o valor 2 e não possui "pixels-on" vizinhos que se encontram nas direções 5, 6 e 7)). Em caso afirmativo, o "pixel-on" marcado com o valor 2 deve permanecer na imagem. Devido a isto este "pixel-on" é desmarcado.

- O programa verifica se ((((o "pixel-on" marcado com o valor 2 possui "pixel-on" vizinho que se encontra na direção 3, 4 ou 5) e (o "pixel-on" *pn* encontra-se na direção 0 em relação ao "pixel-on" marcado com o valor 2 e não possui "pixels-on" vizinhos que se encontram nas direções 3, 4 e 5)) ou ((o "pixel-on" marcado com o valor 2 possui "pixel-on" vizinho que se encontra na direção 0, 1 ou 7) e (o "pixel-on" *pn* encontra-se na direção 4 em relação ao "pixel-on" marcado com o valor 2 e não possui "pixels-on" vizinhos que se encontram nas direções 0, 1 e 7))) e (o "pixel-on" marcado com o valor 2 não possui "pixels-

on" vizinhos que se encontram nas direções 2 e 6)). Em caso afirmativo, o "pixel-on" marcado com o valor 2 deve permanecer na imagem. Devido a isto este "pixel-on" é desmarcado.

- Se nenhum dos itens anteriores for satisfeito, o "pixel-on" marcado com o valor 2 é apagado da imagem.

Ao verificar que todos os "pixels-on" *pn* vizinhos ao "pixel-on" marcado com o valor 2 foram analisados, o programa finaliza a execução da função verifica-conectividade. Caso contrário, o programa executa os procedimentos descritos acima para os "pixels-on" *pn* que não foram analisados.

Ao terminar a execução da função verifica-conectividade, o programa retorna ao ponto da função apaga-contorno em que a função verifica-conectividade foi executada.

Verifica-desvio

A função verifica-desvio tem como objetivo possibilitar que ao longo do esqueleto da imagem os "pixels-on" estejam dispostos de forma seqüencial, isto é, a variação da direção entre "pixels-on" adjacentes seja a menor possível.

Para facilitar o entendimento da função verifica-desvio, optou-se por explicá-la no texto e esboçar o seu fluxograma. Aconselha-se ao leitor que para sua melhor clareza, leia o texto que descreve a função verifica-desvio e acompanhe ao mesmo tempo o seu fluxograma. A figura 3.18 que se encontra nas páginas 347 a 351 do 2º volume deste trabalho, ilustra o fluxograma da função verifica-desvio. Nesta figura alguns comentários que se encontram no texto não constam do fluxograma. Devido à extensão do fluxograma ele foi dividido em 4 partes.

Ao iniciar a função verifica-desvio, se o "pixel-on" marcado com o valor 2 possui "pixels-on" vizinhos que se encontram nas direções 1 e 7 como está ilustrado na figura 3.19 (a), o valor da abscissa do "pixel-on" marcado com o valor 2 é incrementado de uma unidade. O programa denomina de *nva1*, o valor da abscissa do "pixel-on marcado com o valor 2 incrementado de uma unidade. Se o número de "pixels-on" vizinhos à posição *nva1* é menor do que 4, o "pixel-on" marcado com o valor 2 é movido para a posição *nva1* como está ilustrado na figura 3.19 (b). Se o "pixel-on" marcado com o valor 2 possui "pixels-on" vizinhos que se encontram nas direções 1 e 3 como está ilustrado na figura 3.19 (c), o valor da ordenada do "pixel-on" marcado com o valor 2 é decrementado de uma unidade. O programa denomina de *nvo1*, o valor da ordenada do "pixel-on marcado com o valor 2 decrementado de uma unidade. Se o número de "pixels-on" vizinhos à posição *nvo1* é menor do que 4, o "pixel-on" marcado com o valor 2 é movido para a posição *nvo1* como está ilustrado na figura 3.19 (d). Se o "pixel-on" marcado com o valor 2 não possui "pixel-on" vizinho que se encontra na direção 1 ou 3, o programa verifica se o "pixel-on" marcado com o valor 2 possui "pixels-on" vizinhos que se encontram nas direções 3 e 5 como está ilustrado na figura 3.19 (e), o valor da abscissa do "pixel-on" marcado com o valor 2 é decrementado de uma unidade. O programa denomina de *nva2*, o valor da abscissa do "pixel-on marcado com o valor 2 decrementado de uma unidade. Se o número de "pixels-on" vizinhos à posição *nva2* é menor do que 4, o "pixel-on" marcado com o valor 2 é movido para a posição *nva2* como está ilustrado na figura 3.19 (f). Se o "pixel-on" marcado com o valor 2 não possui "pixel-on" vizinho que se encontra na direção 3 ou 5, o programa verifica se o "pixel-on" marcado com o valor 2 possui "pixels-on" vizinhos que se encontram nas direções 5 e 7 como está ilustrado na figura 3.19 (g), o valor da ordenada do "pixel-on" marcado com o valor 2 é incrementado de uma unidade. O programa denomina de *nvo2*, o valor da ordenada do "pixel-on marcado com o valor 2

incrementado de uma unidade. Se o número de "pixels-on" vizinhos à posição *nvo2* é menor do que 4, o "pixel-on" marcado com o valor 2 é movido para a posição *nvo2* como está ilustrado na figura 3.19 (h). O "pixel-on" marcado com o valor 2 analisado na função verifica-desvio é o "pixel-on" analisado na função apaga-contorno que satisfez as condições necessárias para que a função verifica-desvio fosse executada. A figura 3.19 ilustra as análises realizadas pela função verifica-desvio.

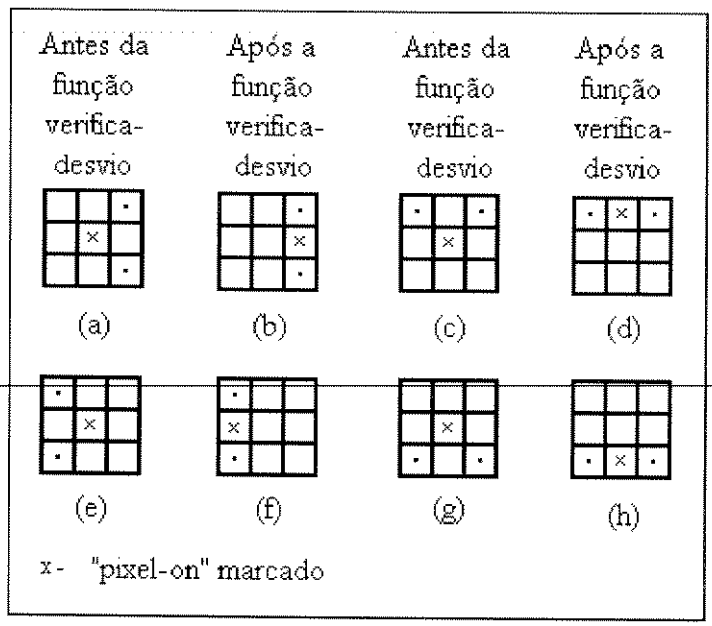


Figura 3.19 - Análises realizadas pela função verifica-desvio.

A figura 3.20 ilustra um caso em que o "pixel-on" marcado com o valor 2 é movido para a posição encontrada.

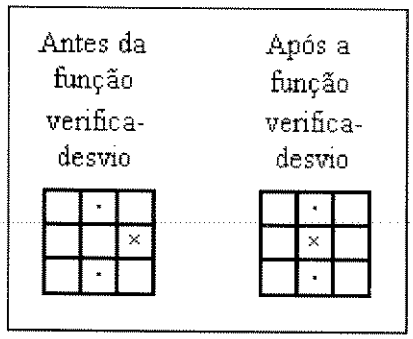


Figura 3.20 - Exemplo de um caso em que o "pixel-on" marcado com o valor 2 é movido para a posição encontrada.

Ao terminar a execução da função verifica-desvio, o programa retorna ao ponto da função apaga-contorno em que a função verifica-desvio foi executada.

3.6 - RESULTADOS OBTIDOS

O algoritmo ARA desenvolvido neste trabalho foi testado com 200 dígitos escritos por 20 pessoas. Foi solicitado a cada uma das pessoas que escrevesse em uma folha branca de papel sulfite e em apenas uma linha os números de 0 a 9, isolados uns dos outros para que não ocorresse o problema dos números ficarem sobrepostos ou conectados, pois os testes tinham como objetivo testar apenas dígitos isolados. A linha onde foram escritos os números de 0 a 9 é considerada pelo algoritmo de segmentação explicado neste capítulo no item 3.3, como a imagem de uma linha de texto. Os números de 0 a 9 escritos por cada pessoa foram digitalizados e gravados em um arquivo.

O algoritmo ARA também foi testado utilizando-se 260 caracteres alfabéticos escritos por 10 pessoas. Foi pedido a cada uma destas pessoas que escrevesse em uma folha de papel sulfite e em apenas uma linha os caracteres maiúsculos de A a M e em outra linha os caracteres maiúsculos de N a Z. As pessoas escreveram as letras em 2 linhas, pois em apenas uma linha não seria possível colocar de maneira satisfatória todas as 26 letras do alfabeto português.

Várias pessoas ao escreverem os números de 0 a 9 e os caracteres maiúsculos de A a Z utilizaram caneta azul, outras utilizaram caneta preta, ou mesmo caneta vermelha. Independente da cor da caneta utilizada todos os números e caracteres alfabéticos foram varridos por um "scanner" monocromático.

Com o objetivo de comparar o desempenho do algoritmo ARA com o desempenho de outros algoritmos, foram realizados testes com os algoritmos ARA, OPTA e PSTA. Os dois últimos algoritmos foram descritos no item 3.4 deste capítulo. As figuras 3.21, 3.22, 3.23, 3.24, 3.25, 3.26 mostram os resultados obtidos em alguns dos testes e a tabela 3.1 mostra o tempo de execução dos algoritmos de afilamento OPTA, PSTA e ARA. Os testes foram realizados utilizando-se um microcomputador Pentium - 100 MHz com 16 Mbytes de memória RAM.

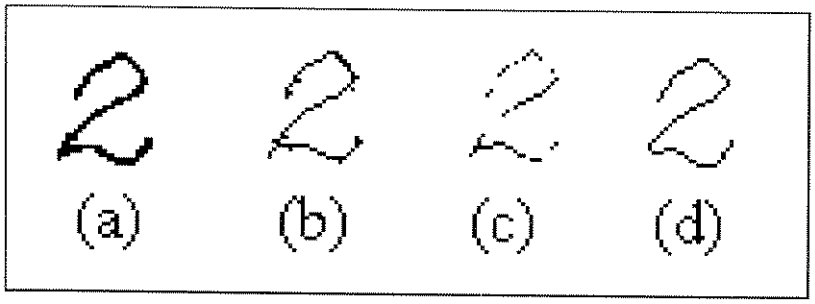


Figura 3.21 - a) Imagem original do número 2; b) Esqueleto obtido pelo algoritmo OPTA; c) Esqueleto obtido pelo algoritmo PSTA; d) Esqueleto obtido pelo algoritmo ARA.

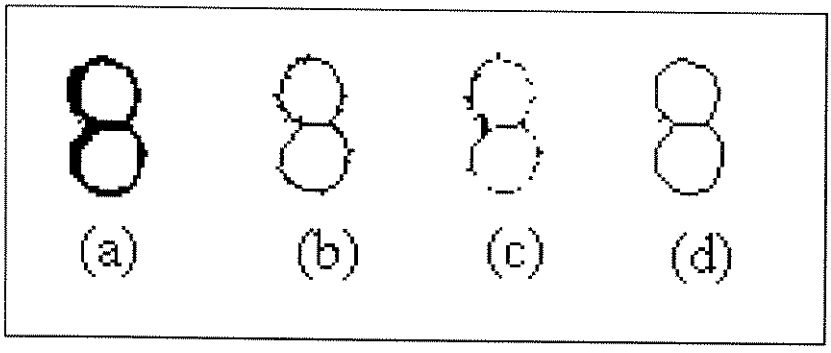


Figura 3.22 - a) Imagem original do número 8; b) Esqueleto obtido pelo algoritmo OPTA; c) Esqueleto obtido pelo algoritmo PSTA; d) Esqueleto obtido pelo algoritmo ARA.

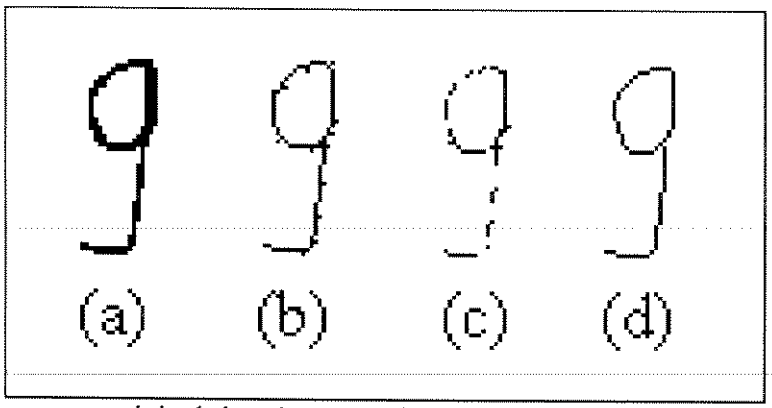


Figura 3.23 - a) Imagem original do número 9; b) Esqueleto obtido pelo algoritmo OPTA; c) Esqueleto obtido pelo algoritmo PSTA; d) Esqueleto obtido pelo algoritmo ARA.

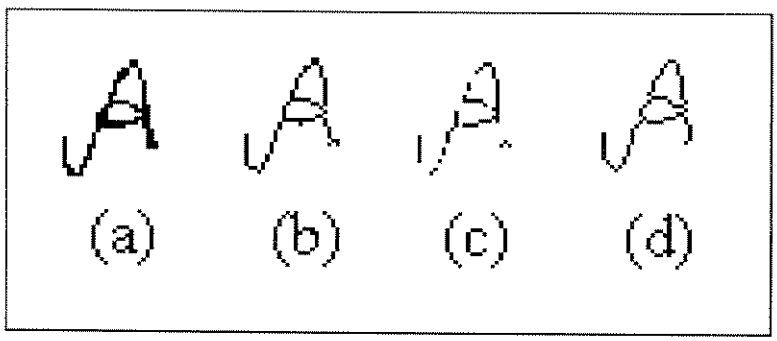


Figura 3.24 - a) Imagem original do caractere alfabético A; b) Esqueleto obtido pelo algoritmo OPTA; c) Esqueleto obtido pelo algoritmo PSTA; d) Esqueleto obtido pelo algoritmo ARA.

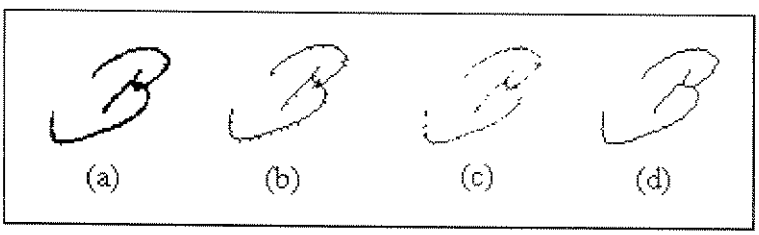


Figura 3.25 - a) Imagem original do caractere alfabético B; b) Esqueleto obtido pelo algoritmo OPTA; c) Esqueleto obtido pelo algoritmo PSTA; d) Esqueleto obtido pelo algoritmo ARA.

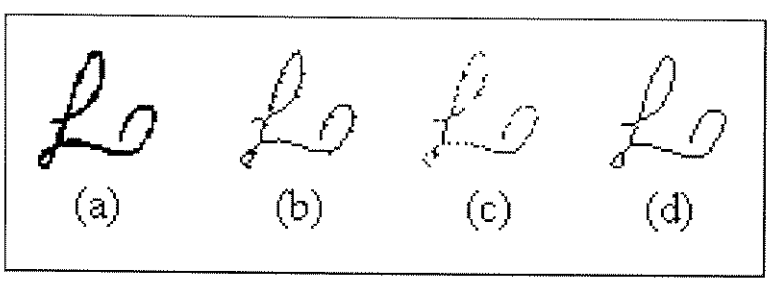


Figura 3.26 - a) Imagem original do caractere alfabético L; b) Esqueleto obtido pelo algoritmo OPTA; c) Esqueleto obtido pelo algoritmo PSTA; d) Esqueleto obtido pelo algoritmo ARA.

Tabela 3.1 - Tempo de execução dos algoritmos: OPTA, PSTA e ARA, utilizando-se um microcomputador Pentium - 100 Mhz com 16 Mbytes de memória RAM.

Caractere	OPTA (seg)	PSTA (seg)	ARA(seg)
2	0.056	0.041	0.013
8	0.100	0.070	0.019
9	0.061	0.056	0.017
A	0.046	0.040	0.012
B	0.179	0.131	0.044
L	0.097	0.094	0.030

3.7 - CONCLUSÕES

Os algoritmos de afilamento têm sido extensivamente estudados no contexto de reconhecimento de padrões e processamento de imagem.

O algoritmo rápido de afilamento (ARA) desenvolvido neste trabalho possui dois passos principais, que são repetidos até que a imagem obtida aproxime-se do eixo médio da imagem original. No primeiro passo é marcado o contorno da imagem e no segundo passo o contorno marcado é analisado visando verificar quais os "pixels-on" pertencentes a este contorno devem ser apagados.

Através da análise dos resultados obtidos, ilustrados nas figuras 3.21, 3.22, 3.23, 3.24, 3.25, 3.26 e na tabela 3.1 pode-se concluir que o algoritmo ARA ao realizar o afilamento de um caractere, mantém a conectividade e preserva os pontos finais da imagem, é praticamente imune ao ruído e, de extrema importância tem rápido tempo de execução. Nos testes realizados o algoritmo ARA é aproximadamente 3,0 vezes mais rápido do que o algoritmo PSTA, e 3,3 a 5,3 vezes mais rápido do que o algoritmo OPTA.

CAPÍTULO 4

UM ALGORITMO PARA RECONHECIMENTO DE CARACTERES NUMÉRICOS MANUSCRITOS ISOLADOS QUE CONSIDERA AS CARACTERÍSTICAS INTRÍNSECAS E COMUNS DE CADA DÍGITO

4.1 - INTRODUÇÃO

Neste capítulo, é descrito o algoritmo desenvolvido neste trabalho para reconhecer caracteres numéricos manuscritos isolados, que podem ser contínuos ou fragmentados [25]. Caracteres fragmentados são caracteres em que uma ou mais partes estão desconectadas e caracteres contínuos são caracteres em que todas as suas partes estão conectadas. Este algoritmo tem como base a verificação das características intrínsecas de cada dígito e a verificação das

características que os dígitos têm em comum. Ao verificar estas caracterísicas, o algoritmo desenvolvido procura aproximar-se do processo de reconhecimento utilizado pelo ser humano.

O algoritmo de reconhecimento de caracteres numéricos manuscritos, agrupa os "pixels-on" que constituem a imagem de cada dígito em linhas horizontais e linhas verticais. São as coordenadas iniciais e finais dessas linhas que definem as características intrínsecas e comuns de cada um dos dígitos.

O programa considera que cada caractere possui uma linha horizontal superior, uma linha horizontal inferior, uma linha vertical esquerda, uma linha vertical direita e pode possuir um ou mais pontos de cruzamento. A linha horizontal superior é constituída pelos "pixels-on" que formam o extremo superior do caractere, enquanto a linha horizontal inferior é constituída pelos "pixels-on" que formam o extremo inferior do caractere. Os "pixels-on" que formam o lado esquerdo do caractere constituem a linha vertical esquerda, enquanto os "pixels-on" que formam o lado direito constituem a linha vertical direita. Um ponto de cruzamento representa uma região do caractere em que um "pixel-on" possui 2 ou mais "pixels-on" vizinhos e pode estar situado em posições aleatórias do caractere.

No programa, foram implementadas as funções: linha horizontal superior, linha horizontal inferior, linha vertical esquerda, ponto de cruzamento, vertical e horizontal. As coordenadas iniciais e finais das linhas horizontal superior, horizontal inferior e vertical esquerda, são determinadas pelas três primeiras funções. Essas coordenadas são a base do processo de reconhecimento utilizado no algoritmo desenvolvido e são essas coordenadas que definem as características intrínsecas e comuns de cada um dos dígitos. As duas últimas funções, vertical e horizontal foram implementadas para possibilitar ao programa varrer a matriz-original. As funções linha horizontal superior e linha horizontal

inferior constituem no programa uma única função, que é denominada de horizontal. Esta função determina as coordenadas inicial e final da linha horizontal superior, se o programa estiver delimitando esta linha e, determina as coordenadas inicial e final da linha horizontal inferior, caso seja esta a linha a ser delimitada. Com o objetivo de proporcionar ao leitor uma maior clareza do texto, a função horizontal foi dividida em duas partes: função linha horizontal superior e função linha horizontal inferior.

No algoritmo de segmentação de texto, explicado no capítulo 3, a primeira segmentação a ser aplicada sobre o texto (imagem lida do arquivo no formato tiff), é a segmentação de suas linhas. O conjunto de "pixels-on" e "pixels-off" que constituem a imagem de cada linha de texto segmentada, são armazenados em uma matriz denominada de matriz-imagem. Após ser segmentada uma linha de texto, inicia-se a segmentação de seus caracteres. Se essa linha possuir somente caracteres isolados os "pixels-on" e "pixels-off" que constituem a imagem de cada caractere segmentado da matriz-imagem, são armazenados em uma matriz denominada de matriz-original. Após o programa segmentar o caractere, é utilizado o algoritmo de afilamento ARA (Algoritmo rápido de afilamento) explicado no capítulo 3, que obtém o esqueleto desse caractere. Ao final da execução deste algoritmo o esqueleto final do caractere estará armazenado na matriz-original e esta matriz é utilizada no reconhecimento do caractere numérico.

É importante lembrar que o algoritmo de reconhecimento desenvolvido é capaz de realizar o reconhecimento do caractere independente do seu tamanho, de sua posição e de sua inclinação.

Neste capítulo, é descrito o algoritmo para o reconhecimento de caracteres numéricos manuscritos isolados passo a passo, comenta-se a respeito dos dados utilizados para testar este algoritmo e, finalmente, são feitas conclusões a respeito

destes resultados. Nesta fase do trabalho, ainda não foi implementado no algoritmo de reconhecimento procedimentos que possibilitem o reconhecimento de dígitos sobrepostos ou conectados.

4.2 - DESCRIÇÃO DO ALGORITMO PARA RECONHECIMENTO DE CARACTERES NUMÉRICOS MANUSCRITOS ISOLADOS QUE CONSIDERA AS CARACTERÍSTICAS INTRÍNSECAS E COMUNS DE CADA DÍGITO

As principais funções que definem as características intrínsecas e comuns de cada dígito são:

Linha Horizontal Superior

O programa considera que a linha horizontal superior é constituída pelos "pixels-on" que formam o extremo superior do caractere e denomina a função que delimita as coordenadas inicial e final dessa linha de horizontal.

O programa delimita as coordenadas inicial e final da linha horizontal superior de cada caractere, varrendo a matriz-original da esquerda para a direita e da direita para a esquerda. Após serem encontradas as coordenadas inicial e final da linha horizontal superior, ao varrer a matriz-original da esquerda para a direita e da direita para a esquerda, o programa considera como coordenadas inicial e final da linha horizontal superior, as coordenadas que proporcionarão maior comprimento a esta linha. A matriz-original armazena os "pixels-on" e os "pixels-off" que constituem o esqueleto da imagem do caractere e, como neste capítulo só foram considerados dígitos isolados, essa matriz contém os "pixels-on" e os "pixels-off" que constituem o esqueleto da imagem de apenas um dígito.

Inicialmente, é descrito o processo de delimitação da linha horizontal superior, quando a matriz-original é varrida da esquerda para a direita e, posteriormente, é descrito o processo de delimitação da linha horizontal superior, quando a matriz-original é varrida da direita para a esquerda.

Para facilitar o entendimento do processo de delimitação das coordenadas inicial e final da linha horizontal superior, quando a matriz-original é varrida da esquerda para direita, optou-se por explicá-lo no texto e esboçar o seu fluxograma. Aconselha-se ao leitor que para sua melhor clareza, leia o texto que descreve a delimitação da linha horizontal superior, quando a matriz-original é varrida da esquerda para a direita e acompanhe ao mesmo tempo o seu fluxograma. A figura 4.1 que se encontra nas páginas 353 a 363 do 2º volume deste trabalho, ilustra o fluxograma que realiza a delimitação das coordenadas inicial e final da linha horizontal superior, quando a matriz-original é varrida da esquerda para a direita. Nesta figura, alguns comentários que se encontram no texto não constam do fluxograma. Devido à extensão do fluxograma ele foi dividido em 10 partes.

O programa inicia a delimitação das coordenadas inicial e final da linha horizontal superior de cada caractere, varrendo a matriz-original da esquerda para a direita e de cima para baixo, até encontrar o primeiro "pixel-on". Este "pixel-on" define o início da linha horizontal superior. O programa armazena a sua coordenada denominando-a de CILHS (coordenada inicial da linha horizontal superior). A coordenada CILHS (ACILHS, OCILHS) consiste de uma abscissa denominada de abscissa da coordenada inicial da linha horizontal superior (ACILHS) e de uma ordenada denominada de ordenada da coordenada inicial da linha horizontal superior (OCILHS).

Ao encontrar o esqueleto de cada caractere, é verificado que os "pixels-on" que constituem as linhas horizontal superior e horizontal inferior do

BIBLIOTECA CENTRAL

caractere, não se encontram todos alinhados e que, em alguns casos, podem até mesmo ocorrer regiões do caractere que apresentam ausência de "pixels-on". Para minimizar este problema e tornar o algoritmo mais flexível, foram definidas no programa as variáveis **"condição-de-folga"** e **"condição-vertical"**. O objetivo da variável **"condição-de-folga"** é indicar que foi encontrada uma região do caractere que apresenta ausência de "pixels-on" na direção em que a matriz-original está sendo varrida e que o programa tentará encontrar, em uma distância de até 2 "pixels-on", algum "pixel-on" a partir do qual essa matriz possa continuar a ser varrida na mesma direção. O objetivo da variável **"condição-vertical"** é indicar se o "pixel-on" que está sendo varrido encontra-se na direção vertical (direção 2 ou 6). Se o "pixel-on" que está sendo varrido encontra-se na direção vertical, o valor da variável **"condição-vertical"** é diferente de 0, sendo que este valor é proporcional à quantidade de "pixels-on" varridos que se encontram na direção vertical. Se o "pixel-on" que está sendo varrido encontra-se na direção horizontal (direção 0 ou 4), o valor da variável **"condição-vertical"** é igual a 0.

A partir da coordenada CILHS, o programa varre a matriz-original para a direita, procurando um novo "pixel-on", que deve estar localizado uma coluna a direita do "pixel-on" de abscissa ACILHS, na mesma linha da ordenada OCILHS (direção 0), uma linha acima (direção 1) ou abaixo (direção 7) (este passo será referenciado adiante como PASSO 1).

Se o PASSO 1 foi satisfeito, as variáveis **"condição-de-folga"** e **"condição-vertical"** são igualadas a 0.

Se ((o PASSO 1 não foi satisfeito e o valor da variável **"condição-de-folga"** é igual a 0) ou (o último "pixel-on" varrido encontra-se na direção vertical (direção 2 ou 6, conforme o código de cadeia de 8 direções [16], ilustrado na figura 3.10 no capítulo 3 deste trabalho) e (o "pixel-on" que está sendo varrido

não se encontra na direção 0 ou 4 (direção horizontal))) (este passo será referenciado adiante como PASSO 2), se o PASSO 1 não foi satisfeito, é procurado um "pixel-on" na matriz-original, que esteja situado na direção 2 ou 6 em relação ao "pixel-on" que está sendo varrido e o programa verifica se ((o valor da variável **"condição-vertical"** é diferente de 10) e (a direção do último "pixel-on" varrido é diferente de 0 ou o "pixel-on" que está sendo varrido não se encontra na direção 2 ou 6)) (esta condição será referenciada adiante como CONDIÇÃO 1). Quando o valor da variável **"condição-vertical"** é igual a 10, indica que o programa está apenas varrendo a matriz-original contando o número de "pixels-on" varridos, sem se preocupar em determinar se estes "pixels-on" encontram-se na direção horizontal ou na direção vertical. Quando o valor da variável **"condição-vertical"** é diferente de 10, indica que o programa está varrendo a matriz-original para a delimitação da linha horizontal superior e, devido a isto, é necessário determinar se os "pixels-on" varridos estão na direção horizontal ou na direção vertical. Se a CONDIÇÃO 1 foi satisfeita, o programa executa os seguintes procedimentos:

- O programa verifica se ((o último "pixel-on" varrido encontra-se na direção 1 e o "pixel-on" que está sendo varrido encontra-se na direção 7) ou (o último "pixel-on" varrido encontra-se na direção 7 e o "pixel-on" que está sendo varrido encontra-se na direção 1) ou (o último "pixel-on" varrido encontra-se na direção 5 e o "pixel-on" que está sendo varrido encontra-se na direção 3) ou (o último "pixel-on" varrido encontra-se na direção 3 e o "pixel-on" que está sendo varrido encontra-se na direção 5)). Se uma destas condições foi satisfeita, os valores das variáveis **"condição-vertical"** e **tot-vert** são decrementados de uma unidade. Se nenhuma dessas condições foi satisfeita, o programa verifica se o "pixel-on" que está sendo varrido não é o primeiro

"pixel-on" a ser varrido ou se esse "pixel-on" encontra-se na direção 2 ou 6. Em caso afirmativo, o programa verifica se o "pixel-on" que está sendo varrido possui "pixel-on" vizinho que se encontra na direção horizontal (direção 0), na direção vertical (direção 2 ou 6) ou na direção diagonal (direção 1 ou 7). Neste caso, os valores das variáveis **"condição-vertical"** e **tot-vert** são incrementados de uma unidade. Se o "pixel-on" que está sendo varrido não possui "pixel-on" vizinho que se encontra na direção horizontal, na direção vertical ou na direção diagonal, a variável **"condição-vertical"** é igualada ao valor 4. Quando o valor da variável **"condição-vertical"** é igual a 4, indica que não foram encontrados "pixels-on" na direção 2 ou 6 (direção vertical). A variável **tot-vert** armazena a quantidade de "pixels-on" varridos que se encontram na direção vertical.

- Se o valor da variável **"condição-vertical"** é maior ou igual a 3 e diferente de 4, o valor contido na variável **tot-vert** é decrementado do valor contido na variável **"condição-vertical"**. Independente do valor da variável **"condição-vertical"** ser igual ou diferente de 4, a coordenada do "pixel-on" que está sendo varrido é armazenada. O programa retorna ao "pixel-on" que tornou o valor da variável **"condição-vertical"**diferente de 0 e, após isto, esta variável é igualada a 0.

Se o PASSO 2 não foi satisfeito e o valor da variável **"condição-vertical"** for diferente de 10, esta variável é igualada a 0.

Se não foram encontrados "pixels-on" que pudessem ser varridos na matriz-original para a direita (esta condição será referenciada adiante como CONDIÇÃO 2), isto é, o "pixel-on" que está sendo varrido não possui "pixel-on" vizinho que se encontra na direção horizontal (direção 0), na direção vertical

(direção 2 ou 6) ou na direção diagonal (direção 1 ou 7), o valor da variável **"condição-de-folga"** é diferente de 0 ou o valor assumido pela variável **"condição-vertical"** é maior ou igual a 3 no PASSO 2, o programa realiza os seguintes procedimentos:

- Se o valor da variável **"condição-de-folga"** é igual a 0 e o valor da variável **npix-lr** é igual a 0 ou maior do que 2, é armazenada a coordenada do "pixel-on" que está sendo varrido (este passo será referenciado adiante como PASSO 3). A variável **npix-lr** armazena a quantidade de "pixels-on" varridos, quando a matriz-original é varrida para a direita, após o valor da variável **"condição-de-folga"** ser diferente de 0 pela primeira vez.

- O programa analisa se o "pixel-on" que está sendo varrido é um ponto ou um ruído. Esta análise será descrita no item 5.2 do capítulo 5. Se nessa análise o programa verificar que o "pixel-on" que está sendo varrido é considerado como sendo ponto ou ruído, o programa reinicia o processo de delimitação das coordenadas inicial e final da linha horizontal superior, quando a matriz-original é varrida da esquerda para a direita.

- Se ao adicionar uma unidade ao valor da abscissa do "pixel-on" que está sendo varrido, este novo valor da abscissa não exceder os limites da matriz-original e o valor da variável **"condição-de-folga"** é diferente de 2, o valor da variável **"condição-de-folga"** é incrementado de uma unidade e o "pixel-off" ("pixel" com valor igual a 0) da coordenada encontrada, ao se incrementar o valor da abscissa do "pixel-on" que está sendo varrido, é varrido utilizando-se os mesmos critérios que foram utilizados na procura de um "pixel-on" vizinho ao

"pixel-on" que estava sendo varrido, ou seja, o programa realiza o PASSO 1 partindo desse "pixel-off". O objetivo de varrer a matriz-original a partir do "pixel-off" dessa coordenada, é tentar encontrar novos "pixels-on" que possam ser varridos.

- Se ao adicionar uma unidade ao valor da abscissa do "pixel-on" que está sendo varrido, este novo valor da abscissa exceder os limites da matriz-original ou o valor da variável **"condição-de-folga"** for igual a 2, o programa retorna ao "pixel-on" cuja coordenada foi armazenada no PASSO 3 e o processo de delimitação das coordenadas inicial e final da linha horizontal superior, quando a matriz-original é varrida da esquerda para a direita, é finalizado.

Se forem encontrados "pixels-on" que possam ser varridos na matriz-original, isto é, a CONDIÇÃO 2 não foi satisfeita, o programa segue os seguintes procedimentos:

- Se o valor da variável **"condição-de-folga"** ou da variável **npix-lr** é diferente de 0, o valor da variável **npix-lr** é incrementado de uma unidade. Caso contrário, a variável **npix-lr** é igualada a 0.

- Se o "pixel-on" que está sendo varrido (a variável que armazena a direção deste "pixel-on" é denominada no programa de **dir**) encontra-se na direção diagonal e o próximo "pixel-on" que será varrido (a variável que armazena essa direção é denominada no programa de **ndir**) encontra-se na direção diagonal e ((o valor da variável **dir** é igual a 7 e o valor da variável **ndir** é igual a 1) ou (o valor da variável **dir** é igual a 1 e o valor da variável **ndir** é

igual a 7)), o valor da variável **diag** é incrementado de uma unidade. Caso contrário, a variável **diag** é igualada a -1, indicando que foi encontrado um "pixel-on" na direção horizontal (direção 0 ou 4) ou na direção vertical (direção 2 ou 6). A variável **diag** armazena a quantidade de "pixels-on" que se encontram na direção diagonal.

- Se ((o último "pixel-on" varrido possui 2 "pixels-on" vizinhos e sua coordenada não é a coordenada CILHS e ((o valor da variável **dir** é igual a 0 e o último "pixel-on" varrido possui "pixels-on" vizinhos que se encontram nas direções 3 e 5) ou (o último "pixel-on" varrido possui "pixel-on" vizinho que se encontra na direção 4 e o valor da variável **dir** é igual a 1 ou 7)) ou (o último "pixel-on" varrido possui mais do que 2 "pixels-on" vizinhos)), o valor da variável **ncruz** é incrementado de uma unidade (este passo será referenciado adiante como PASSO 4). A variável **ncruz** armazena o número de pontos de cruzamento encontrados.

- Se o valor da variável **"condição-vertical"** é igual a 0 e esta variável no PASSO 2 não apresentou valor maior ou igual a 3, o valor da variável **npix-lr** é incrementado de uma unidade.

O processo de delimitação da linha horizontal superior, quando a matriz-original é varrida da esquerda para a direita se repete, enquanto o valor da variável **"condição-de-folga"** é menor do que 2.

Ao terminar de delimitar a linha horizontal superior, quando a matriz-original é varrida da esquerda para a direita (valor da variável **"condição-de-folga"** maior ou igual a 2), se o valor da variável **"condição-vertical"** é igual a

10, a coordenada armazenada quando esta variável foi igualada a 10 é considerada como sendo a coordenada do último "pixel-on" varrido para a direita na linha horizontal superior, ou seja, a coordenada CFLHS (coordenada final da linha horizontal superior). A coordenada CFLHS (ACFLHS, OCFLHS) consiste de uma abscissa denominada de abscissa da coordenada final da linha horizontal superior (ACFLHS) e de uma ordenada denominada de ordenada da coordenada final da linha horizontal superior (OCFLHS). Se o valor da variável **"condição-vertical"** é diferente de 10, a coordenada CFLHS será a coordenada do último "pixel-on" varrido no processo de delimitação da linha horizontal superior, quando a matriz-original é varrida da esquerda para a direita.

A figura 4.2 ilustra a linha horizontal superior delimitada da esquerda para a direita, em 2 números manuscritos.

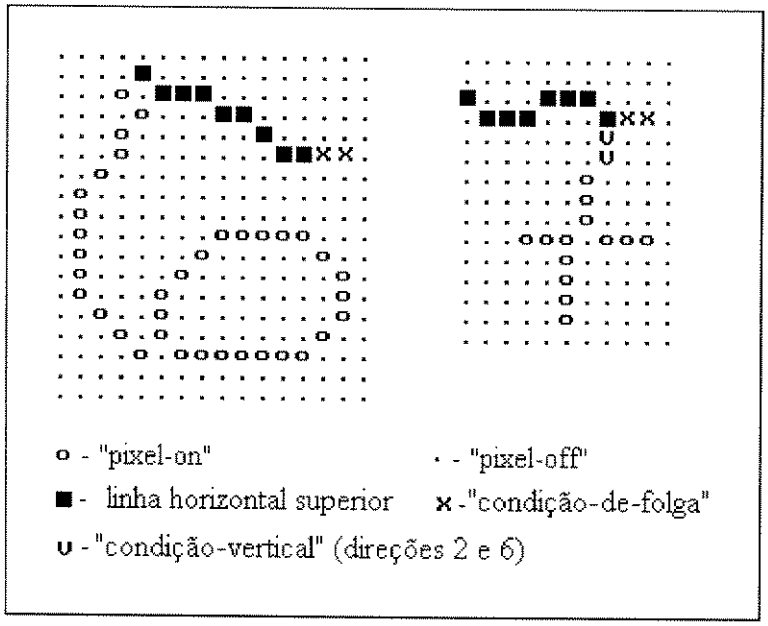


Figura 4.2 - Linha horizontal superior delimitada da esquerda para a direita em 2 números manuscritos.

Realizada a delimitação das coordenadas inicial e final da linha horizontal

superior, quando a matriz-original é varrida da esquerda para a direita, o programa passa, então, a realizar a delimitação das coordenadas inicial e final da linha horizontal superior, quando a matriz-original é varrida da direita para a esquerda. Aqui também é mostrado o fluxograma e, para maior clareza, o leitor deve seguir as mesmas instruções fornecidas no início da função linha horizontal superior. A figura 4.3 que se encontra nas páginas 364 a 376 do 2º volume deste trabalho, ilustra o fluxograma que realiza a delimitação das coordenadas inicial e final da linha horizontal superior, quando a matriz-original é varrida da direita para a esquerda e apenas alguns comentários que estão no texto, não constam do fluxograma. Devido a extensão do fluxograma, o mesmo foi dividido em 12 partes.

A coordenada utilizada para iniciar a delimitação da linha horizontal superior, quando a matriz-original é varrida da direita para a esquerda é a coordenada CFLHS, encontrada quando esta linha foi delimitada da esquerda para a direita. O processo de delimitação das coordenadas inicial e final da linha horizontal superior, quando a matriz-original é varrida da direita para a esquerda, é realizado utilizando-se a função denominada horizontal (explicada adiante neste capítulo). Para cada "pixel-on" varrido pela função horizontal, são realizados os seguintes testes:

- Se ((o valor da variável **"condição-vertical"** antes do "pixel-on" ser varrido era diferente de 0 e após esse "pixel-on" ser varrido o valor dessa variável é igual a 0) e ((o valor da variável **dir** é igual a (2 ou 6)) e (o valor da variável **ndir** é diferente de 4))), o "pixel-on" que está sendo varrido é considerado como um "pixel-on" que se encontra na direção vertical e o valor da variável **"condição-vertical"** é incrementado de uma unidade.

Enquanto a matriz-original é varrida para a esquerda, se é encontrado um

ponto de cruzamento (explicado adiante neste capítulo), a variável **ncruz** é igualada a 1 e se o valor da variável **vert** é maior do que 3 e a quantidade de "pixels-on" varridos que se encontram na direção vertical é maior do que 90% do número de "pixels-on" varridos para a esquerda na matriz-original, a variável **tot-vert** é igualada ao valor 100, indicando que foram encontrados mais "pixels-on" na direção vertical do que na direção horizontal, ao varrer a matriz-original para a esquerda.

- Se ((o valor da variável **"condição-vertical"** é diferente de 0 ou o valor da variável **ndir** é igual a 3 e o valor da variável **dir** é igual a 2) e (o valor da variável **"condição-de-folga"** é diferente de 0)), o "pixel-on" que está sendo varrido encontra-se na direção vertical. Devido a isto, o valor da variável **vert** é incrementado de uma unidade.

- Se o valor da variável **"condição-de-folga"** é igual a 0 e o valor desta variável não foi diferente de 0, anteriormente, o valor da variável **npix-rl** é incrementado de uma unidade e o programa continua a varrer a matriz-original para a esquerda. Caso contrário, se o valor da variável **"condição-de-folga"** é diferente de 0 e o "pixel" que está sendo varrido é um "pixel-off", é verificado se este "pixel-off" possui "pixel-on" vizinho que se encontra na direção 3, 4 ou 5. Em caso afirmativo, o programa verifica se antes da matriz-original ser varrida para a esquerda, esse "pixel-off" não possuía "pixels-on" vizinhos que se encontravam nas direções (((1 ou 3) e (5 ou 7)) e (2 e 6)) (esta condição será referenciada adiante como CONDIÇÃO 3). Se a CONDIÇÃO 3 é satisfeita, o "pixel-off" que representa uma falha (o significado de falha será explicado no item 5.2 do capítulo 5 deste trabalho), é substituído por um

"pixel-on". A variável **npix-rl** indica a quantidade de "pixels-on" que são varridos, independente destes "pixels-on" estarem na direção horizontal ou na direção vertical, quando a matriz-original é varrida para a esquerda.

O processo de delimitação das coordenadas inicial e final da linha horizontal superior, quando a matriz-original é varrida da direita para a esquerda, é repetido, até que a função horizontal verifique que não existem mais "pixels-on" na matriz-original que possam ser varridos.

Ao terminar de delimitar a linha horizontal superior, quando a matriz-original é varrida da direita para a esquerda, a coordenada do último "pixel-on" varrido é armazenada como sendo a coordenada CILHS da linha horizontal superior, quando a matriz-original é varrida da direita para a esquerda e, a coordenada do primeiro "pixel-on" varrido ao se varrer a matriz-original da direita para a esquerda (coordenada CFLHS da linha horizontal superior, encontrada quando a matriz-original foi varrida da esquerda para a direita), é considerada como sendo a coordenada CFLHS da linha horizontal superior, quando a matriz-original é varrida da direita para a esquerda. Isto implica que, independente da matriz-original ser varrida da esquerda para a direita ou da direita para a esquerda, a coordenada CFLHS é a mesma. Esta consideração foi feita, devido a se convencionar no programa, que o início de uma linha horizontal sempre está a esquerda de seu "pixel-on" final e o final desta linha sempre está a direita de seu "pixel-on" inicial.

Ao terminar de delimitar as coordenadas inicial e final da linha horizontal superior, quando a matriz-original é varrida da esquerda para a direita e quando esta matriz é varrida da direita para a esquerda, com o objetivo de encontrar entre estas duas linhas a que possui maior comprimento, o programa segue o procedimento descrito a seguir:

Se o valor absoluto da diferença entre as abscissas ACFLHS e ACILHS, encontradas quando a matriz-original é varrida da esquerda para a direita, é menor do que o valor absoluto da diferença entre as abscissas ACFLHS encontrada quando a matriz-original é varrida da esquerda para a direita e ACILHS encontrada quando a matriz-original é varrida da direita para a esquerda e o valor absoluto da diferença entre a ordenada OCILHS encontrada quando a matriz-original é varrida da esquerda para a direita e a ordenada OCFLHS encontrada quando a matriz-original é varrida da direita para a esquerda é menor ou igual a 6 (este passo será referenciado adiante como PASSO 5). O programa verifica se o valor absoluto da diferença entre a abscissa ACFLHS e a abscissa do último "pixel-on" varrido quando a matriz-original é varrida da direita para a esquerda é maior do que 2 e o valor da variável **tot-vert** é menor ou igual a 3. Em caso afirmativo, a coordenada desse último "pixel-on" varrido é considerada como sendo a coordenada CILHS (este passo será referenciado adiante como PASSO 6).

Ao terminar de delimitar a linha horizontal superior que possui maior comprimento, o programa considera que esta linha é a linha horizontal superior do caractere e verifica se o valor absoluto da diferença entre as abscissas ACFLHS e ACILHS é menor ou igual a 1. Em caso afirmativo, a coordenada CILHS é igualada a coordenada CFLHS.

Ao terminar de delimitar as coordenadas inicial e final da linha horizontal superior do caractere, o programa verifica se esta linha realmente pertence ao caractere ou se ela é apenas um ruído (a definição e o que o programa considera ser um ruído foi feita no item 3.5 do capítulo 3 deste trabalho). Esta verificação, tem como objetivo, evitar que alguns ruídos externos ao caractere , que podem não ter sido eliminados pelo algoritmo de afilamento, interfiram no processo de reconhecimento do caractere. Ao varrer a matriz-original da esquerda para a

direita, quando se está delimitando a linha horizontal superior, é anotado na variável **npix-lr** a quantidade de "pixels-on" varridos para a direita, enquanto o valor da variável **"condição-de-folga"** é igual a 0. No primeiro instante em que o valor desta variável torna-se diferente de 0, cessa-se o armazenamento da quantidade de "pixels-on" varridos na variável **npix-lr**. Este mesmo processo é realizado ao varrer a matriz-original na delimitação da linha horizontal superior da direita para a esquerda, porém, utiliza-se a variável **npix-rl** para armazenar a quantidade de "pixels-on" varridos para a esquerda.

Se o valor da variável **npix-rl** é maior do que 4 ou o valor absoluto da diferença entre as variáveis **npix-rl** e **npix-lr** é maior do que 2, o programa verifica se na CONDIÇÃO 3, o "pixel-off" varrido foi considerado como uma falha. Em caso afirmativo, o programa reinicia os processos de delimitação das coordenadas inicial e final da linha horizontal superior, quando a matriz-original é varrida da esquerda para a direita e quando esta matriz é varrida da direita para a esquerda. Caso contrário, o programa finaliza o processo de delimitação das coordenadas inicial e final da linha horizontal superior, quando a matriz-original é varrida da direita para a esquerda.

Se o valor da variável **npix-rl** é menor ou igual a 4 e o valor absoluto da diferença entre as variáveis **npix-rl** e **npix-lr** é menor ou igual a 2, a variável **npix-lr** é igualada a 0 e o programa varre a matriz-original para a direita, a partir da coordenada CILHS. A matriz-original é varrida para a direita, até que não existam mais "pixels-on" que possam ser varridos na direção horizontal ou na direção vertical ou o valor da variável **npix-lr** seja maior do que 4. Enquanto a matriz-original é varrida para a direita, o programa realiza os seguintes passos:

- Se é encontrado um ponto de cruzamento, o valor da variável **ncruz** é incrementado de uma unidade e se o valor da variável **indi** é igual a 0, esta

variável é igualada a 3, indicando que foi encontrado um ponto de cruzamento antes que o valor da variável **"condição-de-folga"** fosse diferente de 0. A variável **ncruz** armazena a quantidade de pontos de cruzamentos encontrados e, a variável **indi,** indica se foi encontrado um ponto de cruzamento antes ou após a variável **"condição-de-folga"** possuir valor diferente de 0.

- Se não existirem mais "pixels-on" que possam ser varridos na direção horizontal ou na direção vertical e o valor da variável **ncruz** é diferente de 0, o valor da variável **ncruz** é decrementado de uma unidade, a variável **indi** é igualada a 0 e o programa retorna a um dos pontos de cruzamento encontrados, enquanto a matriz-original era varrida para a direita e continua a varrer essa matriz na mesma direção. O programa armazena os "pixels-on" varridos, quando a matriz-original foi varrida para a direita, a partir do ponto de cruzamento em que a variável **indi** foi igualada a 0.

- Se o valor da variável **"condição-de-folga"** é igual a 0 e o valor da variável **indi** é igual a 0 ou 3, o valor da variável **npix-lr** é incrementado de uma unidade e o programa continua a varrer a matriz-original para a direita. A matriz-original é varrida para a direita, até que não existam mais "pixels-on" que possam ser varridos nessa direção ou o valor da variável **npix-lr** seja maior do que 4. Caso contrário, o programa verifica se o valor da variável **indi** é diferente de 0. Se isto ocorrer, o programa continua a varrer a matriz-original para a direita. Se o valor da variável **indi** é igual a 0, os seguintes procedimentos são realizados:

  - O programa verifica se o último "pixel-on" varrido não possui

"pixels-on" vizinhos que se encontram nas direções 5, 6 e 7. Em caso afirmativo, o programa anota em quais direções, o "pixel-off" que se encontra na mesma linha desse último "pixel-on" varrido e uma coluna a sua direita possui "pixels-on" vizinhos (este passo será referenciado adiante como PASSO 7). Caso contrário, o programa retorna ao último "pixel-on" varrido, com o objetivo de tentar varrer outras direções na matriz-original.

- O programa verifica se o "pixel-off" encontrado no PASSO 7 possui "pixel-on" vizinho que se encontra na direção 0, 1, 5, 6 ou 7. Em caso afirmativo, a variável **"condição-de-folga"** é igualada a 0 e o programa substitui esse "pixel-off" por um "pixel-on". Caso contrário, a variável **indi** é igualada a 1, indicando que foi encontrado um "pixel-off" e que o valor da variável **"condição-de-folga"** é diferente de 0.

A matriz-original é varrida da esquerda para a direita, até que o valor da variável **npix-lr** seja maior do que 4 ou não existam mais "pixels-on" que possam ser varridos nessa direção.

Após terminar de varrer a matriz-original da esquerda para a direita, o programa verifica se o valor da variável **npix-lr** é menor ou igual a 4. Em caso afirmativo, os "pixels-on" varridos na delimitação da linha horizontal superior, quando a matriz-original é varrida da esquerda para a direita são apagados dessa matriz, pois os mesmos são considerados pelo programa como um ruído e todo o processo de delimitação das coordenadas inicial e final da linha horizontal superior, quando a matriz-original é varrida da esquerda para a direita e quando esta matriz é varrida da direita para a esquerda, explicado acima, é refeito. Caso

contrário, os "pixels-on" varridos quando a matriz-original foi varrida da esquerda para a direita realmente pertencem a linha horizontal superior do caractere.

Linha Horizontal Inferior

O programa considera que a linha horizontal inferior é constituída pelos "pixels-on" que formam o extremo inferior do caractere e denomina a função que delimita as coordenadas inicial e final dessa linha de horizontal.

O programa delimita as coordenadas inicial e final da linha horizontal inferior de cada caractere, varrendo a matriz-original da esquerda para a direita e da direita para a esquerda. Após serem encontradas as coordenadas inicial e final da linha horizontal inferior, ao varrer a matriz-original da esquerda para a direita e da direita para a esquerda, o programa considera como coordenadas inicial e final da linha horizontal inferior, as coordenadas que proporcionarão maior comprimento à linha horizontal inferior. A matriz-original armazena os "pixels-on" e os "pixels-off" que constituem o esqueleto da imagem de um caractere.

Inicialmente, é descrito o processo de delimitação das coordenadas inicial e final da linha horizontal inferior, quando a matriz-original é varrida da esquerda para a direita e, posteriormente, é descrito o processo de delimitação das coordenadas inicial e final da linha horizontal inferior, quando a matriz-original é varrida da direita para a esquerda.

Para facilitar o entendimento do processo de delimitação das coordenadas inicial e final da linha horizontal inferior, quando a matriz-original é varrida da esquerda para direita, optou-se por explicá-lo no texto e esboçar o seu fluxograma. Aconselha-se ao leitor que para sua melhor clareza, leia o texto que descreve a delimitação das coordenadas inicial e final da linha horizontal inferior

e acompanhe ao mesmo tempo o seu fluxograma. A figura 4.4 que se encontra nas páginas 377 a 388 do 2º volume deste trabalho, ilustra o fluxograma que realiza a delimitação das coordenadas inicial e final da linha horizontal inferior, quando a matriz-original é varrida da esquerda para a direita. Nesta figura, alguns comentários que se encontram no texto não constam do fluxograma. Devido à extensão do fluxograma ele foi dividido em 11 partes.

O programa inicia a delimitação das coordenadas inicial e final da linha horizontal inferior de cada caractere, varrendo a matriz-original da esquerda para a direita e de baixo para cima, até encontrar o primeiro "pixel-on". Este "pixel-on" define o início da linha horizontal inferior e o programa armazena a sua coordenada denominando-a de CILHI (coordenada inicial da linha horizontal inferior). A coordenada CILHI (ACILHI, OCILHI) consiste de uma abscissa denominada de abscissa da coordenada inicial da linha horizontal inferior (ACILHI) e de uma ordenada denominada de ordenada da coordenada inicial da linha horizontal inferior (OCILHI).

Após encontrar a coordenada CILHI, o processo de encontrar a coordenada final da linha horizontal inferior, quando a matriz-original é varrida da esquerda para a direita, é semelhante ao processo utilizado na determinação da coordenada final da linha horizontal superior, quando a matriz-original é varrida da esquerda para a direita. Devido a isto, no texto serão descritos somente as modificações que o processo de delimitação da linha horizontal inferior apresenta em relação ao processo de delimitação da linha horizontal superior. Porém, o fluxograma da figura 4.4 mostra o processo completo utilizado na determinação das coordenadas inicial e final da linha horizontal inferior, quando a matriz-original é varrida da esquerda para a direita.

A partir da coordenada CILHI, repete-se o processo utilizado na delimitação da linha horizontal superior no PASSO 1, com a variante que agora o

programa observa se o "pixel-on" a ser procurado na linha horizontal inferior encontra-se na mesma linha da ordenada OCILHI (direção 0), uma linha abaixo (direção 7) ou uma linha acima (direção 1). No PASSO 2, o mesmo processo é repetido, porém, é procurado primeiro um "pixel-on" que esteja na direção 6 e depois na direção 2. No PASSO 3, o processo utilizado é o mesmo. No PASSO 4, o processo é repetido, porém, quando o valor da variável **ncruz** é incrementado de uma unidade pela primeira vez, o programa armazena a coordenada do "pixel-on" que está sendo varrido.

Ao terminar de varrer a matriz-original da esquerda para a direita, o programa verifica se quando foi satisfeito o PASSO 4 alguma coordenada foi armazenada. Em caso afirmativo, o programa verifica se o valor da variável **"condição-vertical"** é igual a 10. Se isto ocorrer, a coordenada armazenada quando a variável **"condição-vertical"** foi igualada a 10, é considerada como sendo a coordenada do último "pixel-on" varrido para a direita na delimitação da linha horizontal inferior, ou seja, a coordenada CFLHI (coordenada final da linha horizontal inferior). A coordenada CFLHI (ACFLHI, OCFLHI) consiste de uma abscissa denominada de abscissa da coordenada final da linha horizontal inferior (ACFLHI) e de uma ordenada denominada de ordenada da coordenada final da linha horizontal inferior (OCFLHI). Independente do valor da variável **"condição-vertical"** ser igual ou não a 10, o programa armazena a coordenada do último "pixel-on" varrido (este passo será referenciado adiante como PASSO 8) e volta a varrer a matriz-original para a direita, a partir da coordenada armazenada no PASSO 4, até que não existam mais "pixels-on" que possam ser varridos nessa direção. Como foi citado acima, pela grande semelhança dos procedimentos de delimitação da linha horizontal superior e da linha horizontal inferior, no procedimento de delimitação da linha horizontal inferior, seguiu-se a mesma numeração dos passos do procedimento de delimitação da linha horizontal

superior. Devido a isto, neste parágrafo, o passo foi numerado como PASSO 8, pois o último passo numerado na delimitação da linha horizontal superior, quando a matriz-original foi varrida da direita para a esquerda, foi o PASSO 7.

Ao terminar de varrer a matriz-original da esquerda para a direita, o programa verifica se o valor da ordenada armazenada no PASSO 8 é maior do que o valor da ordenada do último "pixel-on" varrido na matriz-original da esquerda para a direita. Em caso afirmativo, a coordenada armazenada no PASSO 8 é considerada como sendo a coordenada do último "pixel-on" varrido na delimitação da linha horizontal inferior, quando a matriz-original é varrida da esquerda para a direita, ou seja, a coordenada CFLHI. Caso contrário, a coordenada CFLHI é a coordenada do último "pixel-on" varrido no processo de delimitação das cordenadas inicial e final da linha horizontal inferior, quando a matriz-original é varrida da esquerda para a direita.

A figura 4.5 ilustra a linha horizontal inferior delimitada da esquerda para a direita em 2 números manuscritos.

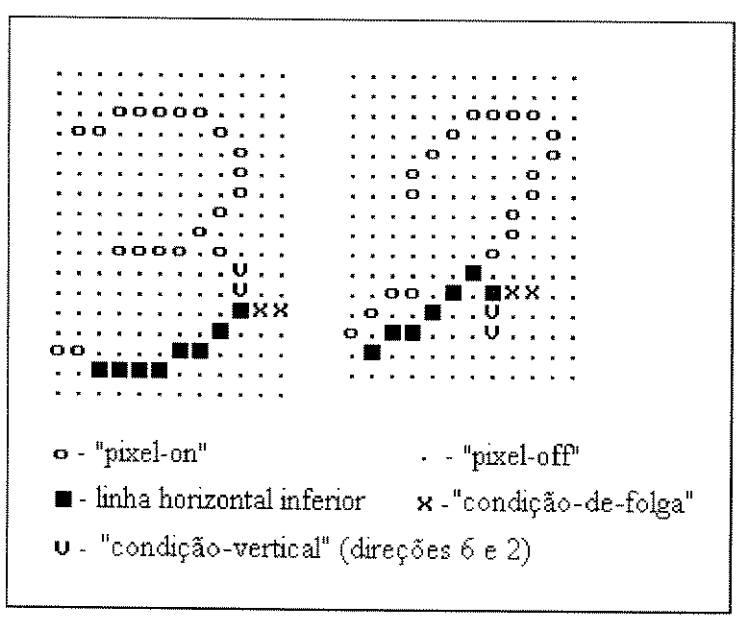


Figura 4.5 - Linha horizontal inferior delimitada da esquerda para a direita em 2 números manuscritos.

Realizada a delimitação das coordenadas inicial e final da linha horizontal inferior da esquerda para a direita, o programa passa então a realizar a delimitação das coordenadas inicial e final da linha horizontal inferior da direita para a esquerda. Aqui também é mostrado o fluxograma e, para maior clareza, o leitor deve seguir as mesmas instruções fornecidas no início da função linha horizontal inferior. A figura 4.6 que se encontra nas páginas 389 a 404 do 2º volume deste trabalho, ilustra o fluxograma que realiza a delimitação das coordenadas inicial e final da linha horizontal inferior, quando a matriz-original é varrida da direita para a esquerda e, devido ao procedimento de delimitação dessa linha, quando a matriz-original é varrida da direita para a esquerda ser semelhante ao procedimento de delimitação das coordenadas inicial e final da linha horizontal superior, quando a matriz-original é varrida da direita para a esquerda, no texto foram descritas apenas as modificações que ocorreram na delimitação das coordenadas inicial e final da linha horizontal inferior, em relação ao procedimento utilizado na delimitação das coordenadas inicial e final da linha horizontal superior. Devido a extensão do fluxograma, o mesmo foi dividido em 15 partes.

No PASSO 5, repete-se o mesmo processo utilizado na delimitação das coordenadas inicial e final da linha horizontal superior, quando a matriz-original é varrida da direita para a esquerda, com exceção que nesse passo, o cálculo do valor absoluto da diferença entre a ordenada OCILHI encontrada, quando a matriz-original é varrida da esquerda para a direita e a ordenada OCILHI encontrada, quando a matriz-original é varrida da direita para a esquerda não é realizado, porque, em alguns caracteres, ao se varrer a matriz-original de baixo para cima para encontrar a coordenada inicial da linha horizontal inferior, esta coordenada CILHI encontrada quando a matriz-original é varrida da esquerda para a direita, está situada em uma posição inferior à coordenada CILHI,

encontrada quando a matriz-original é varrida da direita para a esquerda e, a utilização da coordenada CILHI encontrada quando a matriz-original é varrida da esquerda para a direita, não seria satisfatória, pois esta coordenada não representaria o verdadeiro início da linha horizontal inferior. Se a condição do PASSO 5 é satisfeita, o programa verifica se o valor da variável **ncruz** é diferente de 0, ou seja, se foi encontrado um ponto de cruzamento. Em caso afirmativo, os seguintes procedimentos são executados:

- O programa retorna ao ponto de cruzamento encontrado, que tornou a variável **ncruz** diferente de 0 e varre a matriz-original para baixo na direção 5, 6 ou 7. A matriz-original é varrida para baixo, até que não existam mais "pixels-on" que possam ser varridos nessa direção ou seja encontrada a coordenada CILHI (encontrada quando a matriz-original foi varrida da esquerda para direita) e o número de "pixels-on" varridos na matriz-original para baixo seja maior do que 2.

- Para cada "pixel-on" varrido para baixo, o programa verifica:

  - Se os valores das variáveis **"condição-de-folga"** e **indi** são iguais a 0, o valor da variável **npix-rl** é incrementado de uma unidade. Caso contrário, a variável **indi** é igualada a 1.

  - Se a coordenada do "pixel-on" que está sendo varrido é igual a coordenada CILHI, encontrada quando a matriz-original foi varrida da esquerda para a direita e se neste processo de varrer a matriz-original para baixo são varridos mais do que 2 "pixels-on", a coordenada CILHI encontrada, quando a matriz-original foi varrida da esquerda

para a direita não deve ser alterada, ou seja, não deve ser considerada a coordenada CILHI encontrada quando a matriz-original foi varrida da direita para a esquerda (este passo será referenciado adiante como PASSO 9).

- Ao terminar de varrer a matriz-original para baixo, o programa verifica se existe algum ponto de cruzamento cuja coordenada foi armazenada quando estava varrendo essa matriz da direita para a esquerda ou quando o PASSO 9 estava sendo executado. Em caso afirmativo, o programa retorna para esse ponto de cruzamento e o processo de varrer a matriz-original para baixo é reiniciado.

- Ao terminar de delimitar a linha horizontal inferior, o programa verifica se ((o valor da variável **ncruz** é igual a 0) ou (o valor da variável **ncruz** é diferente de 0 e o PASSO 9 não foi executado)). Em caso afirmativo, se ((o valor absoluto da diferença entre a abscissa ACFLHI e a abscissa do último "pixel-on" varrido na matriz-original da direita para a esquerda é maior do que 2 e o valor da variável **ncruz** é igual a 0) ou (o valor absoluto da diferença entre a abscissa ACFLHI e a abscissa do último "pixel-on" varrido na matriz-original para baixo é maior do que 2 e o valor da variável **ncruz** é diferente de 0)) e o valor da variável **tot-vert** é menor ou igual a 3, a coordenada do último "pixel-on" varrido é armazenada como sendo a coordenada CILHI da linha horizontal inferior que possui maior comprimento (este passo equivale ao PASSO 6 da linha horizontal superior, com as modificações necessárias à linha horizontal inferior).

No PASSO 7, ao delimitar a linha horizontal superior, quando a matriz-

original é varrida da direita para a esquerda, o programa verifica se o último "pixel-on" varrido não possui "pixels-on" vizinhos que se encontram nas direções 5, 6 e 7. Na delimitação da linha horizontal inferior, neste mesmo passo, o programa verifica se o último "pixel-on" varrido não possui "pixels-on" vizinhos que se encontram nas direções 3, 2 e 1, enquanto no PASSO 7, na delimitação da linha horizontal superior, quando a matriz-original é varrida da direita para a esquerda, o programa verificava se o "pixel-off" encontrado neste passo possuía "pixel-on" vizinho que se encontrava na direção 0, 1, 5, 6 ou 7. Na delimitação da linha horizontal inferior, quando a matriz-original é varrida da direita para a esquerda, o programa verifica se o "pixel-off" encontrado possui "pixel-on" vizinho que se encontra na direção 0, 1, 2, 3 ou 6.

Se ((o "pixel-off" varrido não foi considerado uma falha na CONDIÇÃO 3) ou (após terminar de varrer a matriz-original da esquerda para a direita, o valor da variável **npix-lr** é maior do que 4)), o programa verifica se o valor absoluto da diferença entre as abscissas ACFLHI e ACILHI é menor ou igual a 1 ou o valor da variável **tot-vert** é maior ou igual a 40% do valor da variável **vert**. Em caso afirmativo, a coordenada CFLHI é igualada a coordenada CILHI. Quando o valor da variável **tot-vert** é maior ou igual a 40% do valor da variável **vert**, isto significa que existe uma quantidade muito grande de "pixels-on" na direção vertical, o que resulta em uma descaracterização da linha horizontal inferior. Para minimizar este problema, o programa iguala a coordenada CFLHI à coordenada CILHI. A variável **vert** indica a quantidade de "pixels-on" varridos na direção vertical, quando a linha horizontal inferior foi delimitada da direita para a esquerda.

Como foi dito na introdução deste capítulo, a função horizontal determina as coordenadas inicial e final da linha horizontal superior, se o programa estiver delimitando essa linha e, determina as coordenadas inicial e final da linha

horizontal inferior, caso seja essa a linha a ser delimitada.

Linha Vertical Esquerda

O programa considera que os "pixels-on" que formam o lado esquerdo do caractere constituem a linha vertical esquerda e denomina a função que delimita as coordenadas inicial e final dessa linha de vertical.

Para facilitar o entendimento do processo de delimitação das coordenadas inicial e final da linha vertical esquerda, optou-se por explicá-lo no texto e esboçar o seu fluxograma. Aconselha-se ao leitor que para sua melhor clareza, leia o texto que descreve a delimitação da linha vertical esquerda e acompanhe ao mesmo tempo o seu fluxograma. A figura 4.7 que se encontra nas páginas 405 a 411 do 2º volume deste trabalho, ilustra o fluxograma que realiza a delimitação das coordenadas inicial e final da linha vertical esquerda. Nesta figura, alguns comentários que se encontram no texto não constam do fluxograma. Devido à extensão do fluxograma, ele foi dividido em 6 partes.

Ao encontrar o esqueleto de cada caractere, é verificado que os "pixels-on" que constituem a linha vertical do caractere não se encontram todos alinhados e, que em alguns casos, podem até mesmo ocorrer regiões do caractere que apresentam ausência de "pixels-on". Para minimizar este problema e tornar o algoritmo mais flexível, foram definidas no programa as variáveis **"condição-de-folga"** e **"condição-horizontal"**. O objetivo da variável **"condição-horizontal"** é indicar se o "pixel-on" que está sendo varrido encontra-se na direção horizontal (direção 0 ou 4). Se o "pixel-on" que está sendo varrido encontra-se na direção horizontal, o valor da variável **"condição-horizontal"** é diferente de 0, sendo que esse valor é proporcional à quantidade de "pixels-on" varridos que se encontram na direção horizontal. Se o "pixel-on" que está sendo varrido encontra-se na

direção vertical (direção 2 ou 6), o valor da variável **"condição-horizontal"** é igual a 0.

O programa inicia a delimitação das coordenadas inicial e final da linha vertical esquerda a partir da coordenada CILHS (coordenada inicial da linha horizontal superior). Devido a isto, a coordenada CILVE (coordenada inicial da linha vertical esquerda) sempre coincide com a coordenada CILHS. A coordenada CILVE (ACILVE, OCILVE) consiste de uma abscissa denominada de abscissa da coordenada inicial da linha vertical esquerda (ACILVE) e de uma ordenada denominada de ordenada da coordenada inicial da linha vertical esquerda (OCILVE).

A partir da coordenada CILVE, é procurado um novo "pixel-on" na matriz-original, que deve estar localizado uma linha abaixo do "pixel-on" de ordenada OCILVE, na mesma coluna da abscissa ACILVE (direção 6), uma coluna a esquerda (direção 5) ou a direita (direção 7) (este passo será referenciado adiante como PASSO 1). A matriz-original armazena os "pixels-on" e os "pixels-off" que constituem o esqueleto da imagem de um caractere.

Se ((o PASSO 1 não foi satisfeito e o valor da variável **"condição-de-folga"** é igual a 0) ou (o PASSO 1 foi satisfeito e o valor da variável **"condição-horizontal"** é diferente de 0)), o programa realiza os seguintes procedimentos:

- Se o PASSO 1 não foi satisfeito, o programa procura um "pixel-on" que esteja na direção 4 ou 0, ou seja, um "pixel-on" que esteja na mesma linha do "pixel-on" de ordenada OCILVE e uma coluna a esquerda (direção 4) ou a direita (direção 0) do "pixel-on" de abscissa ACILVE (este passo será referenciado adiante como PASSO 2).

- Se (o valor da variável **dir** é igual a 0 ou 4) e (o valor da variável **ndir** é

diferente de 2 e 6), o "pixel-on" que está sendo varrido é considerado como um "pixel-on" que se encontra na direção horizontal, logo, o valor da variável **"condição-horizontal"** é incrementado de uma unidade. Caso contrário, se o valor da variável **ndir** é igual a 2 ou 6, o "pixel-on" que está sendo varrido encontra-se na direção vertical, portanto, a variável **"condição-horizontal"** é igualada a 0. A variável **dir** armazena a direção que se encontra o último "pixel-on" varrido em relação ao "pixel-on" que está sendo varrido e, a variável **ndir,** armazena a direção que se encontra o "pixel-on" que está sendo varrido em relação ao próximo "pixel-on" que será varrido ("pixel-on" encontrado no PASSO 1 ou 2, vizinho ao "pixel", "on" ou "off", que está sendo varrido).

- Se o PASSO 2 foi satisfeito, o valor da variável **"condição-horizontal"** é incrementado de uma unidade. Caso contrário, a variável **"condição-horizontal"** é igualada a 3. Após isto, o programa verifica se o valor da variável **"condição-horizontal"** é maior ou igual a 3. Em caso afirmativo, o programa retorna ao último "pixel-on" varrido enquanto o valor da variável **"condição-horizontal"** era igual a 0 e, esta variável que possuía o valor maior ou igual a 3, é igualada a 0.

Se não foi encontrado "pixel-on" que pudesse ser varrido no PASSO 1 ou 2, o valor da variável **"condição-horizontal"** era maior ou igual a 3 no PASSO 2 ou este passo não foi executado porque o valor da variável **"condição-de-folga"** é diferente de 0, o programa armazena a coordenada do "pixel-on" que está sendo varrido (este passo será referenciado adiante como PASSO 3) e os seguintes procedimentos são executados:

- Se ao adicionar uma unidade ao valor da ordenada do "pixel-on" que está sendo varrido, este novo valor da ordenada não exceder os limites da matriz-original e o valor da variável **"condição-de-folga"** é diferente de 1, o valor da variável **"condição-de-folga"** é incrementado de uma unidade e o "pixel-off" ("pixel" com valor igual a 0) da coordenada encontrada, ao se incrementar o valor da ordenada do "pixel-on" que está sendo varrido, é varrido usando-se os mesmos critérios que foram utilizados na procura de um "pixel-on" vizinho ao "pixel-on" que estava sendo varrido, ou seja, o programa realiza o PASSO 1 partindo desse "pixel-off". O objetivo de varrer a matriz-original a partir do "pixel-off" dessa coordenada, é tentar encontrar novos "pixels-on" que possam ser varridos.

- Se ao adicionar uma unidade ao valor da ordenada do "pixel-on" que está sendo varrido, este novo valor da ordenada exceder os limites da matriz-original ou o valor da variável **"condição-de-folga"** é igual a 1, o programa retorna ao "pixel-on" cuja coordenada foi armazenada no PASSO 3 e o processo de delimitação das coordenadas inicial e final da linha vertical esquerda é finalizado.

- Se o PASSO 1 ou o PASSO 2 é satisfeito, a variável **"condição-de-folga"** é igualada a 0, o "pixel-on" vizinho ao "pixel-on" que está sendo varrido será o próximo "pixel-on" a ser varrido e se o valor da variável **ndir** é igual a 0, 2, 4 ou 6, o valor da variável **dir** é igualado ao valor da variável **ndir**.

- Enquanto a matriz-original é varrida para baixo na delimitação da linha vertical esquerda, se é encontrado um ponto de cruzamento que possui um "pixel-on"

vizinho que se encontra na mesma linha (direção 4) do "pixel-on" que está sendo varrido, o "pixel-on" que se encontra na direção 4 tem preferência sobre os outros "pixels-on" vizinhos ao "pixel-on" que está sendo varrido. Após isto, a matriz-original continua a ser varrida para baixo, a partir do "pixel-on" vizinho que se encontra na direção 4.

A matriz-original é varrida para baixo na delimitação das coordenadas inicial e final da linha vertical esquerda, até que o valor da variável **"condição-de-folga"** seja maior ou igual a 1 ou seja encontrada a coordenada CILHI (coordenada inicial da linha horizontal inferior). Existem caracteres em que, a partir da coordenada CILHS, é possível encontrar a coordenada CILHI ou mesmo ultrapassá-la. Com o objetivo de possibilitar que nesses caracteres a coordenada CFLVE coincida com a coordenada CILHI, o processo de delimitação da linha vertical esquerda é finalizado quando a coordenada CILHI é encontrada. Alguns caracteres que possuem essas características são os números 0, 8, 6 e 1. Ao encontrar o último "pixel-on" da linha vertical esquerda, o programa denomina a sua coordenada de CFLVE (coordenada final da linha vertical esquerda). A coordenada CFLVE (ACFLVE, OCFLVE) consiste de uma abscissa denominada de abscissa da coordenada final da linha vertical esquerda (ACFLVE) e de uma ordenada denominada de ordenada da coordenada final da linha vertical esquerda (OCFLVE).

Se o valor absoluto da diferença entre as coordenadas CILVE e CFLVE é menor ou igual a 1, a coordenada CFLVE é igualada a coordenada CILVE.

A figura 4.8 ilustra a linha vertical esquerda em 2 números manuscritos.

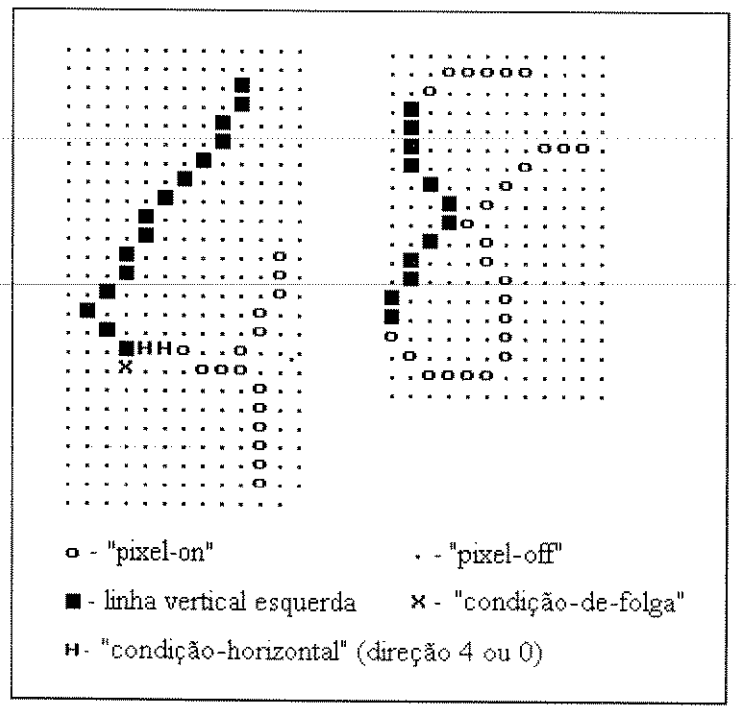


Figura 4.8 - Linha vertical esquerda em 2 números manuscritos.

Ponto de Cruzamento

Ponto de cruzamento é uma região do caractere em que um "pixel-on" possui 2 ou mais "pixels-on" vizinhos e este "pixel-on" pode estar situado em posições aleatórias do caractere. O programa denomina a função que analisa e armazena a coordenada dos pontos de cruzamento encontrados de cruzamento.

O programa considera que existem dois tipos de pontos de cruzamento. No primeiro tipo, o ponto de cruzamento é representado pelo "pixel-on" vizinho ao "pixel-on" que está sendo varrido e, devido a isto, a coordenada desse "pixel-on" vizinho é armazenada. O "pixel-on" vizinho é escolhido de acordo com a direção em que a matriz-original está sendo varrida. É importante ressaltar que a medida em que a matriz-original está sendo varrida os "pixels-on" varridos são apagados dessa matriz. A matriz-original armazena os "pixels-on" e os "pixels-off" que constituem o esqueleto da imagem de um caractere. A figura 4.9 ilustra os vários

casos existentes do primeiro tipo de ponto de cruzamento considerado pelo programa.

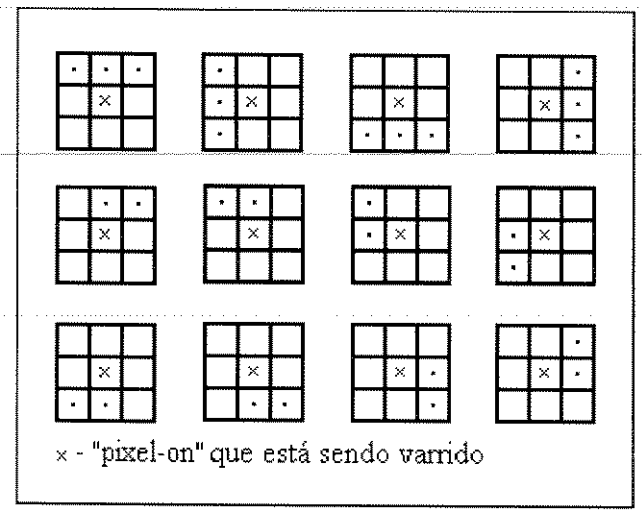


Figura 4.9 - Vários casos existentes do primeiro tipo de ponto de cruzamento considerado pelo programa.

No segundo tipo, o ponto de cruzamento é representado pelo "pixel-on" que está sendo varrido e, devido a isto, a coordenada deste "pixel-on" é armazenada. A figura 4.10 ilustra os casos em que mais ocorrem o segundo tipo de ponto de cruzamento considerado pelo programa.

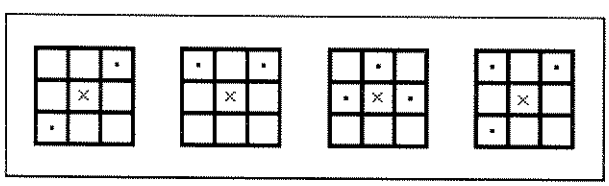


Figura 4.10 - Os casos em que mais ocorrem o segundo tipo de ponto de cruzamento considerado pelo programa.

A figura 4.11 ilustra o ponto de cruzamento em 2 números manuscritos.

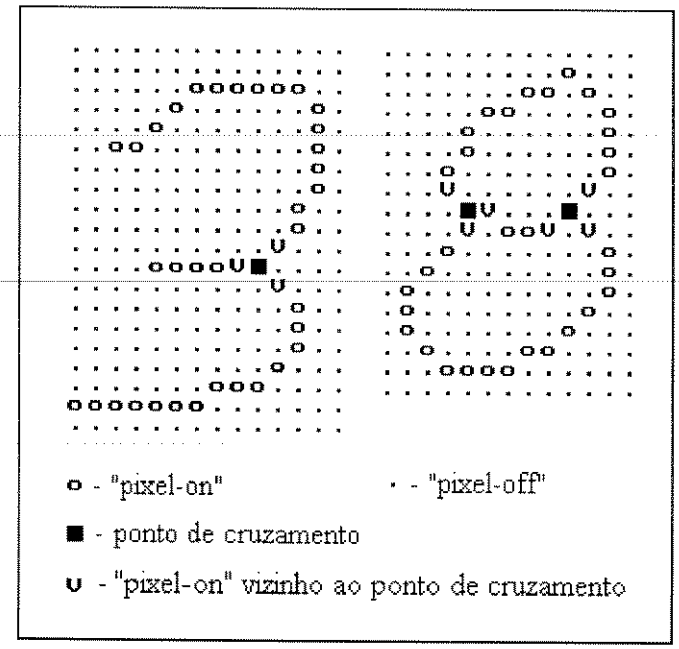


Figura 4.11 - Ponto de cruzamento em 2 números manuscritos.

Além das funções linha horizontal superior, linha horizontal inferior, linha vertical esquerda e ponto de cruzamento descritas anteriormente, foram implementadas no programa, as funções vertical e horizontal. A implementação destas funções teve como objetivo possibilitar ao programa varrer a matriz-original na direção vertical e na direção horizontal. A seguir são descritas as funções vertical e horizontal.

Vertical

Com o objetivo de varrer a matriz-original na direção vertical (direção 2, 1, 3, 6, 7 ou 5), foi criada a função denominada no programa de vert. A matriz-original armazena os "pixels-on" e os "pixels-off" que constituem o esqueleto da imagem de um caractere.

Para varrer a matriz-original na direção vertical, a função vertical recebe a coordenada de um "pixel" (este "pixel" pode ser "on" ou "off") a ser varrido

(denominado no texto por "pixel" que está sendo varrido) e verifica se este "pixel" possui "pixel"-on" vizinho (denominado no texto de próximo "pixel-on" a ser varrido) na direção 2, 1 ou 3, se a matriz-original está sendo varrida para cima ("up"), ou na direção 6, 7 ou 5, se a matriz-original está sendo varrida para baixo ("down").

Na função vertical existem duas opções que possibilitam varrer o "pixel" na direção vertical. Se a opção para varrer o "pixel" é "up" (para cima) o próximo "pixel-on" a ser varrido deverá estar acima (direção 2, 1 ou 3) do "pixel" que está sendo varrido. Se a opção para varrer o "pixel" é "down" (para baixo) o próximo "pixel-on" a ser varrido deverá estar abaixo (direção 6, 7 ou 5) do "pixel" que está sendo varrido.

A função vertical devolve à função que a chamou, a coordenada do próximo "pixel-on" a ser varrido, a direção que se encontra este "pixel-on" em relação ao "pixel" cuja coordenada foi recebida por esta função, a posição, segundo o código de cadeia de 8 direções dos "pixels-on" vizinhos do "pixel" recebido e devolve a quantia 1 se o "pixel" que estava sendo varrido nesta função, não possui mais "pixels-on" vizinhos que possam ser varridos na direção vertical, devolve a quantia 0 se o "pixel" que estava sendo varrido, nesta função, possuir "pixels-on" vizinhos que possam ser varridos na direção vertical (observando a opção "up" ou "down" desejada) ou devolve o número de "pixels-on" vizinhos ao ponto de cruzamento, se o "pixel" que estava sendo varrido for um ponto de cruzamento.

Ao terminar de varrer um "pixel", se ele for um "pixel-on", a função vertical apaga este "pixel-on" da matriz-original, com o objetivo de que ele não interfira quando um "pixel-on" vizinho for varrido.

Para facilitar o entendimento do processo utilizado pela função vertical, optou-se por explicá-lo no texto e esboçar o seu fluxograma. Aconselha-se ao

leitor que para sua melhor clareza, leia o texto que descreve a função vertical e acompanhe ao mesmo tempo o seu fluxograma. A figura 4.12 que se encontra nas páginas 412 a 422 do 2º volume deste trabalho, ilustra o fluxograma da função vertical. Nesta figura, alguns comentários que se encontram no texto não constam do fluxograma. Devido à extensão do fluxograma, ele foi dividido em 10 partes.

Se o "pixel" (este "pixel" pode ser "on" ou "off") que está sendo varrido pela função vertical possui 2 "pixels-on" vizinhos, o programa verifica se este "pixel" é um ponto de cruzamento. Esta verificação é realizada porque, em alguns casos, o "pixel" pertencente a um ponto de cruzamento varrido anteriormente, pode vir a fazer parte da análise do "pixel" que está sendo varrido, gerando deste modo, um novo ponto de cruzamento que no caso é indesejável. Portanto, o "pixel-on" pertencente a um ponto de cruzamento varrido anteriormente deve ser desconsiderado. A seguir, é descrito o procedimento que verifica se o "pixel" que está sendo varrido pela função vertical é um ponto de cruzamento:

- Se ((a opção para varrer o "pixel" é "down" (direção 6, 5 ou 7) e o "pixel" a ser desconsiderado encontra-se na direção 2, 1 ou 3) ou (a opção para varrer o "pixel" é "up" (direção 2, 1 ou 3) e o "pixel" a ser desconsiderado encontra-se na direção 6, 5 ou 7)), o programa armazena a coordenada do "pixel" que está sendo varrido.

- Se anteriormente foram armazenadas coordenadas de pontos de cruzamento, é verificado se o valor absoluto da diferença entre a coordenada do "pixel" que está sendo varrido e a coordenada do último ponto de cruzamento armazenado é menor ou igual a 1. Em caso afirmativo, o "pixel" que está sendo varrido não é considerado um ponto de cruzamento.

- Se não existirem coordenadas de pontos de cruzamento armazenadas e se (a opção para varrer o "pixel" é "down" (direção 6, 5 ou 7) e (o "pixel" a ser desconsiderado encontra-se na direção 3 ou 1) e (o "pixel" que está sendo varrido encontra-se na direção 6)) ou (a opção para varrer o "pixel" é "up" (direção 2, 1 ou 3) e (o "pixel" a ser desconsiderado encontra-se na direção 5 ou 7) e (o "pixel" que está sendo varrido encontra-se na direção 2)), o programa considera que o "pixel" que está sendo varrido não é um ponto de cruzamento.

Após a verificação da existência ou não de um ponto de cruzamento, o programa realiza a análise dos "pixels-on" vizinhos ao "pixel" que está sendo varrido da seguinte maneira:

- Se a opção para varrer o "pixel" é "up" (direção 2, 1 ou 3), o programa procura "pixels-on" vizinhos que se encontram na direção 2, 1 ou 3. Se existir "pixel-on" vizinho em uma dessas direções, é anotada a direção em que o "pixel-on" vizinho se encontra.

- Se a opção para varrer o "pixel" é "down" (direção 6, 5 ou 7), o programa procura "pixels-on" vizinhos que se encontram na direção 6, 5 ou 7. Se existir "pixel-on" vizinho em uma dessas direções, é anotada a direção em que o "pixel-on" vizinho se encontra.

Se for verificado que o "pixel" que está sendo varrido é realmente um ponto de cruzamento, ou seja, este "pixel" possui 2 ou mais "pixels-on" vizinhos, o programa armazena a sua coordenada (este passo será referenciado adiante como PASSO 1) e independente da matriz-original estar sendo varrida para cima

(opção "up")" ou para baixo (opção "down"), se o "pixel" que está sendo varrido possui "pixels-on" vizinhos que se encontram nas direções 4 e (3 ou 5), o programa anota a direção 4 e estabelece que esta direção prevalece sobre as demais direções anotadas. Caso contrário, se o "pixel" que está sendo varrido possui "pixels-on" vizinhos que se encontram nas direções 0 e (1 ou 7), o programa anota a direção 0 e estabelece que esta direção prevalece sobre as demais direções anotadas. Quando o programa estabelece que a direção 4 ou 0 (direção horizontal) prevalece sobre as demais direções anotadas, ele está possibilitando que seja varrido um "pixel" na direção horizontal e que posteriormente a matriz-original continue a ser varrida na direção vertical.

Ao encontrar o esqueleto de cada caractere, é verificado que os "pixels-on" que constituem a linha vertical do caractere não se encontram todos alinhados e que, em alguns casos, podem até mesmo ocorrer regiões do caractere que apresentam ausência de "pixels-on". Para minimizar este problema e tornar o algoritmo mais flexível, foram definidas no programa as variáveis **"condição-de-folga"** e **"condição-horizontal"**, que são utilizadas da seguinte maneira:

Se em alguma das análises descritas anteriormente nesta função foi anotada a direção de algum "pixel-on" vizinho ao "pixel" que está sendo varrido, a variável **"condição-horizontal"** é igualada a 0. Se não for encontrado nenhum "pixel-on" vizinho ao "pixel" que está sendo varrido e o valor da variável **"condição-de-folga"** for igual a 0, o programa verifica se o valor da variável **"condição-horizontal"** é igual a 0. Em caso afirmativo, a coordenada do "pixel" que está sendo varrido é armazenadada. Independente do valor da variável **"condição-horizontal"** ser igual a 0 ou não, os seguintes procedimentos são realizados:

- Se existe um "pixel-on" vizinho ao "pixel" que está sendo varrido que se encontra na direção 4, o programa anota esta direção. Caso contrário, se existe um "pixel-on" vizinho ao "pixel" que está sendo varrido que se encontra na direção 0, o programa anota esta direção.

- Se for encontrado na direção 4 ou 0, "pixel-on" vizinho ao "pixel" que está sendo varrido, o valor da variável **"condição-horizontal"** é incrementado de uma unidade. Caso contrário, a variável **"condição-horizontal"** é igualada a 0, o programa considera que não foi encontrado "pixel-on" vizinho ao "pixel" que está sendo varrido e retorna ao "pixel" cuja coordenada foi armazenada quando o valor da variável **"condição-horizontal"** era igual a 0.

- Se após as análises realizadas nos itens acima, não for encontrado "pixel-on" vizinho ao "pixel" que está sendo varrido e esse "pixel-on" vizinho não pode ser varrido, o programa executa o Procedimento 1. Caso contrário, se for encontrado "pixel-on" vizinho ao "pixel" que está sendo varrido e esse "pixel-on" vizinho pode ser varrido, o Procedimento 2 é executado. Estes 2 procedimentos são descritos a seguir.

Procedimento 1

- Se o valor da variável **"condição-de-folga"** é igual a 0, o programa armazena a coordenada do "pixel" que está sendo varrido.

- Se a opção para varrer o "pixel" for "up" (direção 2, 1 ou 3), o programa decrementa de uma unidade a ordenada do "pixel" que está sendo varrido e se

este valor da ordenada exceder os limites da matriz-original, o programa considera que não existem mais "pixels-on" que possam ser varridos na direção vertical, isto é, o final da linha vertical foi encontrado. Caso a opção para varrer o "pixel" seja "down" (direção 6, 5 ou 7), o programa incrementa de uma unidade a ordenada do "pixel" que está sendo varrido e se este valor da ordenada exceder os limites da matriz-original, o final da linha vertical foi encontrado.

- Se foi encontrado o final da linha vertical ou o valor da variável **"condição-de-folga"** é igual a 1, o programa retorna ao "pixel" cuja coordenada foi armazenada quando o valor da variável **"condição-de-folga"** era igual a 0 e a função vertical retorna o valor 1 à função que a chamou, indicando que não foi encontrado "pixel-on" que pudesse ser varrido na direção desejada. Caso contrário, o valor da variável **"condição-de-folga"** é incrementado de uma unidade.

Procedimento 2

A variável **"condição-de-folga"** é igualada a 0 e a função vertical retorna à função que a chamou, a coordenada do "pixel-on" vizinho ao "pixel" que está sendo varrido, encontrado nos procedimentos que antecedem o Procedimento 1. A coordenada do "pixel-on" vizinho é retornada como sendo a coordenada do próximo "pixel-on" a ser varrido. Se no PASSO 1 foi armazenada a coordenada de um ponto de cruzamento e o "pixel" que está sendo varrido não é o primeiro "pixel" a ser varrido pela função vertical, a função ponto de cruzamento é executada, visando classificar o tipo de ponto de cruzamento formado pelo "pixel" que está sendo varrido e pelos "pixels-on" vizinhos a ele. A função ponto

de cruzamento é executada com o objetivo de armazenar a coordenada do "pixel" que define o ponto de cruzamento, isto é, o "pixel" que está sendo varrido ou um "pixel-on" vizinho a ele.

Independente do Procedimento 1 ou 2 ter sido executado, o programa verifica se o final da linha vertical não foi encontrado, foi encontrado ponto de cruzamento e o "pixel" que está sendo varrido não é o primeiro "pixel" a ser varrido pela função vertical. Em caso afirmativo, a função vertical retorna à função que a chamou, a quantidade de "pixels-on" vizinhos ao ponto de cruzamento e a coordenada do próximo "pixel" a ser varrido. Caso contrário, se o "pixel" que está sendo varrido é o primeiro "pixel" a ser varrido pela função vertical, o programa retorna à função que a chamou, a coordenada do próximo "pixel" a ser varrido e o valor 0, indicando que o "pixel" que está sendo varrido possui apenas um "pixel-on" vizinho.

A figura 4.13 ilustra os "pixels-on" de 2 números manuscritos, varridos pela função vertical.

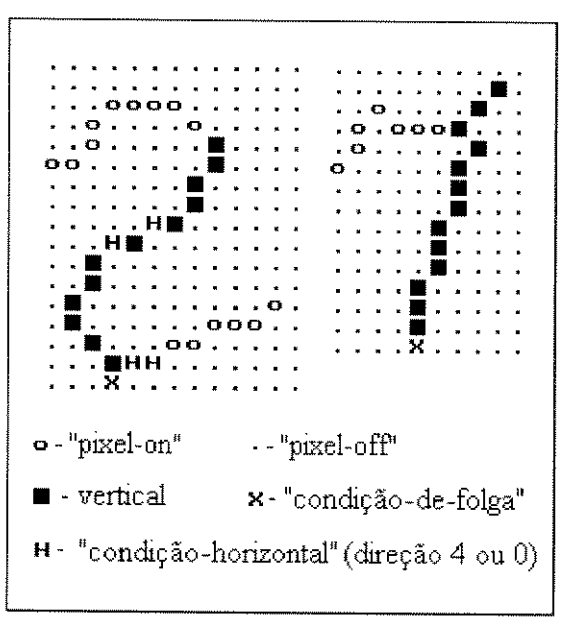


Figura 4.13 - Números manuscritos varridos pela função vertical.

Horizontal

Com o objetivo de varrer a matriz-original na direção horizontal (direção (1, 0, 7, 3, 4 ou 5), foi criada a função denominada no programa de horiz. A matriz-original armazena os "pixels-on" e os "pixels-off" que constituem o esqueleto da imagem de um caractere.

Para varrer a matriz-original na direção horizontal, a função horizontal recebe a coordenada de um "pixel" (este "pixel" pode ser "on" ou "off") a ser varrido (denominado no texto por "pixel" que está sendo varrido) e verifica se este "pixel" possui "pixel-on" vizinho (denominado no texto de próximo "pixel-on" a ser varrido) na direção 0, 1 ou 7, se a matriz-original está sendo varrida para a direita ("right") ou direção 4, 3 ou 5, se a matriz-original está sendo varrida para a esquerda ("left").

Na função horizontal, existem duas opções que possibilitam varrer o "pixel" na direção horizontal. Se a opção para varrer o "pixel" for "right" (para a direita), o próximo "pixel-on" a ser varrido deverá estar à direita (direção 0, 1 ou 7) do "pixel" que está sendo varrido. Se a opção para varrer o "pixel" for "left" (para a esquerda), o próximo "pixel-on" a ser varrido deverá estar à esquerda (direção 4, 3 ou 5) do "pixel" que está sendo varrido.

A função horizontal devolve à função que a chamou, a coordenada do próximo "pixel-on" a ser varrido, a direção que se encontra este "pixel-on" em relação ao "pixel" cuja coordenada foi recebida por esta função, a posição, segundo o código de cadeia de 8 direções, dos "pixels-on" vizinhos do "pixel" recebido e devolve a quantia 1 se o "pixel" que estava sendo varrido nesta função, não possui mais "pixels-on" vizinhos que possam ser varridos na direção horizontal. Devolve a quantia 0 se o "pixel" que estava sendo varrido nesta função possuir "pixels-on" vizinhos que possam ser varridos na direção

horizontal (observando a opção "right" ou "left" desejada) ou devolve o número de "pixels-on" vizinhos ao ponto de cruzamento, se o "pixel" que estava sendo varrido for um ponto de cruzamento.

Ao terminar de varrer um "pixel-on", se ele for um "pixel-on", a função horizontal apaga este "pixel-on" da matriz-original, para que ele não interfira quando um "pixel-on" vizinho for varrido.

Para facilitar o entendimento do processo utilizado pela função horizontal, optou-se por explicá-lo no texto e esboçar o seu fluxograma. Aconselha-se ao leitor que para sua melhor clareza, leia o texto que descreve a função horizontal e acompanhe ao mesmo tempo o seu fluxograma. A figura 4.14 que se encontra nas páginas 423 a 433 do 2º volume deste trabalho, ilustra o fluxograma da função horizontal. Nesta figura, alguns comentários que se encontram no texto não constam do fluxograma. Devido à extensão do fluxograma ele foi dividido em 10 partes.

Se o "pixel" que está sendo varrido pela função horizontal possui 2 "pixels-on" vizinhos, o programa verifica se este "pixel" é um ponto de cruzamento. Esta verificação é realizada porque, em alguns casos, o "pixel-on" pertencente a um ponto de cruzamento varrido anteriormente, pode vir a fazer parte da análise do "pixel" que está sendo varrido, gerando deste modo um novo ponto de cruzamento, que no caso é indesejável. Portanto, o "pixel-on" pertencente a um ponto de cruzamento varrido anteriormente deve ser desconsiderado. A seguir é descrito o procedimento que verifica se o "pixel" que está sendo varrido pela função horizontal é um ponto de cruzamento:

- Se ((a opção para varrer o "pixel" é "left" (direção 4, 3 ou 5) e o "pixel" a ser desconsiderado encontra-se na direção 0, 1 ou 7) ou (a opção para varrer o "pixel" é "right" (direção 0, 1 ou 7) e o "pixel" a ser desconsiderado encontra-

se na direção 4, 3 ou 5)), o programa armazena a coordenada do "pixel" que está sendo varrido.

- Se anteriormente foram armazenadas as coordenadas de pontos de cruzamento, é verificado se o valor absoluto da diferença entre a coordenada do "pixel" que está sendo varrido e a coordenada do último ponto de cruzamento armazenado é menor ou igual a 1. Em caso afirmativo, o "pixel" que está sendo varrido não é considerado um ponto de cruzamento.

- Se não existirem coordenadas de pontos de cruzamento armazenadas e se (a opção para varrer o "pixel" é "left" (direção 4, 3 ou 5) e (o "pixel" a ser desconsiderado encontra-se na direção 1 ou 7) e (o "pixel" que está sendo varrido encontra-se na direção 4)) ou (a opção para varrer o "pixel" é "right" (direção 0, 1 ou 7) e (o "pixel" a ser desconsiderado encontra-se na direção 3 ou 5) e (o "pixel" que está sendo varrido encontra-se na direção 0)), o programa considera que o "pixel" que está sendo varrido não é um ponto de cruzamento.

Após a verificação da existência ou não de ponto de cruzamento, o programa realiza a análise dos "pixels-on" vizinhos ao "pixel" que está sendo varrido da seguinte maneira:

- Se a opção para varrer o "pixel" é "right" (direção 0, 1 ou 7), o programa procura "pixels-on" vizinhos que se encontram na direção 0, 1 ou 7. Se existir "pixel-on" vizinho em uma dessas direções, é anotada a direção em que o "pixel-on" vizinho se encontra.

- Se a opção para varrer o "pixel" é "left" (direção 4, 3 ou 5), o programa procura "pixels-on" vizinhos que se encontram na direção 4, 3 ou 5. Se existir "pixel-on" vizinho em uma dessas direções, é anotada a direção em que o "pixel-on" vizinho se encontra.

Se for verificado que o "pixel" que está sendo varrido é realmente um ponto de cruzamento, ou seja, este "pixel" possui 2 ou mais "pixels-on" vizinhos, o programa armazena a sua coordenada (este passo será referenciado adiante como PASSO 1) e, independente da matriz-original estar sendo varrida para a direita (opção "right")" ou para a esquerda (opção "left"), se o "pixel" que está sendo varrido possui "pixels-on" vizinhos que se encontram na direção 2 e (1 ou 3), o programa anota a direção 2 e estabelece que esta direção prevalece sobre as demais direções anotadas. Caso contrário, se o "pixel" que está sendo varrido possui "pixels-on" vizinhos que se encontram na direção 6 e (5 ou 7), o programa anota a direção 6 e estabelece que esta direção prevalece sobre as demais direções anotadas. Quando o programa estabelece que a direção 2 ou 6 (direção vertical) prevalece sobre as demais direções anotadas, ele está possibilitando que seja varrido um "pixel" na direção vertical e que posteriormente a matriz-original continue a ser varrida na direção horizontal.

Ao encontrar o esqueleto de cada caractere, é verificado que os "pixels-on" que constituem a linha horizontal do caractere, não se encontram todos alinhados e que, em alguns casos, pode até mesmo ocorrer regiões do caractere que apresentam ausência de "pixels-on". Para minimizar este problema e tornar o algoritmo mais flexível, foram definidas no programa as variáveis **"condição-de-folga"** e **"condição-vertical"**, que são utilizadas da seguinte maneira:

Se em alguma das análises realizadas anteriormente nesta função, foi

anotada a direção de algum "pixel-on" vizinho ao "pixel" que está sendo varrido, a variável **"condição-vertical"** é igualada a 0. Se não for encontrado nenhum "pixel-on" vizinho ao "pixel" que está sendo varrido e o valor da variável **"condição-de-folga"** é igual a 0, o programa verifica se o valor da variável **"condição-vertical"** é igual a 0. Em caso afirmativo, a coordenada do "pixel" que está sendo varrido é armazenada. Independente do valor da variável **"condição-vertical"** ser igual a 0 ou não, os seguintes procedimentos são realizados:

- Se existe um "pixel-on" vizinho ao "pixel" que está sendo varrido que se encontra na direção 2, o programa anota esta direção. Caso contrário, se existe um "pixel-on" vizinho ao "pixel" que está sendo varrido que se encontra na direção 6, o programa anota esta direção.

- Se é encontrado na direção 2 ou 6, "pixel-on" vizinho ao "pixel" que está sendo varrido, o valor da variável **"condição-vertical"** é incrementado de uma unidade. Caso contrário, a variável **"condição-vertical"** é igualada a 0, o programa considera que não foi encontrado "pixel-on" vizinho ao "pixel" que está sendo varrido e retorna ao "pixel" cuja coordenada foi armazenada quando o valor da variável **"condição-vertical"** era igual a 0.

- Se após as análises realizadas nos itens acima, não for encontrado "pixel-on" vizinho ao "pixel" que está sendo varrido e esse "pixel-on" vizinho não pode ser varrido, o programa executa o Procedimento 1. Caso contrário, se for encontrado "pixel-on" vizinho ao "pixel" que está sendo varrido e esse "pixel-on" vizinho pode ser varrido, o Procedimento 2 é executado. Estes 2

procedimentos são descritos a seguir.

Procedimento 1

- Se o valor da variável **"condição-de-folga"** é igual a 0, o programa armazena a coordenada do "pixel" que está sendo varrido.

- Se a opção para varrer o "pixel" for "right" (direção 0, 1 ou 7), o programa incrementa de uma unidade a abscissa do "pixel" que está varrido e se este valor da abscissa exceder os limites da matriz-original, o programa considera que não existem mais "pixels-on" que possam ser varridos na direção horizontal, isto é, o final da linha horizontal foi encontrado. Caso a opção seja "left" (direção 4, 3 ou 5), o programa decrementa de uma unidade a abscissa do "pixel" que está sendo varrido e se este valor da abscissa exceder os limites da matriz-original, o final da linha horizontal foi encontrado.

- Se foi encontrado o final da linha horizontal ou o valor da variável **"condição-de-folga"** é igual a 1, o programa retorna ao "pixel" cuja coordenada foi armazenada quando o valor da variável **"condição-de-folga"** era igual a 0 e a função horizontal retorna o valor 1 à função que a chamou, indicando que não foi encontrado "pixel-on" que pudesse ser varrido na direção desejada. Caso contrário, o valor da variável **"condição-de-folga"** é incrementado de uma unidade.

Procedimento 2

A variável **"condição-de-folga"** é igualada a 0 e a função horizontal retorna à função que a chamou, a coordenada do "pixel-on" vizinho ao "pixel" que está sendo varrido, encontrado nos procedimentos que antecedem o Procedimento 1. A coordenada do "pixel-on" vizinho é retornada como sendo a coordenada do próximo "pixel-on" a ser varrido. Se no PASSO 1 foi armazenada a coordenada de um ponto de cruzamento e o "pixel" que está sendo varrido não é o primeiro "pixel" a ser varrido pela função horizontal, a função ponto de cruzamento é executada, visando classificar o tipo de ponto de cruzamento formado pelo "pixel" que está sendo varrido e pelos "pixels-on" vizinhos a ele. A função ponto de cruzamento é executada com o objetivo de armazenar a coordenada do "pixel" que define o ponto de cruzamento, isto é, o "pixel" que está sendo varrido ou um "pixel-on" vizinho a ele.

Independente do Procedimento 1 ou 2 ter sido executado, o programa verifica se o final da linha horizontal não foi encontrado, foi encontrado ponto de cruzamento e o "pixel" que está sendo varrido não é o primeiro "pixel" a ser varrido pela função horizontal. Em caso afirmativo, a função horizontal retorna à função que a chamou, a quantidade de "pixels-on" vizinhos ao ponto de cruzamento e a coordenada do próximo "pixel" a ser varrido. Caso contrário, se o "pixel" que está sendo varrido é o primeiro "pixel" a ser varrido pela função horizontal, o programa retorna à função que a chamou, a coordenada do próximo "pixel" a ser varrido e o valor 0, indicando que o "pixel" que está sendo varrido possui apenas um "pixel-on" vizinho.

A figura 4.15 ilustra os "pixels-on" de 2 números manuscritos, varridos pela função horizontal.

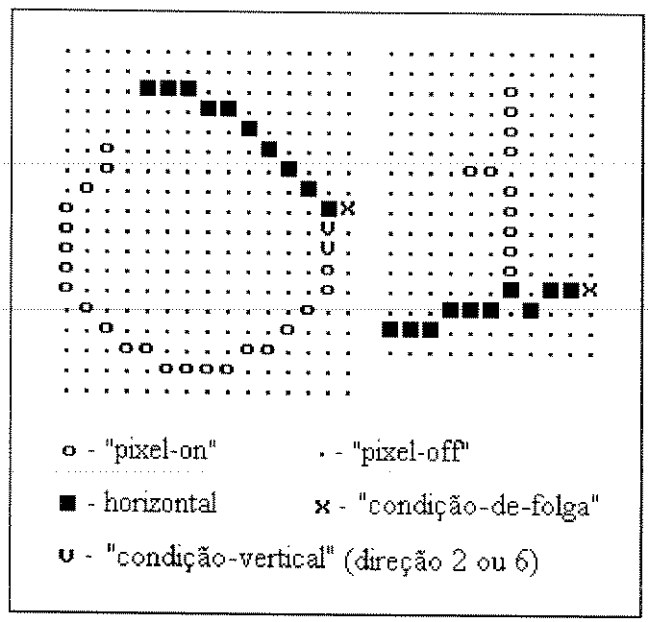


Figura 4.15 - Números manuscritos varridos pela função horizontal.

No capítulo 3, foi explicado que o texto a ser reconhecido é digitalizado e gravado. Após isto, o programa realiza a leitura do arquivo onde está gravado o texto digitalizado e em seguida são segmentadas as linhas desse texto. Neste trabalho foi considerado que o texto é composto de somente uma linha.

Para facilitar o entendimento de todo o processo utilizado no reconhecimento de caracteres numéricos manuscritos isolados, optou-se por explicá-lo no texto e esboçar o seu fluxograma. Aconselha-se ao leitor que para sua melhor clareza, leia o texto que descreve esse processo e acompanhe ao mesmo tempo o seu fluxograma. A figura 4.16 que se encontra nas páginas 434 a 439 do 2º volume deste trabalho, ilustra o fluxograma de todo o processo utilizado no reconhecimento de caracteres numéricos manuscritos isolados. Nesta figura, alguns comentários que se encontram no texto não constam do fluxograma. Devido à extensão do fluxograma ele foi dividido em 5 partes.

Após terminar de segmentar a linha de texto, o programa inicia a segmentação dos caracteres. A cada caractere segmentado, é executado o

processo de afilamento, com o objetivo de obter o seu esqueleto e o processo de reconhecimento, visando reconhecer o caractere segmentado. O processo de segmentação, afilamento e reconhecimento de cada caractere é repetido, até que não existam mais caracteres na linha de texto.

Para o reconhecimento de cada caractere, o programa encontra na matriz-original as coordenadas da linha horizontal superior (CILHS e CFLHS), as coordenadas da linha horizontal inferior (CILHI e CFLHI) e as coordenadas da linha vertical esquerda (CILVE e CFLVE). A matriz-original armazena os "pixels-on" e os "pixels-off" que constituem o esqueleto da imagem de um caractere. Após obter as coordenadas CILHS, CFLHS, CILHI, CFLHI, CILVE e CFLVE, o programa executa os seguintes procedimentos:

- Se ((o valor absoluto da diferença entre as abscissas ACFLHS e ACILHS é menor ou igual a 2) e (o valor absoluto da diferença entre as abscissas ACFLHI e ACILHI é diferente de 0)), é executada a função denominada no programa de verif-1-3-5-6-0. Na função verif-1-3-5-6-0, o programa delimita novamente as coordenadas inicial e final da linha horizontal superior, quando a matriz-original é varrida da direita para a esquerda, com o objetivo de certificar se esta linha realmente possui comprimento menor ou igual a 2 "pixels-on" (condição de entrada da função verif-1-3-5-6-0). Ao delimitar as coordenadas inicial e final da linha horizontal superior, quando a matriz-original é varrida da direita para a esquerda na função verif-1-3-5-6-0, se esta linha possuir comprimento maior do que a linha horizontal superior encontrada anteriormente à execução da função verif-1-3-5-6-0, esta linha será a nova linha horizontal superior do esqueleto da imagem do caractere. Devido à mudança das coordenadas da linha horizontal superior, o programa delimita novamente a coordenada final da linha vertical esquerda, pois sua coordenada

CILVE sempre coincide com a coordenada CILHS.

- Se o caractere não foi reconhecido na função verif-1-3-5-6-0 e se ((a abscissa ACILHI é diferente da abscissa ACFLHI) e (a abscissa ACILHS é diferente da abscissa ACFLHS) e ((o valor absoluto da diferença entre as ordenadas OCFLVE e OCILHI é menor ou igual a 2) e (o valor absoluto da diferença entre as abscissas ACFLVE e ACILHI é menor ou igual a 2)) ou o (valor absoluto da diferença entre as ordenadas OCFLVE e OCFLHI é menor ou igual a 2) e (o valor absoluto da diferença entre as abscissas ACFLVE e ACFLHI é menor ou igual a 2))), é executada a função denominada no programa de verif-0-8-6-3-5-1-2-4.

- Se o caractere não foi reconhecido na função verif-0-8-6-3-5-1-2-4 e se ((a abscissa ACILHS é diferente da abscissa ACFLHS) e (a abscissa ACILHI é diferente da abscissa ACFLHI)), é executada a função denominada no programa de verif-2-3-5-7-9-8-0.

- Se o caractere não foi reconhecido na função verif-2-3-5-7-9-8-0 e se ((a abscissa ACILHS é diferente da abscissa ACFLHS) e (o valor absoluto da diferença entre as abscissas ACFLHI e ACILHI é menor ou igual a 2)), é executada a função denominada no programa de verif-7-9-4-1-2-0-6-8-3.

- Se o caractere não foi reconhecido na função verif-7-9-4-1-2-0-6-8-3 e se ((o valor absoluto da diferença entre as abscissas ACFLHS e ACILHS é menor ou igual a 2) e (a abscissa ACILHI é diferente da abscissa ACFLHI)), o programa executa a função verif-1-3-5-6-0. Certos números, por exemplo, os números 1,

4 e 7, para o olho humano, apresentam as coordenadas CFLHI e CILHI coincidentes, porém, ao serem varridos por um "scanner", as coordenadas CFLHI e CILHI destes números, podem não coincidir. Com o objetivo de tornar o processo de reconhecimento do algoritmo desenvolvido neste trabalho o mais próximo possível do processo de reconhecimento utilizado pelo ser humano, o programa verifica se a ordenada OCFLHI é menor do que a ordenada OCILHI. Em caso afirmativo, a coordenada CFLHI é igualada a coordenada CILHI. Caso contrário, a coordenada CILHI é igualada a coordenada CFLHI. Com estas considerações, o programa executa o próximo procedimento.

- Se o caractere não foi reconhecido nas funções citadas acima e se ((o valor absoluto da diferença entre as abscissas ACFLHS e ACILHS é menor ou igual a 2) e (o valor absoluto da diferença entre as abscissas ACFLHI e ACILHI é menor ou igual a 2)), é executada a função denominada no programa de verif-1-4-5-6-7.

- Se o caractere não foi reconhecido nas funções verif-1-3-5-6-0, verif-0-8-6-3-5-1-2-4, verif-2-3-5-7-9-8-0, verif-7-9-4-1-2-0-6-8-3 e verif-1-4-5-6-7, o programa envia a mensagem que o número analisado não foi reconhecido.

A seguir são descritas as funções verif-1-3-5-6-0, verif-0-8-6-3-5-1-2-4, verif-2-3-5-7-9-8-0, verif-7-9-4-1-2-0-6-8-3 e verif-1-4-5-6-7:

Verif-1-3-5-6-0

O objetivo da função verif-1-3-5-6-0 é reconhecer os dígitos 1, 3, 5, 6 e 0.

Para facilitar o entendimento da função verif-1-3-5-6-0, optou-se por explicá-la no texto e esboçar o seu fluxograma. Aconselha-se ao leitor que para sua melhor clareza, leia o texto que descreve a função verif-1-3-5-6-0 e acompanhe ao mesmo tempo o seu fluxograma. A figura 4.17 que se encontra nas páginas 440 a 461 do 2º volume deste trabalho, ilustra o fluxograma da função verif-1-3-5-6-0. Nesta figura, alguns comentários que se encontram no texto não constam do fluxograma. Devido à extensão do fluxograma ele foi dividido em 21 partes.

O programa inicia a função verif-1-3-5-6-0 varrendo a matriz-original para a esquerda a partir da coordenada CFLHS, até que não existam mais "pixels-on" que possam ser varridos nessa direção. Enquanto a matriz-original é varrida para a esquerda, é armazenado o número de "pixels-on" que são varridos e o número de "pixels-on" varridos que se encontram na direção 5 ou 6. Se durante o processo em que a matriz-original é varrida para a esquerda, o "pixel-on" que está sendo varrido encontra-se na direção diagonal (direção 1, 3, 5 ou 7) e o "pixel-on" que acabou de ser varrido encontra-se na direção 6 (direção vertical), o programa considera a direção do "pixel-on" que está sendo varrido como sendo a direção 6 e o número de "pixels-on" varridos que se encontram na direção 5 ou 6 é incrementado de uma unidade. Devido a direção diagonal ser considerada como uma direção vertical, a variável **"condição-vertical"** é incrementada de uma unidade (este passo será referenciado adiante como PASSO 1). A matriz-original armazena os "pixels-on" e os "pixels-off" que constituem o esqueleto da imagem de um caractere.

Após terminar de varrer a matriz-original para a esquerda no PASSO 1, se o valor absoluto da diferença entre a coordenada do último "pixel-on" varrido nesse passo e a coordenada CFLHS é maior do que 2, o valor da coordenada CILHS é substituído pelo valor da coordenada do último "pixel-on" varrido no

PASSO 1. Esta consideração foi feita porque a linha horizontal superior possui comprimento maior do que o comprimento estabelecido inicialmente na função linha horizontal superior.

Se ((o número de "pixels-on" varridos no PASSO 1 é maior ou igual a 2) e (ocorreu mudança na coordenada CILHS como explicado no parágrafo anterior)), é verificado se o "pixel-on" dessa coordenada possui "pixels-on" vizinhos que se encontram na direção 6 ou 7. Em caso afirmativo, a matriz-original é varrida para baixo a partir do "pixel-on" cuja coordenada é a coordenada CILHS. A matriz-original é varrida para baixo, até que não existam mais "pixels-on" que possam ser varridos nessa direção ou seja encontrado um ponto de cruzamento. Enquanto a matriz-original é varrida para baixo, é armazenada a quantidade de "pixels-on" varridos que se encontram na direção 6 ou 7 (este passo será referenciado adiante como PASSO 2).

A partir da coordenada CILHI, o programa varre a matriz-original para a direita, até que não existam mais "pixels-on" que possam ser varridos nessa direção ou seja encontrado um ponto de cruzamento. Caso tenha sido encontrado um ponto de cruzamento, sua coordenada é armazenada (este passo será referenciado adiante como PASSO 3).

Se o valor absoluto da diferença entre as ordenadas OCFLHS e OCILHS é maior do que 5, o programa varre a matriz-original para baixo a partir da coordenada CFLHS, até que não existam mais "pixels-on" que possam ser varridos nessa direção. Se o valor absoluto da diferença entre as ordenadas OCFLHS e OCILHS é menor ou igual a 5, o programa varre a matriz-original para baixo a partir da coordenada CILHS, até que não existam mais "pixels-on" que possam ser varridos nessa direção. Enquanto a matriz-original é varrida para baixo, independente de partir da coordenada CFLHS ou CILHS, o programa anota a ordem em que ocorrem os conjuntos de "pixels-on" varridos que se

encontram nas direções 7 e 5 e, até que seja encontrada a coordenada CFLHI ou a coordenada do ponto de cruzamento encontrado no PASSO 3, é anotado o número de "pixels-on" varridos que se encontram nas direções 7 ou 5, o número de "pixels-on" varridos para baixo e o número de "pixels-on" varridos que se encontram na direção 6. Enquanto a matriz-original é varrida para baixo, se o "pixel-on" que está sendo varrido encontra-se na direção diagonal (direção 1, 3, 5 ou 7) e o "pixel-on" que acabou de ser varrido encontra-se na direção 6 (direção vertical), o programa considera a direção do "pixel-on" que está sendo varrido como sendo a direção 6 e o número de "pixels-on" varridos que se encontram na direção 6 é incrementado de uma unidade (este passo será referenciado adiante como PASSO 4).

Ao terminar de varrer a matriz-original para baixo, se (((a porcentagem de "pixels-on" varridos no PASSO 1 que se encontram nas direções 5 e 6 foi maior do que 60%) e (o número de "pixels-on" varridos no PASSO 2 que se encontram nas direções 6 e 7 foi menor do que 3)) ou ((não foram varridos conjuntos de "pixels-on" que se encontram nas direções 7, 5, 7 e 5 nesta ordem no PASSO 4) e (a coordenada CILHI não foi a primeira coordenada a ser encontrada no PASSO 2) e (a porcentagem de "pixels-on" varridos no PASSO 4 que se encontram na direção 6 foi maior ou igual a 70%))) (este passo será referenciado adiante como PASSO 5), então o programa segue os seguintes passos:

- A matriz-original é varrida para baixo a partir da coordenada CFLHS, até que não existam mais "pixels-on" que possam ser varridos nessa direção ou nas direções citadas a seguir (estas direções serão: para cima, para a esquerda ou para a direita). Enquanto a matriz-original é varrida para baixo, se é encontrado ponto de cruzamento, a variável **cruz** é incrementada de uma unidade e, se nesse ponto de cruzamento existirem "pixels-on" vizinhos que se encontram na

direção 1, 2 ou 3, o programa varre a matriz-original para cima, até que não existam mais "pixels-on" que possam ser varridos nessa direção. Enquanto a matriz-original é varrida para cima, é armazenada a quantidade de "pixels-on" varridos. Ao terminar de varrer a matriz-original para cima, o programa retorna ao ponto de cruzamento encontrado enquanto a matriz-original era varrida para baixo. Independente desse ponto de cruzamento possuir ou não "pixels-on" vizinhos que se encontram na direção 1, 2 ou 3, o programa continua a varrer a matriz-original para a esquerda, até que não existam mais "pixels-on" que possam ser varridos nessa direção. Enquanto a matriz-original é varrida para a esquerda, se ((o valor da variável **"condição-de-folga"** torna-se diferente de 0) e ((não é encontrado ponto de cruzamento) ou (é encontrado apenas um ponto de cruzamento))), o programa continua a varrer a matriz-original para a esquerda, até que não existam mais "pixels-on" que possam ser varridos nessa direção. Enquanto a matriz-original é varrida para baixo, para a esquerda ou para a direita, se são encontradas as coordenadas CILHI e/ou CFLHI, o programa anota a ordem em que estas coordenadas são encontradas (este passo será referenciado adiante como PASSO 6).

- O programa verifica se ((o último "pixel-on" varrido no PASSO 6 não possui "pixel-on" vizinho que se encontra na direção 3, 2 ou 1) ou (no PASSO 6 foi varrido pelo menos 3 "pixels-on" que se encontram na direção vertical, antes de encontrar um ponto de cruzamento que possui "pixels-on" vizinhos que se encontram na direção 1, 2 ou 3) e (o número de "pixels-on" varridos que se encontram na direção vertical no PASSO 6, antes do ponto de cruzamento encontrado foi maior do que 2 ou o número de "pixels-on" varridos que se encontram na direção horizontal, quando o programa varreu a matriz-original para a direita no PASSO 6 após o ponto de cruzamento encontrado era maior

do que 2) ou (as coordenadas CILHI e CFLHI foram encontradas)) (esta condição será referenciada adiante como CONDIÇÃO 1) então:

- O programa varre a matriz-original para cima, para a esquerda e/ou para a direita conforme os "pixels-on" estejam dispostos, a partir da coordenada CILHI. A matriz-original é varrida para cima, para a esquerda e/ou para a direita, até que não existam mais "pixels-on" que possam ser varridos nessas direções ou seja encontrado um ponto de cruzamento. Enquanto a matriz-original é varrida para cima, para a esquerda e/ou para a direita, é anotado se são varridos conjuntos de "pixels-on" que se encontram nas direções 0, 1, 2, 3 e 4, nesta ordem. Ao terminar de varrer a matriz-original para cima, para a esquerda e/ou para a direita, o programa verifica se o último "pixel-on" varrido nesse processo possui 2 ou mais "pixels-on" vizinhos. Em caso afirmativo, o programa anota que o último "pixel-on" varrido para cima, para a esquerda e/ou para a direita é um ponto de cruzamento (este passo será referenciado adiante como PASSO 7).

- Se no PASSO 7 não foram varridos conjuntos de "pixels-on" que se encontram nas direções 0, 1, 2, 3 e 4 nesta ordem e o último "pixel-on" varrido nesse passo foi um ponto de cruzamento, o programa varre a matriz-original para cima, para a esquerda e/ou para a direita conforme os "pixels-on" estejam dispostos, a partir do ponto de cruzamento encontrado no PASSO 7. A matriz-original é varrida para cima, para a esquerda e/ou para a direita, até que não existam mais "pixels-on" que possam ser varridos nessas direções. Enquanto a matriz-original é varrida para cima, para a esquerda e/ou para a

direita, o programa anota a cada "pixel-on" varrido, se o ponto de cruzamento encontrado no PASSO 7 é reencontrado (este passo será referenciado adiante como PASSO 8).

- No passo 8 foi verificado que alguns caracteres não apresentam conjuntos de "pixels-on" que se encontram nas direções 0, 1, 2, 3 e 4, nesta ordem. Para solucionar este problema, é verificado enquanto a matriz-original é varrida no PASSO 8, se não existe "pixel-on" que se encontra na direção 2 e existe "pixel-on" que se encontra na direção 3. Neste caso, houve uma imperfeição no caractere e para saná-la, o programa considera que existe um "pixel-on" que se encontra na direção 2. Ao varrer a matriz-original no PASSO 8, se ((não existe "pixel-on" que se encontra na direção 4 e existe "pixel-on" que se encontra na direção 5) ou (o "pixel-on" que está sendo varrido é um ponto de cruzamento e possui "pixel-on" vizinho que se encontra na direção 5)), o programa considera que existe um "pixel-on" que se encontra na direção 4, pelo mesmo motivo citado acima.

- Se ((no PASSO 7 não foram varridos conjuntos de "pixels-on" que se encontram nas direções 0, 1, 2, 3 e 4 nesta ordem e no PASSO 8 não foi reencontrado o ponto de cruzamento do PASSO 7) ou (o último "pixel-on" varrido no PASSO 7 não foi um ponto de cruzamento)) (este passo será referenciado adiante como PASSO 9), o programa verifica se foi anotado que o último "pixel-on" varrido no PASSO 7 é um ponto de cruzamento. Em caso afirmativo, a matriz-original é varrida para a esquerda a partir do ponto de cruzamento encontrado

no PASSO 7, até que não existam mais "pixels-on" que possam ser varridos nessa direção. Enquanto a matriz-original é varrida para a esquerda, o programa armazena o número de "pixels-on" varridos que se encontram na direção 3, 2 ou 1. Após terminar de varrer a matriz-original para a esquerda, é verificado se ((o último "pixel-on" varrido nesse processo possui "pixel-on" vizinho que se encontra na direção 3, 2 ou 1) e (o número de "pixels-on" varridos que se encontram nas direções 3, 2 e 1 armazenados enquanto a matriz-original era varrida para a esquerda é maior do que 2)) (esta condição será referenciada adiante como CONDIÇÃO 2)). Se o PASSO 9 não foi satisfeito, é verificado se o "pixel-on" de maior ordenada da matriz-original está acima de 45% do número total de linhas da matriz-imagem. Em caso afirmativo, o caractere é reconhecido como número 1. Caso contrário, o caractere é eliminado por ser considerado pelo programa como uma vírgula. No capítulo 5 deste trabalho, é explicado o motivo de eliminar a vírgula no programa. A matriz-imagem armazena os "pixels-on" e os "pixels-off" que constituem o esqueleto da imagem da linha de texto.

- Se no PASSO 7 foram varridos conjuntos de "pixels-on" que se encontram nas direções 0, 1, 2, 3 e 4 nesta ordem, o programa verifica se foi anotado que o último "pixel-on" varrido nesse passo é um ponto de cruzamento. Em caso afirmativo, a matriz-original é varrida para baixo a partir do ponto de cruzamento encontrado no PASSO 7, até que não existam mais "pixels-on" que possam ser varridos nessa direção ou o "pixel-on" que está sendo varrido está próximo da coordenada CILHI (diferença de 1 "pixel-on"). Se ao varrer a matriz-

original para baixo, o programa verifica que o "pixel-on" que está sendo varrido está próximo da coordenada CILHI, o caractere é reconhecido como número 6. Caso contrário, o programa retorna para o ponto de cruzamento encontrado no PASSO 7 ou se este ponto de cruzamento não foi encontrado, retorna ao último "pixel-on" varrido no PASSO 7 (este passo será referenciado adiante como PASSO 10).

- A partir do "pixel-on" encontrado no PASSO 10, a matriz-original é varrida para baixo e/ou para cima conforme os "pixels-on" estejam dispostos. A matriz-original é varrida para baixo e/ou para cima, até que não existam mais "pixels-on" que possam ser varridos nessas direções. Ao terminar de varrer a matriz-original para baixo e/ou para cima, se (((o valor absoluto da diferença entre a ordenada do último "pixel-on" varrido nesse processo e a ordenada OCILHS é menor ou igual a 3) e (o valor absoluto da diferença entre a abscissa desse último "pixel-on" varrido e a abscissa ACILHS é maior do que 2)) ou ((enquanto a matriz-original é varrida para baixo e/ou para cima é encontrado um "pixel-on" cujo valor absoluto da diferença entre a sua coordenada e a coordenada CILHS é menor ou igual a 1) e (também é varrido um "pixel-on" cujo valor absoluto da diferença entre a sua coordenada e a coordenada CILHI é menor ou igual a 1))), o caractere é reconhecido como 0.

- Se no PASSO 7 foram varridos conjuntos de "pixels-on" que se encontram nas direções 0, 1, 2, 3 e 4 nesta ordem, o programa varre a matriz-original para cima a partir do "pixel-on" encontrado no PASSO 10. A cada "pixel-on" varrido para cima, o programa verifica

se na linha do "pixel-on" varrido e em alguma das colunas à sua direita é encontrado algum "pixel-on". Em caso afirmativo, o "pixel-on" encontrado é denominado de *PN* . A matriz-original é varrida para cima, até que não existam mais "pixels-on" que possam ser varridos nessa direção ou seja encontrado o "pixel-on" *PN*. Se o "pixel-on" *PN* não é um ponto de cruzamento e não é um "pixel-on" que foi varrido quando a matriz-original era varrida para cima a partir do "pixel-on" encontrado no PASSO 10, o caractere é reconhecido como número 5. Caso contrário, o caractere é reconhecido como número 1. O procedimento de verificação do "pixel-on" *PN* foi realizado para reconhecer o número 5, pois seu traço superior encontra-se desconectado do restante do número.

- Se a CONDIÇÃO 2 ou PASSO 9 foi satisfeito ou a CONDIÇÃO 1 não foi satisfeita, o caractere não é reconhecido na função verif-1-3-5-6-0.

Se (((o PASSO 5 não foi satisfeito) e ((no PASSO 4 foram varridos conjuntos de "pixels-on" que se encontram nas direções 7, 5, 7 e 5 nesta ordem) e (a porcentagem de "pixels-on" varridos no PASSO 4 que se encontram na direção 6 é maior ou igual a 50% do número total de "pixels-on" varridos nesse passo))), o caractere é reconhecido como número 3. Caso contrário, o caractere não é reconhecido na função verif-1-3-5-6-0.

A figura 4.18 ilustra alguns números reconhecidos pela função verif-1-3-5-6-0.

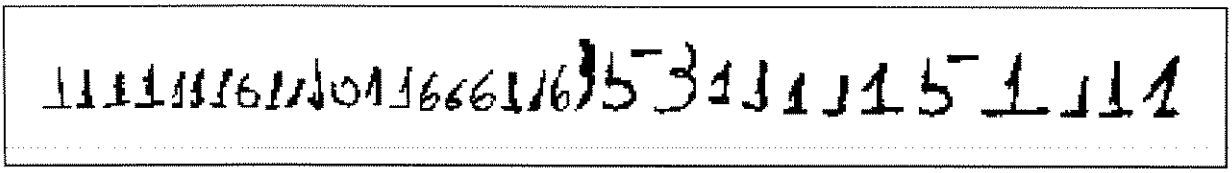


Figura 4.18 - Alguns números reconhecidos pela função verif-1-3-5-6-0.

Verif-0-8-6-3-5-1-2-4

O objetivo da função verif-0-8-6-3-5-1-2-4 é reconhecer os dígitos 0, 8, 6, 3, 5, 1, 2 e 4.

Para facilitar o entendimento da função verif-0-8-6-3-5-1-2-4, optou-se por explicá-la no texto e esboçar o seu fluxograma. Aconselha-se ao leitor que para sua melhor clareza, leia o texto que descreve a função verif-0-8-6-3-5-1-2-4 e acompanhe ao mesmo tempo o seu fluxograma. A figura 4.19 que se encontra nas páginas 462 a 495 do 2º volume deste trabalho, ilustra o fluxograma da função verif-0-8-6-3-5-1-2-4. Nesta figura, alguns comentários que se encontram no texto não constam do fluxograma. Devido à extensão do fluxograma ele foi dividido em 33 partes.

O programa inicia a função verif-0-8-6-3-5-1-2-4 verificando se o valor absoluto da diferença entre as coordenadas CILHI e CILHS é menor ou igual a 2. Em caso afirmativo, a matriz-original é varrida para baixo e/ou para a direita conforme os "pixels-on" estejam dispostos, a partir da coordenada CILHI. A matriz-original é varrida para baixo e/ou para a direita, até que não existam mais "pixels-on" que possam ser varridos nessas direções ou a coordenada CFLHS seja encontrada e possua "pixels-on" vizinhos que não foram varridos nesse processo. A partir da coordenada CFLHS a matriz-original é varrida para a esquerda, até que não existam mais "pixels-on" que possam ser varridos nessa direção. Enquanto a matriz-original é varrida para a esquerda, o programa

armazena a quantidade de "pixels-on" varridos nessa direção. Se esta quantidade é menor ou igual a 2 e o "pixel-on" de maior ordenada da matriz-original está acima de 40% do número total de linhas da matriz-imagem, o caractere é reconhecido como número 1. Caso contrário, o caractere é eliminado por ser considerado pelo programa como uma vírgula. No item 5.2 do capítulo 5 deste trabalho é explicado o motivo da eliminação da vírgula no programa. A matriz-original armazena os "pixels-on" e os "pixels-off" que constituem o esqueleto da imagem de um caractere e a matriz-imagem armazena os "pixels-on" e os "pixels-off" que constituem o esqueleto da imagem da linha de texto.

Se o valor absoluto da diferença entre as coordenadas CILHI e CILHS é maior do que 2, a matriz-original é varrida para a direita, a partir da coordenada CILHI. A matriz-original é varrida para a direita, até que não existam mais "pixels-on" que possam ser varridos nessa direção, seja encontrado um ponto de cruzamento ou a coordenada CFLHI. Enquanto a matriz-original é varrida para a direita, se é encontrado um ponto de cruzamento, essa matriz é varrida para baixo, a partir desse ponto de cruzamento. A matriz-original é varrida para a baixo, até que não existam mais "pixels-on" que possam ser varridos nessa direção ou seja encontrada a coordenada CFLHI. Enquanto a matriz-original é varrida para baixo ou para a direita, se é encontrada a coordenada CFLHI; o programa anota se esta coordenada possui "pixel-on" vizinho que se encontra na direção 3, 2 ou 1 (este passo será referenciado adiante como PASSO 1) e a partir dessa coordenada a matriz-original é varrida para cima e/ou para a esquerda conforme os "pixels-on" estejam dispostos. A matriz-original é varrida para cima e/ou para a esquerda, até que não existam mais "pixels-on" que possam ser varridos nessas direções ou as coordenadas CILHS ou CFLHS sejam encontradas (este passo será referenciado adiante como PASSO 2).

Enquanto a matriz-original é varrida para cima e/ou para a esquerda no

PASSO 2, se é encontrado um ponto de cruzamento, o programa denomina-o de *PC* e verifica se ((o número de "pixels-on" vizinhos do ponto de cruzamento *PC* é igual a 2 e estes "pixels-on" encontram-se nas direções (3 e 2) ou (2 e 1) ou (3 e 4) ou (4 e 5)) ou ((o número de "pixels-on" varridos no PASSO 2 é menor ou igual a 6) e (o número de "pixels-on" vizinhos do ponto de cruzamento *PC* é maior ou igual a 2) e (o ponto de cruzamento *PC* coincide com o ponto de cruzamento encontrado no PASSO 1))). Em caso afirmativo, se no PASSO 1 o programa anotou que a coordenada CFLHI não possui "pixel-on" vizinho que se encontra na direção 3, 2 ou 1, é verificado se o ponto de cruzamento *PC* possui "pixel-on" vizinho que se encontra na direção 3, 2 ou 1. Em caso afirmativo, o programa continua a varrer a matriz-original para cima e/ou para a esquerda a partir do "pixel-on" vizinho que se encontra na direção 3, 2 ou 1. Caso contrário, a matriz-original é varrida para cima e/ou para a esquerda a partir do ponto de cruzamento *PC*.

Se no PASSO 2 foi encontrado o ponto de cruzamento *PC*, que possui "pixels-on" vizinhos que se encontram nas direções ((4, 3, 2 ou 5) e (2, 1 ou 0)) (este passo será referenciado adiante como PASSO 3), o programa anota se o ponto de cruzamento *PC* possui "pixels-on" vizinhos que se encontram nas direções 1, 3, 5 e (7 ou 6) e varre a matriz-original para a esquerda e/ou para baixo conforme os "pixels-on" estejam dispostos, a partir do ponto de cruzamento *PC* para a esquerda e/ou para baixo. A matriz-original é varrida para a esquerda e/ou para baixo, até que não existam mais "pixels-on" que possam ser varridos nessas direções ou passe próximo da coordenada CILHI, isto é, seja varrido um "pixel-on" que se encontra a uma distância de até 2 "pixels-on" dessa coordenada (este passo será referenciado adiante como PASSO 4).

Independente da direção em que a matriz-original é varrida, se ((é encontrado um ponto de cruzamento que se encontra a uma distância de até 2

"pixels-on" de outro ponto de cruzamento encontrado anteriormente) ou (os "pixels-on" que formam o ponto de cruzamento estão em direção oposta à direção que a matriz-original é varrida, o programa não considera este ponto de cruzamento encontrado como um ponto de cruzamento. Um exemplo ilustrativo disto, seria a matriz-original estar sendo varrida para a esquerda (direção 3, 4 ou 5) e o ponto de cruzamento ser formado por "pixels-on" que se encontram nas direções 4 e (1 ou 7) em relação ao "pixel-on" que está sendo varrido.

Enquanto a matriz-original é varrida para a esquerda e/ou para baixo no PASSO 4, se é encontrado um ponto de cruzamento, o programa anota sua coordenada e se esse ponto de cruzamento possui "pixel-on" vizinho que se encontra na direção 1, 3 ou 5. Caso esse ponto de cruzamento encontrado coincida com o ponto de cruzamento *PC*, o ponto de cruzamento encontrado é desconsiderado. Entretanto, se a matriz-original é varrida para a esquerda e/ou para baixo no PASSO 4 e é encontrado um ponto de cruzamento que é desconsiderado pelos motivos explicados no parágrafo acima, o programa anota se o último "pixel-on" varrido antes de ser encontrado o ponto de cruzamento desconsiderado possui "pixel-on" vizinho que se encontra na direção 1, 3 ou 5. Enquanto a matriz-original é varrida para a esquerda e/ou para baixo, se é encontrado um "pixel-on" que possui "pixels-on" vizinhos que se encontram nas direções 3 e 5, o programa continua a varrer a matriz-original para a esquerda e/ou para baixo, a partir do "pixel-on" vizinho que se encontra na direção 5. Como são anotados somente os "pixels-on" que se encontram nas direções 1, 3 e 5, enquanto a matriz-original é varrida para a esquerda e/ou para baixo, se não existir "pixel-on" que se encontra na direção 3 e existir "pixel-on" que se encontra na direção 2, o programa considera que houve uma imperfeição no caractere e para saná-la, considera que existe "pixel-on" que se encontra na direção 3. Se ao varrer a matriz-original para a esquerda e/ou para baixo, existir

"pixel-on" que se encontra na direção 6 e não existir "pixel-on" que se encontra na direção 5, o programa considera que houve uma imperfeição no caractere e para saná-la, considera que existe "pixel-on" que se encontra na direção 5. Quando não existirem mais "pixels-on" que possam ser varridos para a esquerda e/ou para baixo no PASSO 4, o programa verifica se ainda existem pontos de cruzamento cujas coordenadas foram armazenadas e se não foi varrido um "pixel-on" que se encontra próximo (diferença menor ou igual a 2 "pixels-on") da coordenada CILHI. Em caso afirmativo, o programa continua a varrer a matriz-original para a esquerda e/ou para baixo a partir de um dos pontos de cruzamento cujas coordenadas foram armazenadas, desde que o ponto de cruzamento não coincida com o ponto de cruzamento *PC* referido no PASSO 2. O processo de varrer a matriz-original para a esquerda e/ou para baixo, a partir do ponto de cruzamento cuja coordenada foi armazenada se repete, até que não existam mais pontos de cruzamento cujas coordenadas foram armazenadas ou tenha sido varrido algum "pixel-on" cujo valor absoluto da diferença entre a sua coordenada e a coordenada CILHI é menor ou igual a 2. Se foi varrido um "pixel-on" que se encontra próximo da coordenada CILHI, enquanto a matriz-original é varrida para a esquerda e/ou para baixo, programa segue os seguintes passos:

- A matriz-original é varrida para a direita a partir da coordenada CILHI. A matriz-original é varrida para a direita, até que não existam mais "pixels-on" que possam ser varridos nessa direção ou o valor absoluto da diferença entre a coordenada do "pixel-on" que está sendo varrido e a coordenada do ponto de cruzamento *PC* encontrado no PASSO 3 seja menor ou igual a 1. Enquanto a matriz-original é varrida para a direita, se é encontrado um ponto de cruzamento, o programa verifica se este ponto de cruzamento possui "pixel-on" vizinho que se encontra na direção 6, 7, 0 ou 1, nesta ordem. Em caso

afirmativo, o programa varre a matriz-original para a direita, a partir do "pixel-on" vizinho do ponto de cruzamento encontrado. A matriz-original é varrida para a direita, até que não existam mais "pixels-on" que possam ser varridos nessa direção ou o valor absoluto da diferença entre a coordenada do "pixel-on" que está sendo varrido e a coordenada do ponto de cruzamento *PC* encontrado no PASSO 3 seja menor ou igual a 1 (este passo será referenciado adiante como PASSO 5).

- Se o valor absoluto da diferença entre a coordenada do "pixel-on" que está sendo varrido no PASSO 5 e a coordenada do ponto de cruzamento *PC* encontrado no PASSO 3 é maior do que 1, o programa retorna ao ponto de cruzamento *PC* e realiza os seguintes procedimentos:

  - Se o ponto de cruzamento *PC* encontrado no PASSO 3 possui "pixels-on" vizinhos que se encontram nas direções 2 e (1 ou 3), a matriz-original é varrida para cima e/ou para a direita conforme os "pixels-on" estejam dispostos, a partir do "pixel-on" vizinho que se encontra na direção 2. A matriz-original é varrida para cima e/ou para a direita, até que não existam mais "pixels-on" que possam ser varridos nessas direções. Caso contrário, a matriz-original é varrida para cima e/ou para a direita conforme os "pixels-on" estejam dispostos, a partir do ponto de cruzamento *PC* encontrado no PASSO 3. A matriz-original é varrida para cima e/ou para a direita, até que não existam mais "pixels-on" que possam ser varridos nessas direções.

  - Enquanto a matriz-original é varrida para cima e/ou para a direita, é

anotado se passa próximo da coordenada CILHS (valor absoluto da diferença entre a ordenada do "pixel-on" que está sendo varrido e a ordenada OCILHS é menor ou igual a 1 e o valor absoluto da diferença entre a abscissa do "pixel-on" que está sendo varrido e a abscissa ACILHS é menor ou igual a 2), se é encontrada a coordenada CFLHS e a ordem em que são encontrados os conjuntos de "pixels-on" varridos para cima e/ou para a direita que se encontram nas direções 1 e 3. Ao varrer a matriz-original para cima e/ou para a direita, se as coordenadas CILHS e CFLHS foram encontradas, o programa anota a ordem em que estas coordenadas foram encontradas (este passo será referenciado adiante como PASSO 6). Enquanto a matriz-original é varrida para cima e/ou para a direita no PASSO 6, se é um encontrado ponto de cruzamento, o programa varre a matriz-original para a direita e/ou para cima conforme os "pixels-on" estejam dispostos, a partir desse ponto de cruzamento encontrado. A matriz-original é varrida para a direita e/ou para cima, até que não existam mais "pixels-on" que possam ser varridos nessas direções. Enquanto a matriz-original é varrida para cima e/ou para a direita, se é encontrado um ponto de cruzamento, o programa anota se este ponto de cruzamento possui "pixels-on" vizinhos que se encontram nas direções 1, 3, 5 ou 7. Ao terminar de varrer a matriz-original para cima e/ou para a direita, o programa armazena na variável **L1** a ordenada do último "pixel-on" varrido nesse processo.

- Ao varrer a matriz-original para cima e/ou para a direita no PASSO 6, se o programa encontrou as coordenadas CILHS e CFLHS nesta ordem, é verificado se no PASSO 6 foram varridos conjuntos de

"pixels-on" que se encontram nas direções 1 e 3, nesta ordem. Em caso afirmativo, o caractere é reconhecido como número 2. Caso contrário, o caractere é reconhecido como número 6. Se as coordenadas CILHS e CFLHS não foram encontradas nesta ordem, apenas uma ou nenhuma delas foi encontrada, o programa retorna ao ponto de cruzameno *PC* encontrado no PASSO 3 e varre a matriz-original para baixo, até que não existam mais "pixels-on" que possam ser varridos nessa direção ou a coordenada CILHS seja encontrada. Enquanto a matriz-original é varrida para baixo a partir do ponto de cruzamento *PC* encontrado no PASSO 3, se a coordenada CILHS é encontrada, o caractere é reconhecido como número 6. Se a coordenada CILHS não é encontrada, enquanto a matriz-original é varrida para baixo, o programa verifica se ((os pontos de cruzamento encontrados nos PASSOS 4 e 6 possuem "pixels-on" vizinhos que se encontram nas direções 1, 5 e 7) e (no PASSO 6 foi encontrada a coordenada CFLHS) ou ((o valor absoluto da diferença entre a ordenada OCFLHS e a ordenada armazenada na variável **L1** é menor ou igual a 5) e (o valor absoluto da diferença entre a ordenada OCFLHI e a ordenada armazenada na variável **L1** é maior ou igual a 4))). Em caso afirmativo, o caractere é reconhecido como número 8. Caso contrário, o caractere é reconhecido como número 0.

- Se o programa ao varrer a matriz-original para a direita no PASSO 5 encontrou um "pixel-on" cujo valor absoluto da diferença entre a sua coordenada e a coordenada do ponto de cruzamento *PC* encontrado no PASSO 3 é menor ou igual a 1, o programa realiza os seguintes procedimentos:

- A matriz-original é varrida para baixo e/ou para a esquerda conforme os "pixels-on" estejam dispostos, a partir da coordenada CFLHS. A matriz-original é varrida para baixo e/ou para a esquerda, até que não existam mais "pixels-on" que possam ser varridos nessas direções ou seja encontrada a coordenada CILHS.

- Se a coordenada CILHS foi encontrada no item anterior, o caractere é reconhecido como número 6. Se a coordenada CILHS não foi encontrada no item anterior, a matriz-original é varrida para cima a partir do ponto de cruzamento *PC* encontrado no PASSO 3. A matriz-original é varrida para cima, até que não existam mais "pixels-on" que possam ser varridos nessa direção ou seja encontrado um "pixel-on" cujo valor absoluto da diferença entre a sua coordenada e a coordenada CILHS é menor ou igual a 1. Enquanto a matriz-original é varrida para cima, se é encontrado um ponto de cruzamento que possui "pixel-on" vizinho que se encontra na direção 4, 3 ou 2, o programa continua a varrer a matriz-original para cima, a partir desse ponto de cruzamento. A matriz-original é varrida para cima, até que não existam mais "pixels-on" que possam ser varridos nessa direção ou seja encontrado um "pixel-on" cujo valor absoluto da diferença entre a sua coordenada e a coordenada CILHS seja menor ou igual a 1. Enquanto a matriz-original é varrida para cima, se é encontrado um "pixel-on" cujo valor absoluto da diferença entre a sua coordenada e a coordenada CILHS é menor ou igual a 1, o caractere é reconhecido como número 8. Caso contrário, o caractere é reconhecido com número 2.

Se no PASSO 4 não passou próximo da coordenada CILHI, o programa retorna ao ponto de cruzamento *PC* encontrado no PASSO 3 e varre a matriz-original para cima, para a esquerda ou para baixo conforme os "pixels-on" estejam dispostos. A matriz-original é varrida para cima, para a esquerda ou para baixo, até que não existam mais "pixels-on" que possam ser varridos em nenhuma dessas direções. Enquanto a matriz-original é varrida para cima, para a esquerda ou para baixo, o programa verifica se passa próximo (diferença de 1 "pixel-on") das coordenadas CILHS, CFLHS e CILHI e anota a ordem em que as 2 primeiras coordenadas são encontradas. Ao terminar de varrer a matriz-original para cima, para a esquerda ou para baixo, se não foi encontrada a coordenada CILHS ou CFLHS neste processo, o programa retorna ao ponto de cruzamento *PC* encontrado no PASSO 3 e continua a varrer a matriz-original para cima, para a esquerda ou para baixo. Ao terminar de varrer a matriz-original para cima, para a esquerda ou para baixo, se as coordenadas CFLHS e CILHS foram encontradas nesta ordem, o programa verifica se a coordenada CILHI foi encontrada. Em caso afirmativo, o caractere é reconhecido como número 0. Caso contrário, o caractere é reconhecido como número 3. Ao terminar de varrer a matriz-original para cima, para a esquerda ou para baixo, se as coordenadas CFLHS e CILHS não foram encontradas, apenas uma delas foi encontrada ou as duas coordenadas foram encontradas mas não nesta ordem, o programa verifica se a coordenada CILHI foi encontrada. Em caso afirmativo, o caractere é reconhecido como número 0. Caso contrário, o caractere é reconhecido como número 5.

Se no PASSO 2 foi encontrado o ponto de cruzamento *PC* que não possui "pixels-on" vizinhos que se encontram nas direções (4, 3, 2 ou 5) e (2, 1 ou 0), o programa varre a matriz-original para cima e/ou para a esquerda conforme os "pixels-on" estejam dispostos, a partir do ponto de cruzamento *PC*. A matriz-original é varrida para cima e/ou para a esquerda, até que não existam mais

"pixels-on" que possam ser varridos nessas direções. Enquanto a matriz-original é varrida para cima e/ou para a esquerda, o programa verifica se passa próximo (diferença de 2 "pixels-on") da coordenada CFLHS e se é encontrada a coordenada CILHS. Em caso afirmativo, o caractere é reconhecido como número 0. Caso contrário, o caractere é reconhecido como número 6.

Ao terminar de varrer a matriz-original no PASSO 2, independente de ter ou não encontrado um ponto de cruzamento, a coordenada do último "pixel-on" varrido nesse passo é armazenada como sendo a coordenada final da linha vertical direita (CFLVD). A coordenada CFLVD (ACFLVD, OCFLVD) consiste: de uma abscissa denominada de abscissa da coordenada final da linha vertical direita (ACFLVD) e de uma ordenada denominada de ordenada da coordenada final da linha vertical direita (OCFLVD). O programa considera que os "pixels-on" que formam o lado direito do caractere constituem a linha vertical direita do caractere.

Se enquanto a matriz-original era varrida para cima e/ou para a esquerda no PASSO 2, o programa verifica se ((foi varrido algum "pixel-on" cujo valor absoluto da diferença entre a sua coordenada e a coordenada CFLHS é menor ou igual a 1) e (a coordenada CFLVD coincide com a coordenada CILHS)). Em caso afirmativo, a variável **indi** é igualada a 0. Caso contrário, a variável **indi** é igualada a 1.

Após o programa executar os 6 passos descritos anteriormente, a matriz-original é varrida para baixo a partir da coordenada CILHS, até que não existam mais "pixels-on" que possam ser varridos nessa direção ou o valor absoluto da diferença entre a coordenada do "pixel-on" varrido e a coordenada CFLVD seja menor do que 2 e o valor da variável **indi** seja igual a 1 (este passo será referenciado adiante como PASSO 7).

Após terminar de varrer a matriz-original para baixo no PASSO 7, o

programa verifica se ((o valor absoluto da diferença entre a coordenada do último "pixel-on" varrido nesse passo e a coordenada CFLVD é menor do que 2) e (o valor da variável **indi** é igual a 1)). Em caso afirmativo, os seguintes procedimentos são realizados:

- A matriz-original é varrida para a direita ou para a esquerda conforme os "pixels-on" estejam dispostos, a partir da coordenada CILHS, até que não existam mais "pixels-on" que possam ser varridos em nenhuma dessas direções. Enquanto a matriz-original é varrida para a direita ou para a esquerda, o programa anota a ordem em que as coordenadas CFLHS e CFLHI são encontradas.

- Se ao varrer a matriz-original para a direita ou para a esquerda a primeira coordenada a ser encontrada foi a coordenada CFLHS, o programa verifica se foi encontrada a coordenada CFLHI. Em caso afirmativo, se foi encontrado um ponto de cruzamento enquanto a matriz-original era varrida para a direita ou para a esquerda, o programa retorna a esse ponto de cruzamento e varre a matriz-original para a esquerda. A matriz-original é varrida para a esquerda, até que não existam mais "pixels-on" que possam ser varridos nessa direção ou seja encontrado um "pixel-on" cujo valor absoluto da diferença entre a sua coordenada e a coordenada CILHS seja menor ou igual a 2. Se ao varrer a matriz-original para a direita ou para a esquerda não foi encontrado ponto de cruzamento ou ao varrer essa matriz apenas para a esquerda não foi varrido um "pixel-on" cujo valor absoluto da diferença entre a sua coordenada e a coordenada CILHS é menor ou igual a 2, o caractere é reconhecido como número 3. Caso contrário, o caractere é reconhecido como número 0. Se a coordenada CFLHI não foi encontrada enquanto a matriz-original era varrida

para a direita ou para a esquerda, o programa varre a matriz-original para cima a partir dessa coordenada. A matriz-original é varrida para cima, até que não existam mais "pixels-on" que possam ser varridos nessa direção. Enquanto a matriz-original é varrida para cima, o programa anota a quantidade de "pixels-on" varridos que se encontram na direção vertical (direção 1, 2 ou 3), se são encontradas as coordenadas CFLHS e CILHS e a ordem em que estas coordenadas são encontradas. Ao terminar de varrer a matriz-original para cima, se houver encontrado apenas uma das coordenadas (CFLHS ou CILHS), o programa verifica se foram armazenadas coordenadas de pontos de cruzamento que possuem "pixels-on" vizinhos que possam ser varridos. Em caso afirmativo, o programa continua a varrer a matriz-original para cima a partir de um desses pontos de cruzamento cujas coordenadas foram armazenadas. Caso contrário, o processo de varrer a matriz-original para cima é finalizado. Ao terminar de varrer a matriz-original para cima, se o número de "pixels-on" varridos que se encontram na direção vertical é menor ou igual a 2, o caractere é reconhecido como número 0. Caso contrário, se as coordenadas CFLHS e CILHS foram encontradas nesta ordem enquanto a matriz-original era varrida para cima, o caractere é reconhecido como número 2. Se as coordenadas CFLHS e CILHS não foram encontradas nesta ordem, o caractere é reconhecido como número 5.

- Se ao varrer a matriz-original para a direita ou para a esquerda, a primeira coordenada a ser encontrada não foi a coordenada CFLHS, o caractere não é reconhecido na função verif-0-8-6-3-5-1-2-4.

Se após terminar de varrer a matriz-original para baixo no PASSO 7, o programa verificar que ((o valor absoluto da diferença entre a coordenada do

último "pixel-on" varrido nesse passo e a coordenada CFLVD é maior ou igual a 2) ou (o valor da variável **indi** é diferente de 1)), os seguintes procedimentos são realizados:

- A matriz-original é varrida para cima a partir da coordenada CFLHI, até que não existam mais "pixels-on" que possam ser varridos nessa direção ou a coordenada CFLHS seja encontrada.

- Se ao varrer a matriz-original para cima a coordenada CFLHS não foi encontrada, o programa varre a matriz-original para a esquerda a partir do último "pixel-on" varrido quando essa matriz foi varrida para cima. A matriz-original é varrida para a esquerda, até que não existam mais "pixels-on" que possam ser varridos nessa direção ou o valor absoluto da diferença entre a coordenada do último "pixel-on" varrido nesse processo e a coordenada CFLHI seja menor ou igual a 1. Enquanto a matriz-original é varrida para a esquerda, se ((o "pixel-on" que está sendo varrido encontra-se na direção 5 ou 7) e (o "pixel-on" que acabou de ser varrido encontra-se na direção 6)), o programa considera que o "pixel-on" que está sendo varrido encontra-se na direção 6. Ao varrer a matriz-original para a esquerda, se ((o "pixel-on" que está sendo varrido encontra-se em uma direção diferente de 5 e 7) ou (o último "pixel-on" varrido encontra-se em uma direção diferente de 6), o "pixel-on" que está sendo varrido é considerado como estando na direção horizontal (direção 4) (este passo será referenciado adiante como PASSO 8).

- O programa varre a matriz-original para a esquerda, para baixo e/ou para a direita conforme os "pixels-on" estejam dispostos, a partir da coordenada CFLHS. A matriz-original é varrida para a esquerda, para baixo e/ou para a

direita, até que não existam mais "pixels-on" que possam ser varridos nessas direções. Enquanto a matriz-original é varrida para a esquerda, para baixo e/ou para a direita, o programa anota se são encontradas as coordenadas CILHS, CILHI e CFLHI, nesta ordem.

- Se no PASSO 8 o número de "pixels-on" varridos para a esquerda é menor ou igual a 2, o programa verifica se ((a coordenada CFLVD coincide com a coordenada CFLHS ou CILHS) ou (as coordenadas CILHS, CILHI e CFLHI foram encontradas nesta ordem ao varrer a matriz-original para a esquerda, para baixo e/ou para a direita no item anterior)). Em caso afirmativo, o caractere é reconhecido como número 0. Caso contrário, o caractere é reconhecido como número 4.

- Se no PASSO 8 o número de "pixels-on" varridos para a esquerda é maior do que 2, o caractere é reconhecido como número 6.

A figura 4.20 ilustra alguns números reconhecidos pela função verif-0-8-6-3-5-1-2-4.

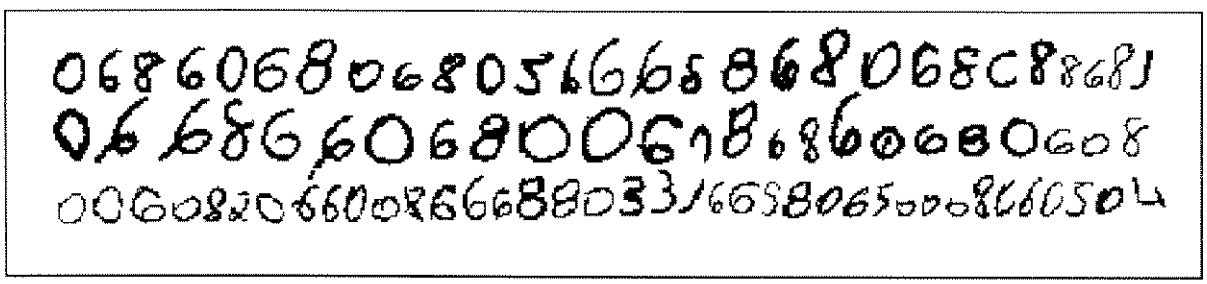


Figura 4.20 - Alguns números reconhecidos pela função verif-0-8-6-3-5-1-2-4.

Verif-2-3-5-7-9-8-0

O objetivo da função verif-2-3-5-7-9-8-0 é reconhecer o dígitos 2, 3, 5, 7, 9, 8 e 0.

Para facilitar o entendimento da função verif-2-3-5-7-9-8-0, optou-se por explicá-la no texto e esboçar o seu fluxograma. Aconselha-se ao leitor que para sua melhor clareza, leia o texto que descreve a função verif-2-3-5-7-9-8-0 e acompanhe ao mesmo tempo o seu fluxograma. A figura 4.21 que se encontra nas páginas 496 a 537 do 2º volume deste trabalho, ilustra o fluxograma da função verif-2-3-5-7-9-8-0. Nesta figura, alguns comentários que se encontram no texto não constam do fluxograma. Devido à extensão do fluxograma ele foi dividido em 41 partes.

O programa inicia a função verif-2-3-5-7-9-8-0 varrendo a matriz-original para cima, a partir da coordenada CFLHI. A matriz-original é varrida para cima, até que não existam mais "pixels-on" que possam ser varridos nessa direção, a quantidade de "pixels-on" varridos que se encontram na direção vertical seja maior do que 2, seja encontrado um ponto de cruzamento ou o valor da variável **"condição-de-folga"** seja diferente de 0. Enquanto a matriz-original é varrida para cima, é verificado para cada "pixel-on" que está sendo varrido se antes do "pixel-on" ser varrido o valor da variável **"condição-horizontal"** era diferente de 0 e após o "pixel-on" ser varrido o valor desta variável é igual a 0. Se isto ocorrer, o programa verifica se o "pixel-on" que está sendo varrido encontra-se em uma direção diferente da direção 2 ou 6. Em caso afirmativo, o programa armazena na variável **"condição-horizontal"** o valor que esta variável possuía antes do "pixel-on" ser varrido incrementado de uma unidade. Se ((os valores das variáveis **"condição-de-folga"** e **"condição-horizontal"** são iguais a 0) ou (o valor da variável **"condição-de-folga"** é igual a 0) e (o número de "pixels-on"

varridos que se encontram na direção vertical é maior do que 1) e (o "pixel-on" que está sendo varrido encontra-se na direção 0, 2, 4 ou 6)), o "pixel-on" que está sendo varrido é considerado como estando na direção vertical (este passo será referenciado adiante como PASSO 1). A matriz-original armazena os "pixels-on" e os "pixels-off" que constituem o esqueleto da imagem de um caractere.

O programa retorna à coordenada CFLHI e se a quantidade de "pixels-on" varridos que se encontram na direção vertical no PASSO 1 é maior do que 2, a matriz-original é varrida para cima e/ou para a esquerda conforme os "pixels-on" estejam dispostos, a partir da coordenada CFLHI. A matriz-original é varrida para cima e/ou para a esquerda, até que não existam mais "pixels-on" que possam ser varridos nessas direções ou seja encontrado um ponto de cruzamento (este passo será referenciado adiante como PASSO 2).

Enquanto a matriz-original é varrida para cima e/ou para a esquerda no PASSO 2, se é encontrado um ponto de cruzamento que possui apenas 2 "pixels-on" vizinhos adjacentes, o programa desconsidera este ponto de cruzamento e continua a varrer a matriz-original para cima e/ou para a esquerda. Se o ponto de cruzamento não foi desconsiderado, o programa verifica se o ponto de cruzamento possui "pixel-on" vizinho que se encontra na direção 4, 3 ou 5. Em caso afirmativo, a matriz-original é varrida na direção 3, 4 ou 5 a partir do ponto de cruzamento, até que não existam mais "pixels-on" que possam ser varridos em nenhuma dessas direções ou o valor absoluto da diferença entre a ordenada do "pixel-on" que está sendo varrido e a ordenada OCILHS seja menor ou igual a 1 e o valor absoluto da diferença entre a abscissa do "pixel-on" que está sendo varrido e a abscissa ACILHS seja menor ou igual a 2. Enquanto a matriz-original é varrida na direção 3, 4 ou 5, se são encontrados "pixels-on" que se encontram nas direções 4 e 3, o programa continua a varrer a matriz-original na direção 4. Ao varrer a matriz-original na direção 3, 4 ou 5, se são encontrados "pixels-on"

que se encontram nas direções 5 e 3, o programa continua a varrer a matriz-original na direção 5. Enquanto a matriz-original é varrida na direção 3, 4 ou 5, se é encontrado ponto de cruzamento, o programa armazena a coordenada deste ponto de cruzamento (este passo será referenciado adiante como PASSO 3).

Enquanto a matriz-original é varrida na direção 3, 4 ou 5 no PASSO 3, se ((o valor absoluto da diferença entre a ordenada do "pixel-on" que está sendo varrido e a ordenada OCILHS é menor ou igual a 1) e (o valor absoluto da diferença entre a abscissa do "pixel-on" que está sendo varrido e a abscissa ACILHS é menor ou igual a 2) e (o último "pixel-on" varrido nesse passo não possui "pixel-on" vizinho que se encontra na direção 1, 2 ou 3)) (este passo será referenciado adiante como PASSO 4), são realizados os seguintes procedimentos:

- O programa verifica se o último "pixel-on" varrido no PASSO 3 possui "pixel-on" vizinho que se encontra na direção 3, 4 ou 5. Em caso afirmativo, a coordenada do "pixel-on" vizinho que se encontra na direção 3, 4 ou 5 é armazenada. Caso contrário, a coordenada do último "pixel-on" varrido no PASSO 3 é armazenada (este passo será referenciado adiante como PASSO 5).

- Se o último "pixel-on" varrido no PASSO 3 possui "pixel-on" vizinho que se encontra na direção 5, 6 ou 7, o programa realiza os seguintes procedimentos:

  - O programa retorna ao ponto de cruzamento encontrado no PASSO 2 e varre a matriz-original para cima e/ou para a esquerda, conforme os "pixels-on" estejam dispostos. A matriz-original é varrida para cima e/ou para a esquerda, até que não existam mais "pixels-on" que possam ser varridos nessas direções ou seja encontrado um "pixel-on" cuja ordenada é igual a ordenada OCILHS e abscissa é diferente da

abscissa ACILHS. Enquanto a matriz-original é varrida para cima e/ou para a esquerda, se ((é encontrado ponto de cruzamento, que será denominado no programa de *PC1*) e (a coordenada CFLHS não é encontrada)), o programa verifica se no PASSO 3 não foi encontrado ponto de cruzamento. Em caso afirmativo, a coordenada do ponto de cruzamento *PC1* é armazenada (este passo será referenciado adiante como PASSO 6). Após isto, é verificado se o ponto de cruzamento *PC1* possui "pixel-on" vizinho que se encontra à sua direita. Em caso afirmativo, o programa continua a varrer a matriz-original para cima e/ou para a esquerda, a partir do "pixel-on" vizinho encontrado à direita do ponto de cruzamento *PC1*. Caso contrário, a matriz-original é varrida para cima e/ou para a esquerda, a partir do ponto de cruzamento *PC1*. Enquanto a matriz-original é varrida para cima e/ou para a esquerda, se é encontrada a coordenada CFLHS, o programa retorna ao ponto de cruzamento encontrado no PASSO 2 ou PASSO 6. O programa só retorna ao ponto de cruzamento encontrado no PASSO 6, se no PASSO 2 não foi encontrado um ponto de cruzamento.

- Ao terminar de varrer a matriz-original para cima e/ou para a esquerda, se ((foi encontrada a coordenada CFLHS) e (o último "pixel-on" varrido nesse processo está acima do ponto de cruzamento encontrado no PASSO 2 ou do ponto de cruzamento encontrado no PASSO 6)), o caractere é reconhecido como número 8.

- O programa retorna ao "pixel-on" cuja coordenada foi armazenada no

PASSO 5 e varre a matriz-original para baixo, a partir desse "pixel-on". A matriz-original é varrida para baixo, até que não existam mais "pixels-on" que possam ser varridos nessa direção ou o valor absoluto da diferença entre a coordenada de um "pixel-on" que está sendo varrido e a coordenada CILHI seja menor ou igual a 1. Ao terminar de varrer a matriz-original para baixo, se ((o valor absoluto da diferença entre a abscissa do último "pixel-on" varrido nesse processo e a abscissa ACILHI é menor ou igual a 2) e (o valor absoluto da diferença entre a ordenada desse último "pixel-on" varrido e a ordenada OCILHI é menor ou igual a 1)), o programa varre a matriz-original para a direita, a partir do último "pixel-on" varrido quando essa matriz foi varrida para baixo. A matriz-original é varrida para a direita, até que não existam mais "pixels-on" que possam ser varridos nessa direção ou seja encontrado um ponto de cruzamento (este passo será referenciado adiante como PASSO 7).

- Se ((o ponto de cruzamento encontrado no PASSO 7 está a uma distância maior ou igual a 2 "pixels-on" do ponto de cruzamento encontrado no PASSO 2) ou (não foi encontrado um ponto de cruzamento no PASSO 7)), o caractere é reconhecido como número 3. Caso contrário, o caractere é reconhecido como número 2.

- Se ((no PASSO 3 o último "pixel-on" varrido não possui "pixel-on" vizinho que se encontra na direção 5, 6 ou 7) ou (esse último "pixel-on" possui "pixel-on" vizinho nessas direções e o caractere não foi reconhecido no item anterior)), o programa retorna ao ponto de cruzamento encontrado no PASSO 2 e varre a matriz-original para cima, para baixo, para a esquerda ou para a

direita, conforme os "pixels-on" estejam dispostos. A matriz-original é varrida para cima, para baixo, para a esquerda ou para a direita, até que não existam mais "pixels-on" que possam ser varridos em nenhuma dessas direções. Enquanto a matriz-original é varrida para cima, para baixo, para a esquerda ou para a direita, são armazenadas as coordenadas de pontos de cruzamento encontrados, a ordem em que as coordenadas CFLHS e CILHS são encontradas e se as coordenadas CILHI e CFLHI são encontradas, independente da ordem em que estas 2 últimas coordenadas são encontradas (este passo será referenciado adiante como PASSO 8).

- Enquanto a matriz-original é varrida no PASSO 8, se ((é encontrado um extremo do caractere) e (não são encontradas as coordenadas CILHS e CFLHS)) (este passo será referenciado adiante como PASSO 9), o programa retorna a coordenada CILHS e varre a matriz-original para a direita, até que não existam mais "pixels-on" que possam ser varridos nessa direção, não existam mais pontos de cruzamento cujas coordenadas foram armazenadas ou o último "pixel-on" varrido no PASSO 9 tenha sido encontrado. Enquanto a matriz-original é varrida para a direita, é armazenada a quantidade de "pixels-on" varridos e as coordenadas dos pontos de cruzamento encontrados. Se ((o último "pixel-on" varrido no PASSO 9 não foi encontrado) e (a quantidade de "pixels-on" varridos para a direita quando a matriz-original foi varrida nessa direção é maior do que 4)) (este passo será referenciado adiante como PASSO 10), o caractere é reconhecido como número 5. O programa considera como extremo do caractere um ponto em que não existem mais "pixels-on" vizinhos ao "pixel-on" que está sendo varrido.

- Se no PASSO 9 a coordenada CILHS ou CFLHS foi encontrada e existem

pontos de cruzamento cujas coordenadas foram armazenadas no PASSO 8, o programa anota se as coordenadas CILHS e CFLHS já foram encontradas e retorna ao último ponto de cruzamento encontrado no PASSO 8 cuja coordenada foi armazenada (este passo será referenciado adiante como PASSO 11).

- A matriz-original é varrida no PASSO 8, enquanto existirem "pixels-on" que possam ser varridos para cima, para baixo, para a esquerda ou para a direita, existirem pontos de cruzamento cujas coordenadas foram armazenadas nesse passo e não foram varridos ou se a condição do PASSO 10 for satisfeita. Se ((ao varrer a matriz-original no PASSO 8 foram encontradas as coordenadas CFLHS e CILHS nesta ordem) ou se ((no PASSO 11 as coordenadas CILHS e/ou CFLHS não foram encontradas) e (no PASSO 8 as coordenadas CILHS e CFLHS foram encontradas nesta ordem))), o programa verifica se no PASSO 8 foram encontradas as coordenadas CILHI e CFLHI, independente da ordem em que estas coordenadas foram encontradas. Em caso afirmativo, o caractere é reconhecido como número 8. Caso contrário, se foi encontrado no PASSO 8 apenas a coordenada CILHI, o caractere é reconhecido como número 0. Se no PASSO 8 a coordenada CILHI não foi encontrada, o programa verifica se após varrer mais de 3 "pixels-on" nesse passo foi varrido algum "pixel-on" que esteja próximo (distância menor ou igual a 1 "pixel-on") de algum ponto de cruzamento cuja coordenada foi armazenada nesse mesmo passo. Em caso afirmativo, o caractere é reconhecido como número 9. Caso contrário, o caractere é reconhecido como número 3.

- Se ((ao varrer a matriz-original no PASSO 8 não foram encontradas as coordenadas CFLHS e CILHS nesta ordem) e se ((no PASSO 11 as

coordenadas CILHS e CFLHS foram encontradas) ou (no PASSO 8 as coordenadas CILHS e CFLHS não foram encontradas nesta ordem))), o programa verifica se no PASSO 8 foram encontradas as coordenadas CILHS e CFLHS, nesta ordem. Em caso afirmativo, o caractere é reconhecido como número 5. Se apenas a coordenada CILHS foi encontrada no PASSO 8, o programa retorna ao ponto de cruzamento encontrado no PASSO 2 e a partir deste ponto de cruzamento varre a matriz-original para a direita. A matriz-original é varrida para a direita, até que não existam mais "pixels-on" que possam ser varridos nessa direção ou seja encontrada a coordenada CFLHS. Se ao varrer a matriz-original para a direita é encontrada a coordenada CFLHS, o caractere é reconhecido como número 3. Se ((a coordenada CFLHS não é encontrada ao varrer a matriz-original para a direita) ou (essa coordenada foi a primeira coordenada a ser encontrada no PASSO 8)), o caractere não é reconhecido na função verif-2-3-5-7-9-8-0.

Se o PASSO 4 não foi satisfeito, o programa segue os seguintes procedimentos:

- A matriz-original é varrida para a direita a partir da coordenada CILHS, até que não existam mais "pixels-on" que possam ser varridos nessa direção. Enquanto a matriz-original é varrida para a direita é verificado se a coordenada de algum "pixel-on" que está sendo varrido é igual a coordenada do ponto de cruzamento encontrado no PASSO 2 e se é a primeira vez que isto ocorre (a variável que indica se esta condição ocorreu ou não, é a variável **ind0**. Quando esta condição ocorrer o valor da variável **ind0** será igual a 1), o programa iguala a variável **ind0** a 2 e continua a varrer a matriz-original para a direita a partir da coordenada CILHS. Enquanto a matriz-original é varrida para a

direita, se ((o valor absoluto da diferença entre a coordenada CFLHS e a coordenada de algum "pixel-on" que está sendo varrido é menor ou igual a 2) e (o valor da variável **ind0** é igual a 2)), a variável **ind0** é igualada a 3. Ao terminar de varrer a matriz-original para a direita, se o valor da variável **ind0** é maior ou igual a 2, o programa verifica se ((o valor da variável **ind0** é igual a 3) e (ao varrer a matriz-original no PASSO 3 foi encontrada a coordenada CFLHS)). Em caso afirmativo, o caractere é reconhecido como número 0. Caso contrário, o caractere é reconhecido como número 5.

- O programa retorna ao ponto de cruzamento encontrado no PASSO 2 e a partir deste ponto de cruzamento, varre a matriz-original para cima, para a direita ou para a esquerda conforme os "pixels-on" estejam dispostos. A matriz-original é varrida para cima, para a direita ou para a esquerda, até que não existam mais "pixels-on" que possam ser varridos em nenhuma dessas direções. Enquanto a matriz-original é varrida para cima, para a direita ou para a esquerda são armazenadas as coordenadas dos pontos de cruzamento encontrados, a ordem em que são encontradas as coordenadas CILHS e CFLHS, a quantidade de "pixels-on" varridos que se encontram nas direções 0, 1, 2 ou 3 e se é encontrada a coordenada CILHI (este passo será referenciado adiante como PASSO 12). Quando não existem mais "pixels-on" que possam ser varridos para cima, para a direita ou para a esquerda na matriz-original ou o valor da variável **"condição-de-folga"** é diferente de 0, o programa verifica se as coordenadas CILHS e CFLHS foram encontradas ao varrer a matriz-original para cima, para a direita ou para a esquerda. Em caso afirmativo, é armazenada a coordenada do último "pixel-on" varrido para cima, para a direita ou para a esquerda e se o valor da variável **"condição-de-folga"** é diferente de 0, o programa finaliza o processo de varrer a matriz-original para cima, para a

direita ou para a esquerda (este passo será referenciado adiante como PASSO 13). Ao terminar de varrer a matriz-original para cima, para a direita ou para a esquerda se no PASSO12 foram armazenadas coordenadas de pontos de cruzamento e os "pixels-on" destes pontos de cruzamento não foram varridos, o programa volta a varrer a matriz-original para cima, para a direita ou para a esquerda a partir de um desses pontos de cruzamento cujas coordenadas foram armazenadas. A matriz-original é varrida para cima, para a direita ou para a esquerda, até que não existam mais "pixels-on" que possam ser varridos em nenhuma dessas direções ou não existam mais coordenadas de pontos de cruzamento armazenadas. No processo de varrer a matriz-original para cima, para a direita ou para a esquerda, se o ponto de cuzamento encontrado possui "pixel-on" vizinho que se encontra na direção 0, 1 ou 7, será dada preferência a varrer a matriz-original inicialmente para a direita. Caso contrário, será dada preferência a varrer a matriz-original inicialmente para a esquerda. Se não existem mais "pixels-on" que possam ser varridos para cima, para a direita ou para a esquerda e todos os pontos de cruzamento encontrados no PASSO 12 foram varridos, o programa retorna ao ponto de cruzamento encontrado no PASSO 2 e continua a varrer a matriz-original inicialmente para a esquerda, para cima ou para a direita, conforme os "pixels-on" estejam dispostos. O programa varre a matriz-original para a esquerda, para cima ou para a direita, até que não existam mais "pixels-on" que possam ser varridos em nenhuma dessas direções, não existam mais pontos de cruzamento encontrados no PASSO 12 a serem varridos, seja encontrado um "pixel-on" cujo valor absoluto da diferença entre a sua coordenada e a coordenada CFLHI seja menor ou igual a 2, as coordenadas CILHS e CFLHS sejam encontradas ou seja varrido um "pixel-on" cujo valor absoluto da diferença entre a sua ordenada e a ordenada do último "pixel-on" varrido no PASSO 12 e

armazenado no PASSO 13 seja menor ou igual a 2.

- Quando a matriz-original foi varrida para cima, para a direita ou para a esquerda no PASSO 12, se foram encontradas as coordenadas CFLHS e CILHS nesta ordem, o programa verifica se ((o valor absoluto da diferença entre a coordenada do último "pixel-on" varrido no PASSO 12 e a coordenada CFLHI é menor ou igual a 1) e (não passou pela coordenada CILHI) ou (o valor absoluto da diferença entre a coordenada desse último "pixel-on" varrido e a coordenada do "pixel-on" armazenada no PASSO 13 é menor ou igual a 2)). Em caso afirmativo, o caractere é reconhecido como número 9. Caso contrário, o caractere é reconhecido como número 2.

- Quando a matriz-original foi varrida para cima, para a direita ou para a esquerda no PASSO 12, se foi encontrada apenas a coordenada CFLHS ou foram encontradas as coordenadas CILHS e CFLHS nesta ordem, o caractere é reconhecido como número 5.

- Se no PASSO 12 foi finalizado o processo de varrer a matriz-original para cima, para a direita ou para a esquerda, por não existirem mais "pixels-on" que pudessem ser varridos nesse passo, o programa verifica se ((o valor absoluto da diferença entre a coordenada do último "pixel-on" varrido no PASSO 12 e a coordenada CFLHS é menor ou igual a 2 e não foi encontrada a coordenada CILHS) ou (o número de "pixels-on" varridos nesse passo é maior do que 2)) (este passo será referenciado adiante como PASSO 14), o caractere é reconhecido como número 9. Caso contrário, o caractere é reconhecido como número 5.

- Se no PASSO 12, o processo de varrer a matriz-original para cima, para a direita ou para a esquerda foi finalizado, por ter sido varrido um "pixel-on" cujo valor absoluto da diferença entre a sua coordenada e a coordenada CFLHI é menor ou igual a 2 ou as coordenadas CILHS e CFLHS foram encontradas, o programa verifica se as coordenadas CFLHS e CILHS não foram encontradas no PASSO 12. Em caso afirmativo, o caractere é reconhecido como número 9. Caso contrário, o caractere é reconhecido como número 5.

Se no PASSO 2, a matriz-original foi varrida e não foi encontrado ponto de cruzamento, o programa retorna a coordenada CFLHS e varre a matriz-original para cima, para baixo, para a direita ou para a esquerda, conforme os "pixels-on" estejam dispostos. Enquanto a matriz-original é varrida para cima, para baixo, para a direita ou para a esquerda é anotado a ordem em que são encontradas as coordenadas CFLHI e CILHI e se encontrou as coordenadas CFLHI e CILHS antes de encontrar a coordenada CILHI ou se encontrou as coordenadas CILHI e CILHS antes de encontrar a coordenada CFLHI (este passo será referenciado adiante como PASSO 15). Se ao varrer a matriz-original para cima, para baixo, para a direita ou para a esquerda é encontrado um extremo do caractere, o programa armazena a ordenada do último "pixel-on" varrido e volta a varrer a matriz-original para cima, para baixo, para a direita ou para a esquerda a partir da coordenada CILHS, dando preferência a varrer essa matriz inicialmente para a esquerda e/ou para baixo, caso a coordenada CILHS possua "pixel-on" vizinho que se encontra na direção 4, 5, 6 ou 7 (este passo será referenciado adiante como PASSO 16). Enquanto a matriz-original é varrida para cima, para baixo, para a direita ou para a esquerda, se o programa encontrar um ponto de cruzamento que possui apenas 2 "pixels-on" vizinhos que são adjacentes, desconsidera o ponto de cruzamento e continua a varrer a matriz-original para

cima, para baixo, para a direita ou para a esquerda. O processo de varrer a matriz-original para cima, para baixo, para a direita ou para a esquerda é repetido, até que não existam mais "pixels-on" que possam ser varridos em nenhuma dessas direções, seja encontrado um ponto de cruzamento ou o PASSO 16 tenha sido executado uma vez (este passo será referenciado adiante como PASSO 17).

Enquanto a matriz-original era varrida para cima, para baixo, para a direita ou para a esquerda no PASSO 17, se foi encontrado um ponto de cruzamento (este passo será referenciado adiante como PASSO 18), a matriz-original é varrida para cima ou para baixo conforme os "pixels-on" estejam dispostos, a partir do ponto de cruzamento encontrado. A matriz-original é varrida para cima ou para baixo, até que não existam mais "pixels-on" que possam ser varridos em nenhuma dessas direções. Enquanto a matriz-original é varrida para cima ou para baixo, o programa armazena a quantidade de "pixels-on" varridos. Se no PASSO 15 foram encontradas as coordenadas CFLHI e CILHS antes de encontrar a coordenada CILHI, o caractere é reconhecido como número 5. Caso contrário, se ((a quantidade de "pixels-on" varridos para cima ou para baixo foi menor do que 5) ou (foi encontrado um "pixel-on" cujo valor absoluto da diferença entre a sua coordenada e a coordenada CFLHI é menor ou igual a 1)), o caractere é reconhecido como número 2. Caso contrário, o caractere é reconhecido como número 0.

Se não foi encontrado um ponto de cruzamento no PASSO 18, o programa verifica se a primeira coordenada a ser encontrada no PASSO 15 foi a coordenada CFLHI. Em caso afirmativo, é verificado se no PASSO 15 foram encontradas as coordenadas CFLHI e depois CILHS antes de encontrar a coordenada CILHI. Se esta condição é satisfeita, o caractere é reconhecido como número 5. Caso contrário, se no PASSO 15 a coordenada CILHS não foi

encontrada ou esta coordenada foi encontrada após a coordenada CILHI, o programa verifica se o valor absoluto da diferença entre a ordenada do último "pixel-on" varrido no PASSO 17 e a ordenada do "pixel-on" armazenado no PASSO 16 é maior do que 9. Em caso afirmativo, o caractere é reconhecido como número 3. Se o valor absoluto da diferença entre a ordenada do último "pixel-on" varrido no PASSO 17 e a ordenada do "pixel-on" armazenado no PASSO 16 é maior do que 4 e menor ou igual a 9, o programa varre a matriz-original de cima para baixo, a partir do ponto médio entre as abscissas ACFLHS e ACILHS, até encontrar o primeiro "pixel-on". Encontrado este "pixel-on", a matriz-original é varrida para a direita ou para a esquerda, conforme os "pixels-on" estejam dispostos. O processo de varrer a matriz-original para a direita ou para a esquerda é repetido, até que não existam mais "pixels-on" que possam ser varridos em nenhuma dessas direções ou o "pixel-on" que está sendo varrido e o "pixel-on" que acabou de ser varrido estejam na direção 4. Se ao varrer a matriz-original para a direita ou para a esquerda foram encontrados apenas conjuntos de "pixels-on" que se encontram nas direções 7, 6, 5 e 4 nesta ordem, o caractere é reconhecido como número 0. Caso contrário, o caractere é reconhecido como número 3.

Se não foi encontrado ponto de cruzamento no PASSO 18 ou a coordenada CFLHI não foi a primeira coordenada a ser encontrada no PASSO 15, o programa segue os seguintes procedimentos:

- Se no PASSO 15 as coordenadas CFLHI e CILHI foram encontradas nesta ordem, o caractere é reconhecido como número 2. Caso contrário, se no PASSO 15 a coordenada CFLHI ou CILHI foi encontrada ou a coordenada CILHS foi encontrada antes da coordenada CFLHS, o caractere não é reconhecido na função verif-2-3-5-7-9-8-0.

- Se no PASSO 15 as coordenadas CFLHI e CILHI não foram encontradas, o programa varre a matriz-original 2 vezes:

  - Na primeira vez, a matriz-original é varrida para a direita a partir da coordenada CILHS, até que não existam mais "pixels-on" que possam ser varridos nessa direção. Enquanto a matriz-original é varrida para a direita, é anotado se são varridos conjuntos de "pixels-on' que se encontram nas direções (0 e depois 7) ou (0 e depois 6) (este passo será referenciado adiante como PASSO 19). Ao terminar de varrer a matriz-original para a direita, o programa armazena a coordenada do último "pixel-on" varrido nesse processo e varre a matriz-original para a direita a partir da coordenada CILHI. A matriz-original é varrida para a direita, até que não existam mais "pixels-on" que possam ser varridos nessa direção. Enquanto a matriz-original é varrida para a direita, é anotado se são varridos conjuntos de "pixels-on" que se encontram nas direções (0 e depois 1) ou (0 e depois 2) (este passo será referenciado adiante como PASSO 20). Ao terminar de varrer a matriz-original para a direita no PASSO 20, se ((o valor absoluto da diferença entre a ordenada do último "pixel-on" varrido no PASSO 20 e a ordenada do último "pixel-on" varrido no PASSO 19 é menor ou igual a 4) e (foram varridos conjuntos de "pixels-on" que se encontram nas direções (0 e depois 7) ou (0 e depois 6) no PASSO 19) e (no PASSO 20 foram varridos conjuntos de "pixels-on" que se encontram nas direções (0 e depois 1) ou (0 e depois 2))), o programa varre a matriz-original pela segunda vez. Caso contrário, o caractere é reconhecido como número 5.

- Ao varrer a matriz-original pela segunda vez, o programa retorna à coordenada CILHS e varre essa matriz para baixo, até que não existam mais "pixel-on" que possam ser varridos nessa direção. Ao terminar de varrer a matriz-original para baixo, o programa armazena a coordenada do último "pixel-on" varrido nesse processo (este passo será referenciado adiante como PASSO 21). Após isto, a partir da coordenada CILHI a matriz-original é varrida para cima, até que não existam mais "pixels-on" que possam ser varridos nessa direção. Ao terminar de varrer a matriz-original para cima, se o valor absoluto da diferença entre a ordenada do último "pixel-on" varrido para cima e a ordenada do último "pixel-on" varrido no PASSO 21 é menor ou igual a 4, o caractere é reconhecido como número 0. Caso contrário, o caractere é reconhecido como número 5.

Se a quantidade de "pixels-on" varridos na direção vertical no PASSO 1 foi menor ou igual a 2, o programa verifica se foi encontrado um ponto de cruzamento nesse passo. Em caso afirmativo, a matriz-original é varrida para a direita ou para a esquerda conforme os "pixels-on" estejam dispostos, a partir do ponto de cruzamento encontrado no PASSO 1. A matriz-original é varrida para a direita ou para a esquerda, até que não existam mais "pixels-on" que possam ser varridos em nenhuma dessas direções. Enquanto a matriz-original é varrida para a direita ou para a esquerda, se são encontradas as coordenadas CILHS, CFLHS e passar próximo (diferença de 1 "pixel-on") da coordenada do ponto de cruzamento encontrado no PASSO 1, o caractere é reconhecido como número 0. Se a quantidade de "pixels-on" varridos na direção vertical no PASSO 1 foi menor ou igual a 2 e não foi encontrado ponto de cruzamento nesse passo ou se o ponto de cruzamento foi encontrado e o caractere não foi reconhecido como

número 0 no procedimento acima, o programa retorna a coordenada CFLHS e varre a matriz-original para baixo. A matriz-original é varrida para baixo, até que não existam mais "pixels-on" que possam ser varridos nessa direção, seja encontrado um ponto de cruzamento, a coordenada CILHS ou um "pixel-on" cujo valor absoluto da diferença entre a sua coordenada e a coordenada CILHI é menor do que 2 (este passo será referenciado adiante como PASSO 22).

Ao terminar de varrer a matriz-original para baixo no PASSO 22, os seguintes procedimentos são realizados:

- O programa retorna a coordenada CFLHI e varre a matriz-original para a esquerda, para cima ou para baixo, conforme os "pixels-on" estejam dispostos (este passo será referenciado adiante como PASSO 23).

- Enquanto a matriz-original é varrida para a esquerda, para cima ou para baixo no PASSO 23, se o valor da variável **"condição-de-folga"** é igual a 0, o programa armazena na variável **vet-dir** a quantidade de "pixels-on" varridos nesse passo e verifica se ((o "pixel-on" que está sendo varrido no PASSO 23 encontra-se na direção 3, 4 ou 5) e (a coordenada CILHS foi encontrada no PASSO 22 ou o valor da variável **vet-dir1** é igual a 0)). Em caso afirmativo, a variável **vet-dir0** que armazena a quantidade de "pixels-on" varridos que se encontram na direção 3, 4 ou 5 é incrementada de uma unidade. O programa verifica se ((foi varrido no PASSO 23 um "pixel-on" que se encontra na direção 1 ou 2) e (o valor da variável **vet-dir2** é igual a 0)). Em caso afirmativo, a variável **vet-dir0** é incrementada de uma unidade. O objetivo das variáveis **vet-dir1** e **vet-dir2** é explicado no PASSO 26.

Enquanto a matriz-original é varrida para a esquerda, para cima ou para baixo no PASSO 23, se ((é encontrado um ponto de cruzamento ou no PASSO 22

foi varrido um "pixel-on" cujo valor absoluto da diferença entre a sua coordenada e a coordenada CILHI é menor do que 2) ou (foi encontrado no PASSO 23 um "pixel-on" cujo valor absoluto da diferença entre a sua coordenada e a coordenada do ponto de cruzamento encontrado no PASSO 22 é menor do que 2) e (no PASSO 22 não passou próximo (diferença menor do que 2 "pixels-on") da coordenada CILHI)) (este passo será referenciado adiante como PASSO 25), o programa verifica se ((a quantidade de "pixels-on" varridos no PASSO 23 é diferente de 0) e (a variável **vet-dir1** é igual a 0)). Em caso afirmativo, a porcentagem de "pixels-on" varridos que se encontram na direção 3, 4 ou 5 é armazenada na variável **vet-dir1**. Caso contrário, se o valor da variável **vet-dir2** é igual a 0, o programa armazena nesta variável a porcentagem de "pixels-on" varridos que se encontram na direção 1 ou 2 (este passo será referenciado adiante como PASSO 26). O programa varre a matriz-original para cima, a partir do ponto de cruzamento encontrado no PASSO 25, armazenando na variável **vet-dir** a quantidade de "pixels-on" varridos e na variável **vet-dir0** a quantidade de "pixels-on" varridos que se encontram na direção 1 ou 2 (este passo será referenciado adiante como PASSO 27). O PASSO 27 é repetido enquanto ao varrer a matriz-original é encontrado um ponto de cruzamento que possui 2 "pixels-on" vizinhos.

- A matriz-original é varrida para a esquerda, para cima ou para baixo no PASSO 23, enquanto existirem "pixels-on" que possam ser varridos em alguma dessas direções, não for encontrado ponto de cruzamento e não for satisfeita a condição do PASSO 25.

O programa retorna a coordenada CFLHI e se não foi encontrado um ponto de cruzamento no PASSO 22, a partir dessa coordenada varre a matriz-original para a esquerda, até que não existam mais "pixels-on" que possam ser varridos

nessa direção ou seja encontrado um ponto de cruzamento (este passo será referenciado adiante como PASSO 28). Enquanto a matriz-original é varrida para a esquerda se ((é encontrado ponto de cruzamento) e (no PASSO 22 a coordenada CILHS foi encontrada) e (o ponto de cruzamento encontrado no PASSO 28 possui "pixels-on" vizinhos que se encontram nas direções (((3, 4 ou 5) e (1 ou 2)) ou (4 ou 5) e (1, 2 ou 3)))), a variável **vet-dir2** é igualada a 0 (este passo será referenciado adiante como PASSO 29).

Se no PASSO 22 foi encontrada a coordenada CILHS, o programa verifica se a porcentagem de "pixels-on" varridos no PASSO 24 que se encontram nas direçõs 1 e 2 é maior do que 70% da quantidade de "pixels-on" varridos no PASSO 23. Em caso afirmativo, o caractere é reconhecido como número 5. Caso contrário, o programa armazena na variável **vet-dir0** a porcentagem de "pixels-on" varridos no PASSO 24 que se encontram na direção 1 ou 2.

Enquanto a matriz-original é varrida para a esquerda, para cima ou para baixo no PASSO 23, se ((no PASSO 22 foi encontrado um "pixel-on" cujo valor absoluto da diferença entre a sua coordenada e a coordenada CILHI é menor do que 2) ou ((é encontrado no PASSO 23 um "pixel-on" cujo valor absoluto da diferença entre a sua coordenada e a coordenada do ponto de cruzamento encontrado no PASSO 22 é menor do que 2) e (neste passo não passou próximo (diferença menor do que 2 "pixels-on") da coordenada CILHI) e (o valor da variável **vet-dir1** é maior do que 70% da quantidade de "pixels-on" varridos no PASSO 23) e ((o valor da variável **vet-dir2** é maior do que 70% da quantidade total de "pixels-on" varridos no PASSO 23) ou (o valor da variável **vet-dir2** é igual a 0)))), o caractere é reconhecido como número 2. Caso contrário, se foi encontrado um ponto de cruzamento no PASSO 22, a partir deste ponto de cruzamento o programa varre a matriz-original para a esquerda, até que não existam mais "pixels-on" que possam ser varridos nessa direção. Enquanto a

matriz-original é varrida para a esquerda, o programa armazena a quantidade de "pixels-on" varridos nessa direção e a quantidade destes "pixels-on" que se encontram na direção 4. Se a quantidade de "pixels-on" varridos que se encontram na direção 4 é maior ou igual a 80% da quantidade de "pixels-on" varridos para a esquerda, o caractere é reconhecido como número 7. Caso contrário, o caractere não é reconhecido na função verif-2-3-5-7-9-8-0.

A figura 4.22 ilustra alguns números reconhecidos pela função verif-2-3-5-7-9-8-0.

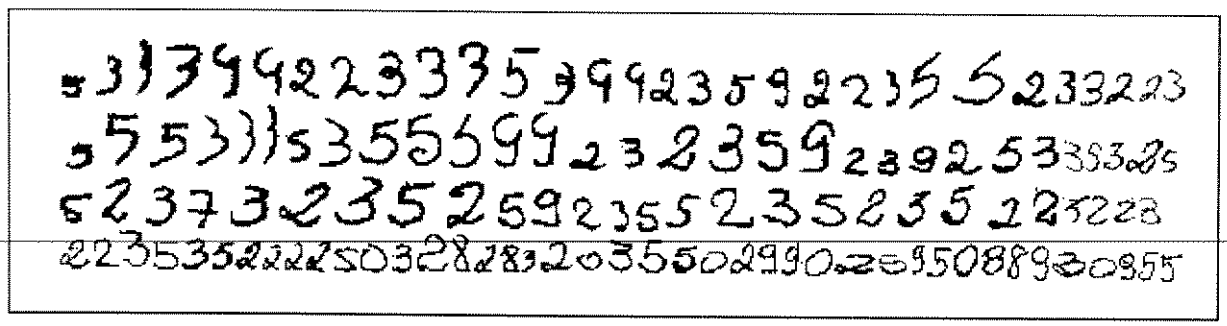


Figura 4.22 - Alguns números reconhecidos pela função verif-2-3-5-7-9-8-0.

Verif-7-9-4-1-2-0-6-8-3

O objetivo da função verif-7-9-4-1-2-0-6-8-3 é reconhecer os dígitos 7, 9, 4, 1, 2, 0, 6, 8 e 3.

Para facilitar o entendimento da função verif-7-9-4-1-2-0-6-8-3, optou-se por explicá-la no texto e esboçar o seu fluxograma. Aconselha-se ao leitor que para sua melhor clareza, leia o texto que descreve a função verif-7-9-4-1-2-0-6-8-3 e acompanhe ao mesmo tempo o seu fluxograma. A figura 4.23 que se encontra nas páginas 538 a 558 do 2º volume deste trabalho, ilustra o fluxograma da função verif-7-9-4-1-2-0-6-8-3. Nesta figura, alguns comentários que se encontram no texto não constam do fluxograma. Devido à extensão do

fluxograma ele foi dividido em 20 partes.

O programa inicia a função verif-7-9-4-1-2-0-6-8-3, varrendo a matriz-original para cima a partir da coordenada CFLHI, até que não existam mais "pixels-on" que possam ser varridos nessa direção ou seja encontrado um ponto de cruzamento (este passo será referenciado adiante como PASSO 1). A matriz-original armazena os "pixels-on" e os "pixels-off" que constituem o esqueleto da imagem de um caractere.

Se foi encontrado um ponto de cruzamento no PASSO 1, é verificado se este ponto de cruzamento possui "pixel-on" vizinho que se encontra na direção 4, 5 ou 3, nesta ordem. Em caso afirmativo, a matriz-original é varrida para a esquerda e/ou para cima, a partir do "pixel-on" vizinho encontrado. A matriz-original é varrida para a esquerda e/ou para cima, até que não existam mais "pixels-on" que possam ser varridos nessas direções, seja encontrado um ponto de cruzamento ou a quantidade de "pixels-on" varridos que se encontram na direção vertical seja maior ou igual a 2. Enquanto a matriz-original é varrida para a esquerda e/ou para cima é armazenada a quantidade de "pixels-on" varridos que se encontram na direção vertical (direção 2) e se é encontrado um ponto de cruzamento é verificado se o valor absoluto da diferença entre a sua coordenada e a coordenada do ponto de cruzamento encontrado no PASSO 1 é menor do que 3. Em caso afirmativo, o ponto de cruzamento encontrado ao varrer a matriz-original para a esquerda e/ou para cima é desconsiderado e é finalizado o processo de varrer essa matriz nessas direções. Ao terminar de varrer a matriz-original para a esquerda e/ou para cima, o programa armazena a coordenada do último "pixel-on" varrido em alguma dessas direções. A partir desse último "pixel-on", a matriz-original é varrida para cima e/ou para a direita conforme os "pixels-on" estejam dispostos. A matriz-original é varrida para cima e/ou para a direita, até que não existam mais "pixels-on" que possam ser varridos nessas

direções ou seja encontrada a coordenada CFLHS (este passo será referenciado adiante como PASSO 2). Enquanto a matriz-original é varrida para cima e/ou para a direita no PASSO 2, se o programa passou próximo (distância menor ou igual a 1 "pixel-on") da coordenada CILHS, é anotado se a coordenada CFLHS é encontrada e se é varrido algum "pixel-on" cujo valor absoluto da diferença entre a sua coordenada e a coordenada do ponto de cruzamento encontrado no PASSO 1 é menor ou igual a 1. Se o ponto de cruzamento encontrado no PASSO 1 não possui "pixel-on" vizinho que se encontra na direção 4, 5 ou 3, o programa armazena a coordenada desse ponto de cruzamento. Esta coordenada também será referenciada adiante como tendo sido armazenada no PASSO 2.

Se o "pixel-on" cuja coordenada foi armazenada no PASSO 2 (ponto de cruzamento encontrado no PASSO 1 ou o último "pixel-on" encontrado no PASSO 2) possui "pixel-on" vizinho que se encontra na direção 1, 2 ou 3, a matriz-original é varrida para cima a partir do "pixel-on" vizinho encontrado. A matriz-original é varrida para cima, até que não existam mais "pixels-on" que possam ser varridos nessa direção ou seja encontrada a coordenada CILHS. Enquanto a matriz-original é varrida para cima é armazenada a quantidade de "pixels-on" varridos que se encontram na direção 1, 2 ou 3 (este passo será referenciado adiante como PASSO 3).

Se ((a coordenada CILHS foi encontrada no PASSO 3) e (a coordenada armazenada no PASSO 2, quando a matriz-original foi varrida para a esquerda coincide com a coordenada do ponto de cruzamento encontrado no PASSO 1)), o programa continua a varrer a matriz-original para a direita ou para a esquerda conforme os "pixels-on" estejam dispostos; a partir do último "pixel-on" varrido no PASSO 3. A matriz-original é varrida para a direita ou para a esquerda, até que não existam mais "pixels-on" que possam ser varridos em nenhuma dessas direções, não existam mais pontos de cruzamento cujas coordenadas são

armazenadas enquanto a matriz-original é varrida para a direita ou para a esquerda ou seja varrido algum "pixel-on" cujo valor absoluto da diferença entre a sua coordenada e a coordenada do ponto de cruzamento encontrado no PASSO 1 seja menor ou igual a 2. Enquanto a matriz-original é varrida para a direita ou para a esquerda, o programa armazena as coordenadas dos pontos de cruzamento encontrados. Ao terminar de varrer a matriz-original para a direita ou para a esquerda, se foram armazenadas as coordenadas de pontos de cruzamento enquanto essa matriz era varrida em alguma dessas direções, o programa retorna a um dos pontos de cruzamento cuja coordenada foi armazenada e continua a varrer a matriz-original para a direita ou para a esquerda. Se o valor absoluto da diferença entre a coordenada de algum "pixel-on" varrido para a direita ou para a esquerda e a coordenada do ponto de cruzamento encontrado no PASSO 1 é menor ou igual a 2, o caractere é reconhecido como número 0.

O programa retorna ao "pixel-on" cuja coordenada foi anotada no PASSO 2 e verifica se (((esse "pixel-on" não possui "pixel-on" vizinho que se encontra na direção 1, 2 ou 3) e (não foi encontrado um ponto de cruzamento no PASSO 2) e (a quantidade de "pixels-on" varridos que se encontram na direção vertical no PASSO 2 é menor do que 2)) ou (o último "pixel-on" varrido no PASSO 1 não possui "pixel-on" vizinho que se encontra na direção 3, 4 ou 5) ou (a quantidade de "pixels-on" varridos que se encontram nas direções 1, 2 e 3 no PASSO 3 é menor do que 2)) (este passo será referenciado adiante como PASSO 4), se pelo menos uma dessas condições é satisfeita, o programa segue os seguintes passos:

- O programa varre a matriz-original para a direita ou para a esquerda conforme os "pixels-on" estejam dispostos, a partir do "pixel-on" cuja coordenada foi armazenada no PASSO 2. Enquanto a matriz-original é varrida para a direita ou para a esquerda, os seguintes procedimentos são realizados:

- Se é encontrado um ponto de cruzamento, o programa verifica se este ponto de cruzamento possui "pixel-on" vizinho que se encontra na direção 4, 5, 6 ou 7. Em caso afirmativo, o programa varre a matriz-original para a direita ou para a esquerda conforme os "pixels-on" estejam dispostos, a partir do "pixel-on" vizinho que se encontra na direção 4, 5, 6 ou 7 (este passo será referenciado adiante como PASSO 5).

- Se ((não existem mais "pixels-on" que possam ser varridos para a direita ou para a esquerda) e (o último "pixel-on" varrido em alguma dessas direções está próximo (diferença menor ou igual a 2 "pixels-on") da coordenada CFLHI)), o programa retorna a coordenada do último "pixel-on" varrido no PASSO 2 e continua a varrer a matriz-original para a direita ou para a esquerda. Caso contrário, o programa finaliza o processo de varrer a matriz-original para a direita ou para a esquerda.

- A matriz-original é varrida para a direita ou para a esquerda, até que não existam mais "pixels-on" que possam ser varridos em nenhuma dessas direções, a coordenada CILHS ou CFLHS seja encontrada ou não seja encontrado um ponto de cruzamento que satisfaça as condições do PASSO 6 (este passo será descrito logo em seguida).

- Se ao varrer a matriz-original para a direita ou para a esquerda não foram encontradas as coordenadas CFLHS e CILHS, o programa verifica se ((o valor absoluto da diferença entre a ordenada do último "pixel-on" varrido no PASSO 2 e a ordenada OCILHI é maior do que 1) e (não foi encontrada a coordenada

CFLHS no PASSO 2)). Em caso afirmativo, o caractere é reconhecido como número 7. Caso contrário, o caractere é reconhecido como número 2.

Se as condições do PASSO 4 foram satisfeitas, o programa retorna à coordenada CILHS ou CFLHS dependendo de qual destas coordenadas tenha sido encontrada ao varrer a matriz-original para a direita ou para a esquerda no processo subseqüente ao PASSO 4. Se o PASSO 3 foi executado e neste passo foi encontrado um ponto de cruzamento, o programa retorna ao ponto de cruzamento encontrado no PASSO 3. Se o PASSO 2 foi executado e neste passo foi encontrado um ponto de cruzamento, o programa retorna ao ponto de cruzamento encontrado no PASSO 2. Caso contrário, o programa retorna a coordenada que foi armazenada no PASSO 2 (ponto de cruzamento encontrado no PASSO 1 ou o último "pixel-on" encontrado no PASSO 2) e varre a matriz-original para cima (direção 1, 2 ou 3). A matriz-original é varrida para cima, até que não existam mais "pixels-on" que possam ser varridos nessa direção ou seja varrido um "pixel-on" cujo valor absoluto da diferença entre a sua coordenada e a coordenada CILHS seja menor ou igual a 1. Enquanto a matriz-original é varrida para cima, é anotada a direção dos "pixels-on" que foram varridos e a quantidade de "pixels-on" varridos (este passo será referenciado adiante como PASSO 6).

A matriz-original é varrida para baixo a partir da coordenada CFLHS, até que não existam mais "pixels-on" que possam ser varridos nessa direção, seja encontrado um ponto de cruzamento que está abaixo do ponto de cruzamento encontrado no PASSO 1 ou cujo valor absoluto da diferença entre a sua ordenada e a ordenada do ponto de cruzamento encontrado no PASSO 1 seja maior do que 3. Enquanto a matriz-original é varrida para baixo, o programa anota a direção do "pixel-on" que está sendo varrido, a quantidade de "pixels-on" varridos nesse processo e se passa pelo ponto de cruzamento encontrado no PASSO 1 (este passo será referenciado adiante como PASSO 7). Ao varrer a matriz-original para

baixo, se o "pixel-on" que está sendo varrido encontra-se na direção diagonal (direção 5 ou 7), o programa considera que esse "pixel-on" encontra-se na direção do último "pixel-on" varrido (esta direção pode ser a direção vertical ou a direção horizontal). O objetivo desta consideração é restringir os "pixels-on" que se encontram nas direções diagonais a direções horizontais e a direções verticais.

Ao terminar de varrer a matriz-original para baixo, o programa retorna a coordenada CFLHS e varre a matriz-original para a esquerda, até que não existam mais "pixels-on" que possam ser varridos nessa direção ou seja encontrado um ponto de cruzamento. Enquanto a matriz-original é varrida para a esquerda, é armazenada a quantidade de "pixels-on" varridos que se encontram na direção 4 (este passo será referenciado adiante como PASSO 8). Ao varrer a matriz-original para a esquerda, se o "pixel-on" que está sendo varrido encontra-se na direção diagonal (direção 3 ou 5), o programa considera que que esse "pixel-on" encontra-se na direção do último "pixel-on" varrido (esta direção pode ser a direção vertical ou a direção horizontal). Esta substituição foi realizada pelo mesmo motivo explicado no parágrafo anterior.

O programa retorna ao ponto de cruzamento encontrado no PASSO 1 e varre a matriz-original para cima, até que não existam mais "pixels-on" que possam ser varridos nessa direção ou seja encontrado um ponto de cruzamento (este passo será referenciado adiante como PASSO 9). Ao terminar de varrer a matriz-original para cima, esta matriz é varrida para a direita a partir do último "pixel-on" varrido no PASSO 6, até que não existam mais "pixels-on" que possam ser varridos nessa direção (este passo será referenciado adiante como PASSO 10).

Se (((a porcentagem de "pixels-on" varridos no PASSO 7, que se encontram na direção 6 é maior ou igual a 80% do número total de "pixels-on" varridos nesse passo) e (a porcentagem de "pixels-on" varridos no PASSO 6, que

se encontram na direção 1 é maior ou igual a 80% do número total de "pixels-on" varridos nesse passo)) e ((a quantidade de "pixels-on" varridos no PASSO 6, que se encontram na direção 4 é menor ou igual a 3) ou (o valor absoluto da diferença entre a abscissa do último "pixel-on" varrido no PASSO 9 e a abscissa do último "pixel-on" varrido no PASSO 10 foi maior do que 4)) e ((a coordenada CFLHS não foi encontrada no PASSO 2 e neste passo não foi anotado que algum "pixel-on" varrido encontrava-se próximo do ponto de cruzamento encontrado no PASSO 1) e/ou (no PASSO 7 não passou pelo ponto de cruzamento encontrado no PASSO 1))) (este passo será referenciado adiante como PASSO 11), o caractere é reconhecido como número 4.

Se o PASSO 11 não foi satisfeito, o programa retorna ao ponto de cruzamento encontrado no PASSO 1 e varre a matriz-original para a esquerda e/ou para cima, conforme os "pixels-on" estejam dispostos. A matriz-original é varrida para a esquerda e/ou para cima, até que não existam mais "pixels-on" que possam ser varridos nessas direções ou o valor absoluto da diferença entre a coordenada do "pixel-on" que está sendo varrido e a coordenada CILHS seja menor ou igual a 1 e a quantidade de "pixels-on" varridos durante esse processo seja maior ou igual a 3. Se (((ao varrer a matriz-original para a esquerda e/ou para cima, foi varrido um "pixel-on" cuja coordenada é igual ou está próxima (diferença de 1 "pixel-on") da coordenada CILHS) e (a quantidade de "pixels-on" varridos para cima é maior ou igual a 3)) e ((a porcentagem de "pixels-on" varridos que se encontram nas direções 1, 2 e 3 é maior ou igual a 90% do número de "pixels-on" varridos para a esquerda e/ou para cima) ou ((foi encontrada a coordenada CFLHS no PASSO 2) e (no PASSO 7 foi anotado que passou pelo ponto de cruzamento encontrado no PASSO 1)))) (esta condição será referenciada adiante como CONDIÇÃO 1), a matriz-original é varrida para cima a partir da coordenada CILHI, até que não existam mais "pixels-on" que possam

ser varridos nessa direção. Enquanto a matriz-original é varrida para cima se é encontrado um ponto de cruzamento, o programa armazena a coordenada desse ponto de cruzamento, retorna à coordenada CILHI e continua a varrer a matriz-original para cima. Se ao varrer a matriz-original para cima é varrido algum "pixel-on" que está próximo (distância menor ou igual a 2 "pixels-on") do ponto de cruzamento cuja coordenada foi armazenada nesse processo, o caractere é reconhecido como número 8. Caso contrário, o caractere é reconhecido como número 9.

Se a CONDIÇÃO 1 não foi satisfeita, o programa retorna à coordenada CFLVE e varre a matriz-original para cima, até que não existam mais "pixels-on" que possam ser varridos nessa direção. Enquanto a matriz-original é varrida para cima, é armazenada a quantidade de "pixels-on" varridos nesse processo e a quantidade de "pixels-on" varridos que se encontram na direção 1 ou 2. Ao varrer a matriz-original para cima, se a porcentagem de "pixels-on" varridos que se encontram nas direções 1 e 2 for maior ou igual a 80% da quantidade de "pixels-on" varridos para cima, o caractere é reconhecido como número 1. Caso contrário, o programa retorna à coordenada CFLHI e varre a matriz-original para cima. A matriz-original é varrida para cima, até que não existam mais "pixels-on" que possam ser varridos nessa direção ou seja encontrado um ponto de cruzamento. Enquanto a matriz-original é varrida para cima, o programa anota se são varridos conjuntos de "pixels-on" que se encontram nas direções 1 e 3, nesta ordem. Ao terminar de varrer a matriz-original para cima, se foram varridos conjuntos de "pixels-on" que se encontram na direção 1 e depois na direção 3, o caractere é reconhecido como número 3. Caso contrário, o caractere é reconhecido como número 2.

Se no PASSO 1 não foi encontrado um ponto de cruzamento, o programa verifica se o valor absoluto da diferença entre a coordenada do último "pixel-on"

varrido nesse passo e a coordenada CFLHS é maior do que 2. Em caso afirmativo, programa varre a matriz-original para baixo a partir da coordenada do último "pixel-on" varrido no PASSO 1. A matriz-original é varrida para baixo, até que não existam mais "pixels-on" que possam ser varridos nessa direção. Enquanto a matriz-original é varrida para baixo se é encontrado um "pixel-on" cujo valor absoluto da diferença entre a sua coordenada e a coordenada CILHI é menor ou igual a 1, o caractere é reconhecido como número 6. Caso contrário, o caractere é reconhecido como número 9.

Se no PASSO 1 não foi encontrado um ponto de cruzamento e se foi verificado que o valor absoluto da diferença entre a coordenada do último "pixel-on" varrido nesse passo e a coordenada CFLHS é menor ou igual a 2, o programa varre a matriz-original para a direita a partir da coordenada CILHS, até que não existam mais "pixels-on" que possam ser varridos nessa direção. Enquanto a matriz-original é varrida para a direita, é armazenada na variável **vet-dir0** a quantidade de "pixels-on" varridos que se encontram na direção 0 ou 7 e na variável **vet-dir1** a quantidade de "pixels-on" varridos que se encontram na direção 1 (este passo será referenciado adiante como PASSO 12).

Se no PASSO 12 o valor da variável **vet-dir1** é menor ou igual ao valor da variável **vet-dir0**, o caractere é reconhecido como número 7. Se porém, no PASSO 12 o valor da variável **vet-dir1** é maior do que o valor da variável **vet-dir0**, o programa verifica se o "pixel-on" de maior ordenada pertencente a matriz-original está acima de 40% do número total de linhas da matriz-imagem. Em caso afirmativo, o caractere é reconhecido como número 1. Caso contrário, o caractere é eliminado por ser considerado pelo programa como uma vírgula. A matriz-imagem armazena os "pixels-on" e os "pixels-off" que constituem o esqueleto da imagem da linha de texto.

A figura 4.24 ilustra alguns números reconhecidos pela função verif-7-9-4-

1-2-0-6-8-3.

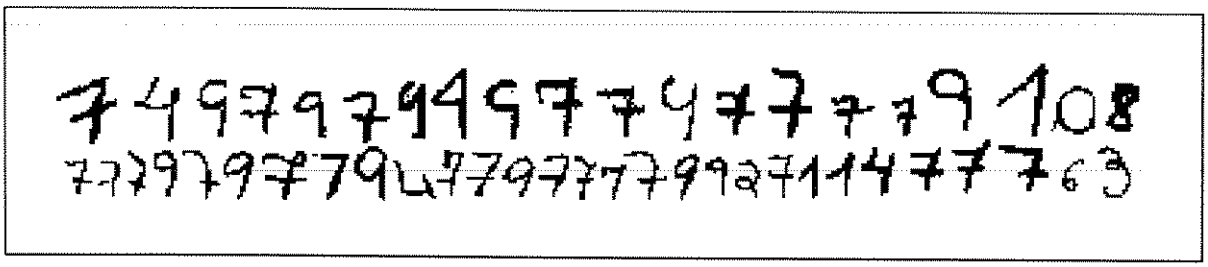


Figura 4.24 - Alguns números reconhecidos pela função verif-7-9-4-1-2-0-6-8-3.

Verif-1-4-5-6-7

O objetivo da função verif-1-4-5-6-7 é reconhecer os dígitos 1, 4, 5, 6 e 7.

Para facilitar o entendimento da função verif-1-4-5-6-7, optou-se por explicá-la no texto e esboçar o seu fluxograma. Aconselha-se ao leitor que para sua melhor clareza, leia o texto que descreve a função verif-1-4-5-6-7 e acompanhe ao mesmo tempo o seu fluxograma. A figura 4.25 que se encontra nas páginas 559 a 570 do 2º volume deste trabalho, ilustra o fluxograma da função verif-1-4-5-6-7. Nesta figura, alguns comentários que se encontram no texto não constam do fluxograma. Devido à extensão do fluxograma ele foi dividido em 11 partes.

O programa inicia a função verif-1-4-5-6-7 varrendo a matriz-original para cima a partir da coordenada CFLHI, até que não existam mais "pixels-on" que possam ser varridos nessa direção ou seja encontrado um ponto de cruzamento (este passo será referenciado adiante como PASSO 1). Enquanto a matriz-original é varrida para cima, se o "pixel-on" que está sendo varrido encontra-se na direção diagonal (direção 1 ou 3), o programa desconsidera que esse "pixel-on" encontra-se na direção diagonal e considera que ele encontra-se na direção do último "pixel-on" varrido (esta direção pode ser a direção vertical ou a direção

horizontal). O objetivo desta consideração é restringir os "pixels-on" que se encontram nas direções diagonais a direções horizontais e a direções verticais. Enquanto a matriz-original é varrida para cima, o programa anota a direção em que se encontra cada "pixel-on" que está sendo varrido. Ao varrer a matriz-original para cima, se é encontrado um ponto de cruzamento as seguintes condições são verificadas: ((o ponto de cruzamento não se encontra na direção 3 em relação ao último "pixel-on" varrido e possui "pixel-on" vizinho que se encontra na direção 3) ou (o ponto de cruzamento não se encontra na direção 4 em relação ao último "pixel-on" varrido e possui "pixel-on" vizinho que se encontra na direção 4) ou (o ponto de cruzamento não se encontra na direção 5 em relação ao último "pixel-on" varrido e possui "pixel-on" vizinho que se encontra na direção 5)). Se uma destas condições é satisfeita, o programa verifica se o "pixel-on" vizinho ao ponto de cruzamento não possui "pixels-on" vizinhos. Em caso afirmativo, o ponto de cruzamento é desconsiderado e o programa continua a varrer a matriz-original para cima. A matriz-original armazena os "pixels-on" e os "pixels-off" que constituem o esqueleto da imagem de um caractere.

Se ao varrer a matriz-original para cima foi encontrado um ponto de cruzamento, o programa verifica se o ponto de cruzamento possui "pixel-on" vizinho que se encontra na direção 0, 1 ou 7, nesta ordem. Em caso afirmativo, a matriz-original é varrida para a direita a partir do "pixel-on" vizinho que se encontra na direção 0, 1 ou 7. A matriz-original é varrida para a direita, até que não existam mais "pixels-on" que possam ser varridos nessa direção. Se ao varrer a matriz-original para a direita foram varridos conjuntos de "pixels-on" que se encontram na direção (1 ou 2) e depois na direção (3 ou 4), o caractere é reconhecido como número 6.

Se o ponto de cruzamento encontrado no PASSO 1 possui "pixel-on"

vizinho que se encontra na direção 3, 4 ou 5, o programa retorna a esse ponto de cruzamento e varre a matriz-original para a esquerda, até que não existam mais "pixels-on" que possam ser varridos nessa direção ou seja encontrado um ponto de cruzamento. Enquanto a matriz-original é varrida para a esquerda, são anotadas as direções dos "pixels-on" que são varridos (este passo será referenciado adiante como PASSO 2). Ao terminar de varrer a matriz-original para a esquerda, o programa armazena a coordenada do último "pixel-on" varrido no PASSO 2.

Se o último "pixel-on" varrido no PASSO 2 possui "pixel-on" vizinho que se encontra na direção 1, 2 ou 3, o caractere é reconhecido como número 4.

O programa retorna ao ponto de cruzamento encontrado no PASSO 1 e varre a matriz-original para cima, até que não existam mais "pixels-on" que possam ser varridos nessa direção ou seja encontrado um ponto de cruzamento (este passo será referenciado adiante como PASSO 3). Enquanto a matriz-original é varrida para cima, se o valor da variável **"condição-horizontal"** é igual a 1, o programa atribui a esta variável o valor 3, com o objetivo de evitar que os "pixels-on" que se encontram na direção horizontal sejam varridos.

Após terminar de varrer a matriz-original no PASSO 3, o programa varre esta matriz para a esquerda a partir do último "pixel-on" varrido nesse passo. A matriz-original é varrida para a esquerda, até que não existam mais "pixels-on" que possam ser varridos nessa direção ou seja encontrado um ponto de cruzamento (este passo será referenciado adiante como PASSO 4). Enquanto a matriz-original é varrida para a esquerda, se o "pixel-on" que está sendo varrido encontra-se na direção diagonal (direção 3 ou 5) e o último "pixel-on" varrido encontra-se na direção horizontal (direção 4), o programa considera que esse "pixel-on" encontra-se direção 4 e incrementa de uma unidade a quantidade de "pixels-on" varridos que se encontram nessa direção O objetivo desta

consideração é restringir os "pixels-on" que se encontram nas direções diagonais a direções horizontais. Enquanto a matriz-original é varrida para a esquerda, se é encontrado um ponto de cruzamento cujo valor absoluto da diferença entre a sua coordenada e a coordenada do ponto de cruzamento encontrado no PASSO 3 (se neste passo é encontrado um ponto de cruzamento) é menor ou igual a 1, o ponto de cruzamento encontrado no PASSO 4 é desconsiderado.

Se no PASSO 3 foi encontrado um ponto de cruzamento, o programa verifica se no PASSO 4 foram varridos mais do que 2 "pixels-on". Em caso afirmativo, a matriz-original é varrida para cima a partir do último "pixel-on" varrido no PASSO 4, até que não existam mais "pixels-on" que possam ser varridos nessa direção. Enquanto a matriz-original é varrida para cima, é armazenada a quantidade de "pixels-on" varridos que se encontram na direção 1, 2 ou 3. Se a quantidade de "pixels-on" varridos para cima que se encontram na direção 1, 2 ou 3 é maior do que 2, o caractere é reconhecido como número 4. Caso contrário, o caractere é reconhecido como número 7. Se no PASSO 4 não foram varridos para cima mais do que 2 "pixels-on" que se encontram na direção 1, 2 ou 3, o programa verifica se na matriz-original o "pixel-on" de maior ordenada está acima de 40% do número total de linhas da matriz-imagem. Em caso afirmativo, o caractere é reconhecido como número 1. Caso contrário, o caractere é eliminado por ser considerado pelo programa como uma vírgula. No capítulo 5 deste trabalho, é explicado o motivo da eliminação da vírgula pelo programa. A matriz-imagem armazena os "pixels-on" e os "pixels-off" que constituem o esqueleto da imagem da linha de texto.

Se no PASSO 3 não foi encontrado um ponto de cruzamento, o programa verifica se ((no PASSO 4 foram varridos mais do que 2 "pixels-on") e (a porcentagem de "pixels-on" varridos no PASSO 4, que se encontram na direção 4 é maior ou igual a 90% do número total de "pixels-on" varridos nesse passo)).

Em caso afirmativo, o caractere é reconhecido como número 7. Caso contrário, o caractere é reconhecido como número 1.

Enquanto a matriz-original é varrida para cima no PASSO 1, se o "pixel-on" que está sendo varrido encontra-se na direção 2, o último "pixel-on" varrido não se encontra nessa direção e o penúltimo "pixel-on" varrido encontra-se na direção 2, o programa considera que o último "pixel-on" varrido encontra-se na direção 2.

Se no PASSO 1 não foi encontrado um ponto de cruzamento, o programa verifica se a porcentagem de "pixels-on" varridos no PASSO 1 que se encontram na direção 2 é menor do que 80% da quantidade de "pixels-on" varridos nesse passo. Em caso afirmativo, o caractere é reconhecido como número 5. Caso contrário, o programa verifica se o "pixel-on" de maior ordenada na matriz-original está acima de 40% do número total de linhas da matriz-imagem. Em caso afirmativo, o caractere é reconhecido como número 1. Se o "pixel-on" de maior ordenada na matriz-original está abaixo de 40% do número total de linhas da matriz-imagem, o caractere é eliminado por ser considerado pelo programa como uma vírgula.

A figura 4.26 ilustra alguns números reconhecidos pela função verif-1-4-5-6-7.

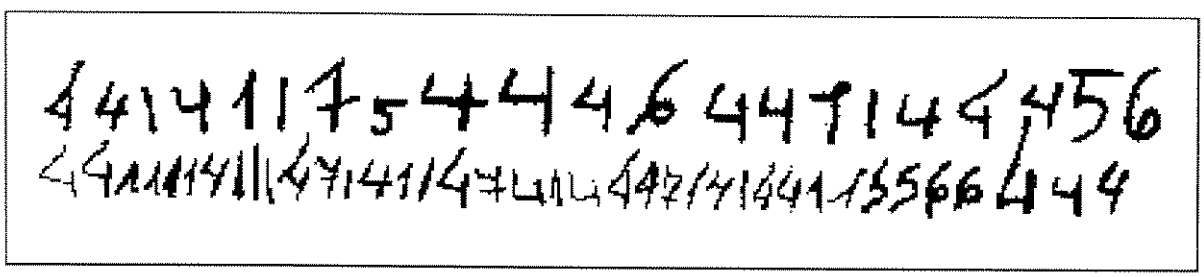


Figura 4.26 - Alguns números reconhecidos pela função verif-1-4-5-6-7.

4.3 - RESULTADOS OBTIDOS

O algoritmo explicado no item 4.2 foi testado com 300 números escritos por 30 pessoas. Foi solicitado a cada uma destas pessoas, que escrevessem em uma folha branca de papel sulfite e em apenas uma linha os números de 0 a 9, isolados uns dos outros. Isto foi feito porque nesta fase do trabalho o algoritmo ainda não está habilitado a reconhecer dígitos sobrepostos ou conectados. Foi pedido às pessoas que escrevessem os números de 0 a 9 em apenas uma linha, pois esta linha representa para o algoritmo de segmentação explicado no capítulo 3, uma linha de texto. Para cada dígito segmentado e afilado, o seu esqueleto é reconhecido pelo algoritmo explicado neste capítulo. Várias pessoas ao escreverem os números de 0 a 9 utilizaram caneta azul, outras utilizaram caneta preta, ou mesmo caneta vermelha. Independente da cor da caneta utilizada todos os números foram varridos por um "scanner" monocromático.

Alguns dos 300 números testados são mostrados nas figuras 4.18, 4.20, 4.22, 4.24, e 4.26.

Dos 300 números escritos, 299 foram reconhecidos corretamente e apenas 1 foi reconhecido de maneira incorreta pelo algoritmo de reconhecimento. Estes resultados apresentaram uma porcentagem de acerto de 99,67% e uma porcentagem de erro de 0,33%.

4.4 - CONCLUSÕES

O algoritmo de reconhecimento de caracteres numéricos manuscritos desenvolvido neste trabalho tem como base a verificação das características intrínsecas de cada dígito e a verificação das características que os dígitos têm em comum. Ao verificar tais caracterísicas, o algoritmo desenvolvido procurou

aproximar-se do processo de reconhecimento utilizado pelo ser humano. O algoritmo de reconhecimento de caracteres numéricos manuscritos, agrupa os "pixels-on" que constituem a imagem de cada dígito em linhas horizontais e linhas verticais. São as coordenadas iniciais e finais dessas linhas que definem as características intrínsecas e comuns de cada um dos dígitos.

As principais características que o algoritmo de reconhecimento de caracteres manuscritos desenvolvido neste trabalho possui são:

a) o reconhecimento dos caracteres manuscritos independe da pessoa que os escreveu;

b) o reconhecimento dos caracteres independe do seu tamanho e da posição em que o caractere foi escrito;

c) desempenho satisfatório, mesmo na presença de ruído;

d) taxa de acerto elevada.

Ao analisar os resultados obtidos no item 4.3 pode-se verificar que o algoritmo desenvolvido neste trabalho para o reconhecimento de caracteres manuscritos isolados apresenta uma excelente porcentagem de reconhecimento.

CAPÍTULO 5

RECONHECIMENTO DE CARACTERES MANUSCRITOS DE CHEQUES BANCÁRIOS

5.1 - INTRODUÇÃO

No capítulo 4 foi descrito o algoritmo desenvolvido neste trabalho que reconhece caracteres numéricos manuscritos isolados. Neste capítulo, são descritos os processos que possibilitam que esse algoritmo seja utilizado no reconhecimento da quantia numérica de cheques bancários.

Na área reservada à escrita do valor numérico do cheque, os caracteres numéricos manuscritos podem estar isolados, sobrepostos ou conectados e apresentar ao longo do seu contorno trechos contínuos ou fragmentados. Além destes caracteres, nessa área são impressos o R e cifrão e são escritos a vírgula e o ponto. O programa prevê a retirada destes 3 últimos caracteres da imagem digitalizada do cheque bancário, com o objetivo que eles não interfiram no processo de reconhecimento dos dígitos.

A área reservada à escrita do valor numérico do cheque é constituída de linhas verticais e/ou horizontais. Neste trabalho considera-se que essas linhas formam o campo que envolve a quantia numérica do cheque bancário. Este campo pode ser gravado junto com a quantia numérica do cheque, quando a área reservada à escrita do valor numérico é varrida pelo "scanner". Caso esse campo esteja presente na matriz-imagem, com o objetivo de que ele não interfira no processo de segmentação e reconhecimento dos caracteres, o programa prevê a sua retirada dessa matriz. A matriz-imagem armazena os "pixels-on" e os "pixels-off" que constituem o esqueleto da área reservada à escrita do valor numérico do cheque bancário.

Além dos caracteres citados anteriormente, podem estar presentes na área reservada à escrita do valor numérico do cheque, caracteres que são utilizados por algumas pessoas na delimitação dessa quantia. Desses caracteres os mais utilizados são as cerquilhas, R e cifrão manuscritos. Para que estes caracteres também não interfiram no reconhecimento dos dígitos, o programa prevê a sua retirada caso eles estejam presentes na matriz-imagem.

No reconhecimento de caracteres manuscritos de cheques, se o valor numérico possui o ponto e a vírgula, estes caracteres são eliminados da matriz-imagem, pois algumas vezes, as pessoas ao escreverem a quantia numérica do cheque, colocam o ponto e a vírgula com dimensões tão pequenas que esses caracteres acabam sendo eliminados pelo algoritmo de afilamento por serem considerados como ruído (a definição de ruído e o que o programa considera ser um ruído foi feita no item 3.5 do capítulo 3 deste trabalho). Se, porém, a dimensão da vírgula é considerável, a sua eliminação é realizada pelo algoritmo de reconhecimento.

No algoritmo de segmentação do texto, explicado no capítulo 3, a primeira segmentação a ser aplicada sobre o texto (imagem lida do arquivo no formato tiff) é a segmentação de suas linhas. O conjunto de "pixels-on" e "pixels-off" que

constituem a imagem de cada linha de texto segmentada, são armazenados em uma matriz denominada no programa de matriz-imagem. Após ser segmentada uma linha de texto, é realizada a segmentação dos caracteres que compõem essa linha. Cada caractere da linha de texto que é segmentado, é armazenado em uma matriz denominada no programa de matriz-original. Após o programa segmentar o caractere, o seu esqueleto é obtido utilizando-se o algoritmo de afilamento ARA explicado no capítulo 3. Ao final da execução deste algoritmo, os "pixels-on" e os "pixels-off" que constituem o esqueleto da imagem do caractere são armazenados na matriz-original e esta matriz é utilizada pelo algoritmo desenvolvido no capítulo 4 para o reconhecimento do caractere.

O algoritmo de segmentação de caracteres utilizado neste trabalho, segmenta os caracteres quando existem "pixels-off" entre eles. O resultado dessa segmentação pode fornecer caracteres isolados, sobrepostos ou conectados. A cada segmentação realizada, os "pixels-on" e os "pixels-off" que constituem a imagem de um ou mais caracteres são armazenados na matriz-original. Devido a isto, esta matriz pode conter os "pixels-on" e os "pixels-off" de um ou mais caracteres. Estes caracteres podem estar isolados, sobrepostos ou conectados. Abaixo são definidos estes 3 tipos de caracteres.

Caracteres Isolados - a matriz-original armazena os "pixels-on" e os "pixels-off" de apenas um caractere. Este caractere pode estar fragmentado ou contínuo. Caractere fragmentado é o caractere em que uma ou mais de suas partes estão desconectadas. Caractere contínuo é o caractere em que todas as suas partes estão conectadas.

Caracteres Sobrepostos - são caracteres isolados que apesar de não se tocarem apresentam "pixels-on" na mesma coluna. Neste

caso, a matriz-original armazena os "pixels-on" e os "pixels-off" da imagem de 2 ou mais caracteres.

Caracteres Conectados - são caracteres que apresentam "pixels-on" em comum. Isto é, um ou mais "pixels-on" pertencem a dois caracteres distintos. Neste caso, a matriz-original armazena os "pixels-on" e os "pixels-off" da imagem de 2 ou mais caracteres.

Neste capítulo, é explicado como o algoritmo realiza a eliminação da vírgula e do ponto na matriz-imagem; é descrito como o algoritmo realiza a segmentação de caracteres sobrepostos [26], é mostrado o processo utilizado no algoritmo para segmentar os caracteres que estão conectados [27, 28]; é explicado como é realizada a eliminação do campo que envolve a área reservada à escrita da quantia numérica. É descrito o processo utilizado para eliminar a cerquilha, é explicado como o programa realiza a eliminação do R e cifrão manuscritos e do R e cifrão impressos. São apresentadas as janelas de entradas e saída do programa, são realizados os comentários sobre os dados utilizados para testar as alterações realizadas neste capítulo, são mostrados os resultados obtidos nestes testes e finalmente são feitas as conclusões a respeito destes resultados.

5.2 - ELIMINAÇÃO DA VÍRGULA E DO PONTO

5.2.1 - Eliminação da Vírgula

A eliminação da vírgula da quantia numérica do cheque bancário, é realizada pelas funções denominadas no programa de elim-virg e virg-1. Com o objetivo de eliminar ruídos, pontos e vírgulas, que porventura podem estar

presentes junto com o caractere na matriz-imagem, o programa executa a função elim-virg para todos os caracteres presentes nessa matriz. Considerando a semelhança do número 1 com a vírgula, o programa executa a função virg-1 quando o caractere é reconhecido como número 1 e não é o primeiro caractere a ser reconhecido. O programa realiza esta consideração, pois a vírgula nunca ocupa a primeira posição na quantia numérica de um cheque bancário. As funções virg-1 e elim-virg são explicadas abaixo:

Virg-1

A função virg-1 tem como objetivo verificar se o número de "pixels-on" que se encontram na direção 4 e 5, é maior ou igual a 90% do número total de "pixels-on" que constituem o caractere reconhecido como número 1. Esta consideração foi realizada, pois foi observado que a maior parte dos "pixels-on" que constituem uma vírgula encontram-se nas direções 4 e 5.

Para facilitar o entendimento da função virg-1, optou-se por explicá-la no texto e esboçar o seu fluxograma. Aconselha-se ao leitor que para sua melhor clareza, leia o texto que descreve a função virg-1 e acompanhe ao mesmo tempo o seu fluxograma. A figura 5.1 que se encontra nas páginas 572 a 574 do 2º volume deste trabalho, ilustra o fluxograma da função virg-1. Nesta figura, alguns comentários que se encontram no texto não constam do fluxograma. Devido à extensão do fluxograma ele foi dividido em 2 partes.

O programa verifica se o caractere reconhecido como número 1 é uma vírgula, varrendo a matriz-original da esquerda para a direita, a uma altura correspondente a 60% da altura do caractere. A matriz-original é varrida da esquerda para a direita, até que o primeiro "pixel-on" seja encontrado. A partir deste "pixel-on", a matriz-original é varrida para baixo, até que não existam mais "pixels-on" que possam ser varridos nessa direção. A matriz-original armazena os

"pixels-on" e os "pixels-off" que constituem o esqueleto da imagem de um caractere.

Ao varrer a matriz-original para baixo, o programa armazena o número de "pixels-on" varridos e o número de "pixels-on" varridos que se encontram nas direções 4 ou 5, conforme o código de cadeia de 8 direções, ilustrado na figura 3.10 do capítulo 3 deste trabalho. A contagem do número de "pixels-on" varridos só é inicializada, após ser encontrado o primeiro "pixel-on" que se encontra na direção 4 ou 5.

Se o número de "pixel-on" presentes na matriz-original é menor ou igual a 13 e ((o número de "pixels-on" varridos após ser encontrado o primeiro "pixel-on" que se encontra na direção 4 ou 5 é maior ou igual a 2) e (a porcentagem de "pixels-on" varridos que se encontram nas direções 4 e 5 é maior ou igual a 90% da quantidade de "pixels-on" varridos, quando a matriz-original foi varrida para baixo)), o caractere reconhecido anteriormente como número 1 é reconhecido como uma vírgula, sendo portanto eliminado da matriz-original. Caso contrário, o caractere reconhecido é realmente o número 1.

Elim-Virg

O objetivo da função elim-virg é varrer a matriz-original em toda a sua extensão, verificando pela quantidade de "pixels-on" varridos e pela altura em que se encontram estes "pixels-on", se a matriz-original possui um caractere que deve ser reconhecido ou possui uma vírgula, ruído ou ponto. A matriz-original armazena os "pixels-on" e os "pixels-off" que constituem o esqueleto da imagem de um caractere.

Para facilitar o entendimento da função elim-virg, optou-se por explicá-la no texto e esboçar o seu fluxograma. Aconselha-se ao leitor que para sua melhor clareza, leia o texto que descreve a função elim-virg e acompanhe ao mesmo

tempo o seu fluxograma. A figura 5.2 que se encontra nas páginas 575 a 586 do 2º volume deste trabalho, ilustra o fluxograma da função elim-virg. Nesta figura, alguns comentários que se encontram no texto não constam do fluxograma. Devido à extensão do fluxograma ele foi dividido em 11 partes.

Devido ao algoritmo de segmentação utilizado, na matriz-original pode estar contida a vírgula e/ou um caractere. Para que o programa proceda a retirada da vírgula, a matriz-original é varrida para cima, para baixo, para a esquerda ou para a direita conforme os "pixels-on" estejam dispostos, a partir da coordenada CILHI (coordenada inicial da linha horizontal inferior, a obtenção desta coordenada foi explicada no capítulo 4 deste trabalho). A matriz-original é varrida para cima, para baixo, para a esquerda ou para a direita, até que não existam mais "pixels-on" que possam ser varridos em nenhuma dessas direções.

Com o objetivo de varrer a matriz-original para cima, para baixo, para a esquerda ou para a direita, não foram utilizados os critérios das variáveis **"condição-vertical"** e **"condição-horizontal"**, explicados no capítulo 4 deste trabalho. Ao desconsiderar essas variáveis, o programa verifica a possibilidade de varrer a matriz-original em todas as direções possíveis, não importando se os "pixels-on" varridos encontram-se na direção horizontal ou na direção vertical.

Enquanto a matriz-original é varrida para cima, para baixo, para a esquerda ou para a direita, o programa armazena a quantidade de "pixels-on" varridos neste processo e a quantidade de "pixels-on" varridos que se encontram na direção 0. Se ((o "pixel-on" que está sendo varrido encontra-se na direção 0, 1 ou 7) e (o último "pixel-on" varrido encontra-se na direção 0)), o programa considera que o "pixel-on" que está sendo varrido encontra-se na direção 0 e portanto a variável que armazena a quantidade de "pixels-on" varridos que se encontram na direção 0 é incrementada de uma unidade.

Após terminar de varrer a matriz-original para cima, para baixo, para a esquerda ou para a direita, o programa verifica se ((a quantidade de "pixels-on"

varridos é menor ou igual a 40% do número total de "pixels-on" presentes na matriz-original) e (o "pixel-on" de maior ordenada na matriz-original encontra-se abaixo de 60% do número total de linhas dessa matriz)). Em caso afirmativo, os "pixels-on" que foram varridos na matriz-original são considerados como sendo uma vírgula e portanto são apagados dessa matriz. O processo de varrer a matriz-original para cima, para baixo, para a esquerda ou para a direita é finalizado.

Após eliminar a vírgula da matriz-original, restará nesta matriz somente os "pixels-on" e os "pixels-off" que constituem o esqueleto da imagem de um caractere que será reconhecido.

Se no processo de segmentação dos caracteres, a vírgula for segmentada, ela será armazenada na matriz-original e no processo de reconhecimento, será eliminada pelas funções verif-1-3-5-6-0, verif-0-8-6-3-5-1-2-4, verif-7-9-4-1-2-0-6-8-3 ou verif-1-4-5-6-7, como foi explicado no capítulo 4 deste trabalho.

Enquanto a matriz-original é varrida para cima, para baixo, para a esquerda ou para a direita, se o valor da variável **"hole"** é igual a 0, o programa anota se são encontradas as coordenadas CILHS, CFLHS e CFLHI e com o objetivo de encontrar os extremos do caractere, o programa anota se a ordenada do "pixel-off" que está sendo varrido é a maior ou a menor ordenada encontrada na matriz-original e se a abscissa desse "pixel-off" é a maior ou a menor abscissa encontrada nessa matriz. Se o valor da variável **"hole"** é diferente de 0, o programa iguala essa variável a 0, pois após varrer um "pixel-off", provavelmente será encontrado na matriz-original, um "pixel-on" que possa ser varrido. A variável **"hole"** indica se o "pixel" que está sendo varrido é "on" ou "off". O programa considera como extremo do caractere um ponto em que não existem mais "pixels-on" vizinhos ao "pixel-on" que está sendo varrido. A obtenção da coordenada CILHS (coordenada inicial da linha horizontal superior), da coordenada CFLHS (coordenada final da linha horizontal superior) e da coordenada CFLHI (coordenada final da linha horizontal inferior), foi explicada

no capítulo 4 deste trabalho.

Nos caracteres manuscritos, devido a resolução utilizada no "scanner" e a maneira como o caractere foi escrito, podem ocorrer imperfeições ou falhas. O programa considera, que uma falha é uma região do caractere em que os "pixels" se encontram apagados, ou seja, uma falha é uma região do carctere onde existem um ou mais "pixels-off". Se o número de "pixels-off" que constituem a falha for relevante, o caractere poderá ficar fragmentado, devido a isto com o objetivo de minimizar a ocorrência de falhas no caractere, a matriz-original é varrida em toda sua extensão e enquanto essa matriz-original é varrida, os procedimentos descritos a seguir são executados:

- Enquanto a matriz-original é varrida na direção horizontal, para a direita (direção 0) ou para a esquerda (direção 4), se o valor da variável **"condição-de-folga"** é diferente de 0, o programa não encontrou "pixel-on" que pudesse ser varrido na direção em que a matriz-original está sendo varrida, portanto o próximo "pixel" a ser varrido será um "pixel-off" que estará uma coluna a direita do último "pixel-on" varrido, se a matriz-original está sendo varrida para a direita, ou uma coluna a esquerda desse último "pixel-on", se a matriz-original está sendo varrida para a esquerda. Se o valor da variável **"condição-de-folga"** é diferente de 0, o programa verifica se o último "pixel-on" varrido antes do valor dessa variável ser diferente de 0 possui algum "pixel-on" vizinho. Em caso afirmativo, as análises descritas a seguir, são realizadas para o "pixel-on" vizinho ao último "pixel-on" varrido. Caso contrário, essas análises serão realizadas para o "pixel-off" que se encontra a direita ou a esquerda do último "pixel-on" varrido. Abaixo são descritas as análises realizadas pelo programa, para o "pixel-on" vizinho ao último "pixel-on" varrido ou para o "pixel-off" que se encontra a esquerda ou a direita do último "pixel-on" varrido:

- Se (a matriz-original é varrida para a direita e o "pixel-off" que está sendo varrido possui "pixel-on" vizinho que se encontra na direção 1, 0 ou 7) ou (a matriz-original é varrida para a esquerda e o "pixel-off" que está sendo varrido possui "pixel-on" vizinho que se encontra na direção 3, 4 ou 5), o programa verifica se ((o "pixel-off" que está sendo varrido possui 2 "pixels-on" vizinhos) ou (esse "pixel-off" não possui "pixels-on" vizinhos que se encontram nas direções ((2 e 6) e (3, 4 ou 5) e (1, 0 ou 7))). Em caso afirmativo, o "pixel-off" é substituído por um "pixel-on" e a variável **"condição-de-folga"** é igualada a 0. Caso contrário, a variável **"hole"** é igualada a 1 e a variável **"condição-de-folga"** é igualada a 0. A variável **hole é** igualada a 1, indicando que o "pixel-on" que está sendo varrido não deve ser considerado na análise dos extremos do caractere.

- Se ((a matriz-original é varrida para a direita e o "pixel" ("pixel-on" vizinho ao último "pixel-on" varrido ou "pixel-off" que se encontra a direita ou a esquerda do último "pixel-on" varrido) que está sendo varrido possui "pixel-on" vizinho que se encontra na direção 3, 4 ou 5) ou (a matriz-original é varrida para a esquerda e o "pixel" que está sendo varrido possui "pixel-on" vizinho que se encontra na direção 1, 0 ou 7)), a variável **"condição-de-folga"** é igualada a 0 e o programa continua a varrer a matriz-original, porém em direção contrária a direção que essa matriz estava sendo varrida.

- Se ((a matriz-original é varrida na direção vertical, para cima (direção 2) ou para baixo (direção 6)) e (o valor da variável **"condição-de-folga"** é diferente de 0)), são realizados os procedimentos descritos a seguir:

- Se (((a matriz-original é varrida para cima e o "pixel-off" (valor da variável **"condição-de-folga"** diferente de 0) que está sendo varrido possui "pixel-on" vizinho que se encontra na direção 1, 2 ou 3) ou (a matriz-original é varrida para baixo e o "pixel-off" que está sendo varrido possui "pixel-on" vizinho que se encontra na direção 5, 6 ou 7)) e (esse "pixel-off" não possui "pixel-on" vizinho que se encontra na direção 0 ou 4)), o programa verifica se ((o valor da variável **npix** (definida no capítulo 4 deste trabalho) é menor ou igual a 40% do número total de "pixels-on" presentes na matriz-original) e (o valor da ordenada do "pixel-off" que está sendo varrido é menor do que 60% do número total de linhas da matriz-original) e (não foi encontrado nenhum ponto de cruzamento, enquanto o programa varre a matriz-original para cima, para baixo, para a esquerda ou para a direita)). Em caso afirmativo, os "pixels-on" que foram varridos na matriz-original são retirados da imagem, pois os mesmos representam uma vírgula. Caso contrário, o "pixel-off" que está sendo varrido é substituído por um "pixel-on", pois o mesmo representa uma falha no caractere.

- Se o "pixel" (este "pixel" pode ser "on" ou "off") que se encontra uma coluna a direita do "pixel" que está sendo varrido possui um "pixel-on" vizinho que se encontra na direção 0, 1 ou 7, a variável **"condição-de-folga"** é igualada a 0, as variáveis **"condição-horizontal"** e **"hole"** são igualadas a 1 e quando o programa voltar a varrer a matriz-original na direção horizontal, esta matriz será varrida para a direita (direção 0, 1 ou 7).

- Se o "pixel" (este "pixel" pode ser "on" ou "off") que se encontra uma coluna a esquerda do "pixel" que está sendo varrido possui um

"pixel-on" vizinho que se encontra na direção 3, 4 ou 5, a variável **"condição-de-folga"** é igualada a 0, as variáveis **"condição-horizontal"** e **"hole"** são igualadas a 1 e quando o programa voltar a varrer a matriz-original na direção horizontal, esta matriz será varrida para a esquerda (direção 3, 4 ou 5).

No processo de varrer a matriz-original para cima, para baixo, para a esquerda ou para a direita, se ao varrer essa matriz na direção vertical o valor da variável **"condição-horizontal"** for diferente de 0, o "pixel-on" que está sendo varrido encontra-se na direção horizontal (direção 0 ou 4) e por este motivo, o programa volta a varrer a matriz-original na direção horizontal (direção 4 ou 0), porém em sentido contrário ao último sentido que essa matriz estava sendo varrida na direção horizontal. Se porém, o programa está varrendo a matriz-original na direção horizontal e o valor da variável **"condição-horizontal"** for diferente de 0, o "pixel-on" que está sendo varrido encontra-se na direção vertical (direção 2 ou 6) e por este motivo, o programa volta a varrer a matriz-original na direção vertical (direção 6 ou 2), porém em sentido contrário ao último sentido que essa matriz estava sendo varrida na direção vertical.

Enquanto a matriz-original é varrida na direção vertical ou na direção horizontal, conforme explicado no parágrafo anterior, se ((é encontrado um ponto de cruzamento) ou (o "pixel-on" que está sendo varrido representa um ponto de cruzamento que foi desconsiderado na função vertical ou na função horizontal)), o programa armazena a coordenada desse ponto de cruzamento, pois após terminar de varrer a matriz-original na direção vertical ou na direção horizontal, o programa retorna a esse ponto de cruzamento e tenta varrer a matriz-original em outras direções. Ao iniciar o processo de varrer a matriz-original na direção vertical ou na direção horizontal, a coordenada CILHI é armazenada como sendo a coordenada de um ponto de cruzamento. Isto foi feito, porque em alguns

caracteres a coordenada CILHI encontra-se em alguma região que não é o extremo do caractere e o programa ao varrer a matriz-original em uma determinada direção, não varre os "pixels-on" que se encontram em direções opostas a direção que está sendo varrida, como resultado deste processo, partes do caractere não são varridas se a coordenada CILHI não for armazenada como sendo a coordenada de um ponto de cruzamento.

Foi observado que em alguns casos o número 5 estava fragmentado, isto é, sua parte superior estava desconectada da sua parte inferior. Neste caso, sua parte inferior era reconhecida como número 0 ou 1, enquanto a sua parte superior era reconhecida como número 1. Com o objetivo de eliminar esse erro, quando for encontrado um caractere cuja quantidade de "pixels-on" varridos que se encontram na direção 0 é maior ou igual a 80% da quantidade de "pixels-on" varridos, a função elim-virg envia para o programa uma mensagem indicando esse fato. Ao receber essa mensagem, o programa verifica se o último caractere reconhecido (parte inferior do número 5) é o número 0 ou 1. Em caso afirmativo, o caractere anteriormente reconhecido como número 0 ou 1, é reconhecido como número 5.

5.2.2 - Eliminação do Ponto e/ou Ruído

Na matriz-imagem, o ponto colocado em quantias numéricas com valor igual ou superior a 1000, e o ruído que pode aparecer na imagem, após o processo de afilamento da imagem são reduzidos a mais ou menos 2 "pixels-on". Devido a isto, a análise para a eliminação do ponto e do ruído é realizada pelo mesmo procedimento do algoritmo de reconhecimento. A função que elimina o ponto e/ou ruído é denominada no programa de elim-ponto-ruído. A matriz-imagem armazena os "pixels-on" e os "pixels-off" que constituem o esqueleto da imagem da área reservada a escrita do valor numérico de um cheque bancário.

Para facilitar o entendimento do processo de eliminação do ponto e/ou ruído, optou-se por explicá-lo no texto e esboçar o seu fluxograma. Aconselha-se ao leitor que para sua melhor clareza, leia o texto que descreve o processo de eliminação do ponto e/ou ruído e acompanhe ao mesmo tempo o seu fluxograma. A figura 5.3 que se encontra nas páginas 587 a 592 do 2º volume deste trabalho, ilustra o fluxograma do processo de eliminação do ponto e/ou ruído. Nesta figura, alguns comentários que se encontram no texto não constam do fluxograma. Devido à extensão do fluxograma ele foi dividido em 5 partes.

Abaixo, são descritos os procedimentos utilizados no programa, no processo de eliminação do ponto e/ou ruído:

- Enquanto a matriz-original é varrida para a direita, na delimitação das coordenadas inicial e final da linha horizontal superior e depois na delimitação das coordenadas inicial e final da linha horizontal inferior, como explicado no capítulo 4 e independente de qual dessas linha está sendo delimitada, o programa armazena na variável **npix-lr** a quantidade de "pixels-on" varridos para a direita, que se encontram na direção horizontal e na variável **tot-vert** armazena a quantidade de "pixels-on" varridos que se encontram na direção vertical. No caso de ocorrer falha no caractere, como explicado na função elim-virg, neste capítulo, o "pixel-off" que compõe essa falha é denominado no programa de "pixel-falha". Se após encontrar um "pixel-falha", o programa encontrar um "pixel-on", a coordenada do "pixel-falha" é armazenada e a quantidade de "pixels-on" que são varridos após a falha é armazenada na variável **npix-rl**. Independente da definição utilizada para a variável **npix-rl** no capítulo 4 deste trabalho, nesse processo essa variável armazena a quantidade de "pixels-on" varridos após ser encontrado um "pixel-falha".

- Se não existirem mais "pixels-on" a serem varridos na direção em que a

matriz-original está sendo varrida, o programa armazena na variável **LC** a coordenada do último "pixel-on" varrido e verifica se ((o valor da variável **npix-rl** está entre 0 e 2, inclusive) ou (o valor da variável **tot-vert** é menor ou igual a 2, o valor da variável **npix-lr** é igual a 1, (o número de "pixels-on" vizinhos ao último "pixel-falha" cuja coordenada foi armazenada ou ao último "pixel-on" varrido na direção vertical é diferente de 0 (o programa analisa os "pixels-on" vizinhos, do último entre esses 2 "pixels" ("pixel-falha" ou "pixel-on" varrido) que foi encontrado)) e (o valor da variável **npix-rl** é menor ou igual a 1))) (esta condição será referenciada adiante como CONDIÇÃO 1). Em caso afirmativo, se o valor da variável **"condição-vertical"** é igual a 10 (o objetivo da utilização deste valor será explicado no próximo parágrafo), o programa varre a matriz-original para a esquerda, a partir da coordenada armazenada na variável **LC**. A matriz-original é varrida para a esquerda, até que não existam mais "pixels-on" que possam ser varridos nessa direção. Ao terminar de varrer a matriz-original para a esquerda, os "pixels-on" varridos nessa direção são apagados dessa matriz, por serem considerados pelo programa como ponto ou ruído. A matriz-original armazena os "pixels-on" e os "pixels-off" do esqueleto da imagem de um caractere.

Se o valor da variável **"condição-vertical"** é diferente de 10, o programa iguala essa variável a 10 e verifica se o "pixel-on" cuja coordenada foi armazenada, quando o valor da variável **"condição-vertical"** era maior ou igual a 3, possui pelo menos um "pixel-on" vizinho. Em caso afirmativo, o programa verifica se as coordenadas inicial e final da linha horizontal superior estão sendo delimitadas. Se isto ocorrer, o programa retorna ao número 18 do fluxograma que realiza a delimitação das coordenadas inicial e final a linha horizontal superior, quando a matriz-original é varrida da esquerda para a direita. Esse número encontra-se na página 373 da figura 4.1(j) do capítulo 4 deste trabalho. Caso

contrário, o programa retorna ao número 17* do fluxograma que realiza a delimitação das coordenadas inicial e final a linha horizontal inferior, quando a matriz-original é varrida da esquerda para a direita. Esse número encontra-se na página 397 da figura 4.4(j) do capítulo 4 deste trabalho. Se o "pixel-on" cuja coordenada foi armazenada, quando o valor da variável **"condição-vertical"** era maior ou igual a 3, não possui "pixels-on" vizinhos, o programa verifica se as coordenadas inicial e final da linha horizontal superior estão sendo delimitadas. Se isto ocorrer, o programa retorna ao número 21* do fluxograma que realiza a delimitação das coordenadas inicial e final a linha horizontal superior, quando a matriz-original é varrida da esquerda para a direita. Esse número encontra-se na página 369 da figura 4.1(f) do capítulo 4 deste trabalho. Caso contrário, o programa retorna ao número 22* do fluxograma que realiza a delimitação das coordenadas inicial e final a linha horizontal inferior, quando a matriz-original é varrida da esquerda para a direita. Esse número encontra-se na página 393 da figura 4.4(f) do capítulo 4 deste trabalho.

Se a CONDIÇÃO 1 não foi satisfeita, o programa verifica se o "pixel-falha" cuja coordenada foi armazenada deve ser substituído por um "pixel-on". Nesta verificação, se ((o valor da variável **npix-rl** é maior ou igual a 2) e (o número de "pixels-on" vizinhos ao "pixel-falha" é igual a 2)), o programa substitui o "pixel-falha" por um "pixel-on". A variável **"condição-vertical"** é igualada a 10, com o objetivo de possibilitar que os "pixels-on" que não foram varridos, quando a matriz-original foi varrida para a direita, possam ser varridos nessa direção e deste modo impedir que partes do caractere sejam consideradas pelo programa como ruído.

5.3 - SEGMENTAÇÃO DE CARACTERES SOBREPOSTOS

No capítulo 3 foi explicado que o texto a ser reconhecido é digitalizado e

gravado. Após este processo, o texto digitalizado é lido do arquivo em que foi gravado e suas linhas são segmentadas. Neste trabalho foi considerado que a área reservada à escrita da quantia numérica constitui apenas uma linha de texto.

Para facilitar o entendimento de todo o processo utilizado no reconhecimento de caracteres manuscritos na área reservada à escrita da quantia numérica de cheques bancários, optou-se por explicá-lo no texto e esboçar o seu fluxograma. Aconselha-se ao leitor que para sua melhor clareza, leia o texto que descreve todo o processo utilizado no reconhecimento de caracteres manuscritos e acompanhe ao mesmo tempo o seu fluxograma. A figura 5.4 que se encontra nas páginas 593 a 603 do 2º volume deste trabalho, ilustra o fluxograma de todo o processo de reconhecimento de caracteres manuscritos na área reservada à escrita da quantia numérica de cheques bancários. Nesta figura, alguns comentários que se encontram no texto não constam do fluxograma. Devido à extensão do fluxograma ele foi dividido em 10 partes.

No reconhecimento de caracteres manuscritos, optou-se por realizar o afilamento da área reservada a escrita da quantia numérica antes de segmentar os caracteres contidos nessa área. Este procedimento foi adotado com o objetivo de calcular a largura média dos caracteres e facilitar o processo de segmentação dos caracteres conectados. Uma observação importante a ser feita é que no cálculo da largura média dos caracteres é necessário que estes caracteres estejam com espessura de um "pixel-on". Por este motivo, o afilamento dos caracteres da área reservada à escrita da quantia numérica é realizado antes do cálculo da largura média.

Antes do programa realizar o cálculo da largura média dos caracteres, são executadas as funções que realizam a verificação e a eliminação do campo que envolve a quantia numérica de um cheque bancário, caso este campo esteja presente na área reservada à escrita dessa quantia. Essas funções serão explicadas adiante no item 5.5 deste capítulo.

Com o objetivo de calcular a largura média dos caracteres, o programa realiza a segmentação dos caracteres contidos na matriz-imagem, varrendo esta matriz da esquerda para a direita e separando os "pixels" (estes "pixels" podem ser "on" ou "off") em conjuntos à medida que uma ou mais colunas de "pixels-off" são encontradas. O conjunto de "pixels" separados da matriz-imagem é armazenado na matriz-original. Para cada conjunto separado, o programa calcula sua largura e adiciona esta largura a largura dos outros conjuntos de "pixels" separados. Após terminar de separar todos os conjuntos presentes na matriz-imagem, a largura média dos caracteres será a soma da largura dos conjuntos de "pixels" separados dividido pelo número destes conjuntos. Cada conjunto de "pixels" separado pode conter um caractere isolado, caracteres sobrepostos ou caracteres conectados. A matriz-imagem armazena os "pixels-on" e os "pixels-off" que constituem o esqueleto da imagem da área reservada à escrita da quantia numérica de um cheque bancário e a matriz-original armazena os "pixels-on" e os "pixels-off" dos esqueletos das imagens dos caracteres.

À medida que a largura média dos caracteres é calculada, se ((a coluna inicial do caractere está entre a coluna inicial e a 35ª coluna da matriz-imagem) ou (a coluna inicial do caractere é maior ou igual a 75% do número total de colunas da matriz-imagem)), o programa executa a função verifica-cerquilha, com o objetivo de eliminar as cerquilhas que estejam presentes na matriz-imagem. A função verifica-cerquilha será explicada adiante no item 5.6 deste capítulo.

Se a coluna inicial do caractere é maior ou igual a 75% do número total de colunas da matriz-imagem, o programa varre a área compreendida entre ((a coordenada formada pela coluna inicial do caractere decrementada de 5 unidades e a linha inicial da matriz-imagem) e (a coordenada formada pela coluna inicial do caractere incrementada de 5 unidades e a linha final da matriz-imagem)). Se o número de "pixels-on" que constituem a área varrida é maior ou igual a 1 e menor ou igual a 15, o programa considera que esses "pixels-on" formam um ruído, e

portanto devem, ser apagados da matriz-imagem.

Se o caractere cuja largura foi calculada é o primeiro caractere da quantia numérica do cheque bancário e não é uma cerquilha, o programa verifica se ((a coluna final desse caractere está localizada a esquerda da 50ª coluna da matriz-imagem) e (o caractere 'R$' manuscrito, ainda não foi reconhecido)). Em caso afirmativo, o programa executa a função verifica-real, com o objetivo de verificar se o caractere cuja largura foi calculada é o 'R' do 'R$' manuscrito. A função verifica-real será explicada no item 5.7 deste capítulo.

Segmentado o caractere, é varrida sua matriz-original, com o objetivo de delimitar como explicado no capítulo 4 deste trabalho, as coordenadas inicial e final da linha horizontal superior (CILHS e CFLHS) e as coordenadas inicial e final da linha horizontal inferior (CILHI e CFLHI) (este passo será referenciado adiante como PASSO 1). No capítulo 4 foi citado que neste ponto do programa era delimitada as coordenadas inicial e final da linha vertical esquerda (CILVE e CFLVE). Como na área reservada à escrita da quantia numérica do cheque, podem ocorrer caracteres isolados, sobrepostos e conectados, as coordenadas CILVE e CFLVE só serão encontradas após o caractere estar isolado.

Após realizar o PASSO 1, com o objetivo de eliminar a vírgula, se ela estiver presente na matriz-original e segmentar os caracteres que se encontram sobrepostos, o programa executa a função elim-virg, explicada no item 5.2.1 deste capítulo. Ao executar a função elim-virg, o programa varre a matriz-original a partir da coordenada CILHI. Ao varrer a matriz-original, o programa varre o esqueleto da imagem do caractere em toda a sua extensão (este passo será referenciado adiante como PASSO 2). Durante o PASSO 2, cada "pixel-on" e "pixel-off" varrido é armazenado em uma nova-matriz e é apagado da matriz-original. Ao terminar de varrer o esqueleto da imagem do caractere em toda a sua extensão, a nova-matriz contém a cópia do esqueleto da imagem do caractere varrido.

Se ao terminar de varrer o esqueleto da imagem do caractere em toda a sua extensão no PASSO 2, o programa verificar que as coordenadas CFLHI, CILHS e CFLHS, encontradas na matriz-original coincidem com as coordenadas CFLHI, CILHS e CFLHS encontradas no PASSO 1 (este passo será referenciado adiante como PASSO 3), o programa verifica se a largura do esqueleto da imagem do caractere contido na nova-matriz é maior ou igual a 110% da largura média dos caracteres, que compõem o esqueleto da imagem da área reservada à escrita da quantia numérica de um cheque bancário (este passo será referenciado adiante como PASSO 4). Em caso afirmativo, a nova-matriz contém o esqueleto da imagem de um caractere isolado ou o esqueleto da imagem de caracteres conectados. A verificação se a nova-matriz contém o esqueleto da imagem de um caractere isolado ou o esqueleto da imagem de caracteres conectados é realizada executando a função verif-conec-isolado, explicada no item 5.4.1 deste capítulo. Ao terminar de executar a função verif-conec-isolado, se for verificado que a nova-matriz contém o esqueleto da imagem de caracteres conectados, a função elim-conec é executada, com o objetivo de desconectar esses caracteres. A função elim-conec será explicada no item 5.4.2 deste capítulo. Ao desconectar um caractere na função elim-conec, o programa armazena na matriz-final a imagem do esqueleto desse caractere e o restante dos caracteres conectados permanecem na nova-matriz. Com o objetivo de delimitar a linha horizontal superior e a linha horizontal inferior do esqueleto da imagem do caractere que se encontra na matriz-final, o programa executa os PASSOS de 1 a 4.

Se no PASSO 4, a largura do esqueleto da imagem do caractere contido na nova-matriz é menor do que 110% da largura média dos caracteres, que compõem o esqueleto da imagem da área reservada à escrita da quantia numérica de um cheque bancário ou é verificado que o caractere é isolado, o programa verifica se ((o valor da abscissa do "pixel-on" que se encontra no extremo direito do esqueleto da imagem do caractere é menor ou igual a 15% do número total de

colunas da matriz-imagem) e (o valor da ordenada do "pixel-on" que se encontra no extremo superior do esqueleto da imagem do caractere é menor ou igual a 40% do número total de linhas da matriz-imagem) e (o número total de "pixels-on" da matriz-final é maior ou igual a 5 e menor ou igual a 20)). Em caso afirmativo, o esqueleto da imagem do caractere contido na matriz-final é considerado como sendo o esqueleto do 'R$' impresso. A verificação se a imagem desse caractere é a imagem do 'R$' impresso e a sua eliminação são explicadas no item 5.7.2 deste capítulo. Caso contrário, a matriz-final contém o esqueleto da imagem de um caractere isolado e portanto, o programa passa para a fase de reconhecimento, onde são executadas as funções verif-1-3-5-6-0, verif-0-8-6-3-5-1-2-4, verif-2-3-5-7-9-8-0, verif-7-9-4-1-2-0 ou verif-1-4-5-6-7, como explicado no capítulo 4. Ao executar estas funções, o caractere poderá ser reconhecido, ou não, pelo programa.

Se ao terminar de varrer o esqueleto da imagem do caractere em toda a sua extensão no PASSO 2, o PASSO 3 não é satisfeito, o programa repete os PASSOS 1 e 2 e prossegue as verificações dos PASSOS 3 e 4.

Após reconhecer o caractere contido na matriz-final (esqueleto da imagem de um caractere), o programa verifica se na nova-matriz ainda existem "pixels-on". Em caso afirmativo, ainda existe na nova-matriz um ou mais caracteres (esqueleto da imagem de um ou mais caracteres), que devem ser reconhecidos. Portanto, o programa repete os PASSOS de 1 a 4 e o reconhecimento dos caracteres, até que não existam mais "pixels-on" na nova-matriz. É importante ressaltar que independente do número de caracteres presentes na nova-matriz e destes caracteres estarem sobrepostos ou conectados, o programa consegue segmentá-los e reconhecê-los.

Após reconhecer o caractere numérico, o programa anota em que posição este caractere encontra-se na matriz-imagem. Este procedimento visa posicionar os dígitos na ordem em que foram escritos, com o objetivo de que o programa

possa realizar a escrita do extenso da quantia numérica reconhecida de forma correta.

Ao terminar de reconhecer todos os caracteres presentes na nova-matriz, o programa verifica se na matriz-imagem ainda existem "pixels-on". Em caso afirmativo, o programa continua o processo de reconhecimento dos caracteres presentes na matriz-imagem. Caso contrário, esse processo é finalizado.

Como resultado do processo de reconhecimento, o programa apresenta: o nome do arquivo cuja quantia numérica foi reconhecida; o conteúdo desse arquivo, isto é, a área do cheque que foi digitalizada e gravada; a quantia numérica reconhecida pelo programa e seu extenso. Todas estas informações são apresentadas ao usuário em uma janela de saída, que será mostrada no item 5.8 deste capítulo. Ao apresentar a quantia numérica reconhecida, o programa insere nesta quantia a vírgula e os pontos necessários, pois estes foram retirados da matriz-imagem durante o processo de reconhecimento.

Após apresentar os resultados obtidos, o programa solicita ao usuário que o informe se existe ou não outro arquivo a ser reconhecido.

Se no processo de reconhecimento, o programa não conseguir reconhecer algum dos dígitos que compõem a quantia numérica do cheque, é apresentado ao usuário na janela de saída, na posição do dígito não reconhecido o símbolo "?". Além de apresentar este símbolo na posição do dígito não reconhecido, o programa informa ao usuário na janela de saída o significado desse símbolo.

A figura 5.5 ilustra alguns dos números sobrepostos que foram segmentados e reconhecidos pelo algoritmo.

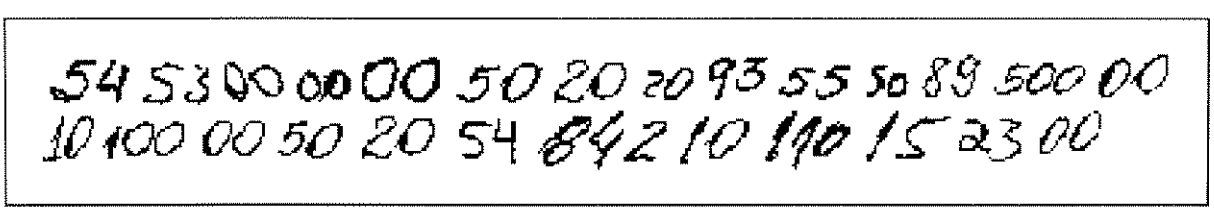


Figura 5.5 - Números sobrepostos segmentados e reconhecidos.

5.4 - VERIFICAÇÃO SE O CARACTERE É CONECTADO OU ISOLADO, CLASSIFICAÇÃO E DESCONEXÃO DOS CARACTERES CONECTADOS

O programa verifica se o caractere é conectado ou isolado e para os caracteres conectados, classifica o modo como eles devem ser desconectados e, após isto, realiza a desconexão desses caracteres. O processo que verifica se o caractere é conectado ou isolado é descrito no item 5.4.1 e o processo que classifica e realiza a desconexão dos caracteres conectados é explicado no item 5.4.2 deste capítulo.

5.4.1 - Verificação se o Caractere é Conectado ou Isolado

O processo que verifica se o caractere é conectado ou isolado é realizado pela função denominada no programa de verif-conec-isolado.

Verif-conec-isolado

Para facilitar o entendimento do processo utilizado na função verif-conec-isolado, optou-se por explicá-lo no texto e esboçar o seu fluxograma. Aconselha-se ao leitor que para sua melhor clareza, leia o texto que descreve a função verif-conec-isolado e acompanhe ao mesmo tempo o seu fluxograma. A figura 5.6 que se encontra nas páginas 604 a 616 do 2º volume deste trabalho, ilustra o fluxograma da função verif-conec-isolado. Nesta figura, alguns comentários que se encontram no texto não constam do fluxograma. Devido à extensão do fluxograma ele foi dividido em 12 partes.

Os procedimentos realizados no programa pela função verif-conec-isolado são descritos a seguir:

- O programa varre a nova-matriz da esquerda para a direita e de cima para baixo, até 70% da altura do conjunto formado pelos caracteres contidos nessa matriz. A nova-matriz é varrida da esquerda para a direita e de cima para baixo, até que um "pixel-on" seja encontrado ou seja alcançada a coluna que antecede a coluna central dessa matriz. Enquanto a nova-matriz é varrida da esquerda para a direita e de cima para baixo, se é encontrado um "pixel-on", o programa armazena sua coordenada denominando-a de CILHSVD (coordenada inicial da linha horizontal superior varrida para a direita). A coordenada CILHSVD (ACILHSVD, OCILHSVD) consiste de uma abscissa denominada de abscissa da coordenada inicial da linha horizontal superior varrida para a direita (ACILHSVD) e de uma ordenada denominada de ordenada da coordenada inicial da linha horizontal superior varrida para a direita (OCILHSVD). O mesmo processo é utilizado para varrer a nova-matriz de cima para baixo, porém da direita para a esquerda. Ao encontrar um "pixel-on", o programa armazena sua coordenada denominando-a de CILHSVE (coordenada inicial da linha horizontal superior varrida para a esquerda). A coordenada CILHSVE (ACILHSVE, OCILHSVE) consiste de uma abscissa denominada de abscissa da coordenada inicial da linha horizontal superior varrida para a esquerda (ACILHSVE) e de uma ordenada denominada de ordenada da coordenada inicial da linha horizontal superior varrida para a esquerda (OCILHSVE). A nova-matriz armazena os "pixels-on" e os "pixels-off" do esqueleto da imagem de um caractere isolado ou de caracteres conectados.

- O item anterior é novamente realizado, com a variante que agora a nova-matriz é varrida de baixo para cima até 30% da altura do conjunto formado pelos caracteres contidos nessa matriz. Neste caso, as coordenadas encontradas são CILHIVD (coordenada inicial da linha horizontal inferior varrida para a

direita) e CILHIVE (coordenada inicial da linha horizontal inferior varrida para a esquerda). A coordenada CILHIVD (ACILHIVD, OCILHIVD) consiste de uma abscissa denominada de abscissa da coordenada inicial da linha horizontal inferior varrida para a direita (ACILHIVD) e de uma ordenada denominada de ordenada da coordenada inicial da linha horizontal inferior varrida para a direita (OCILHIVD). A coordenada CILHIVE (ACILHIVE, OCILHIVE) consiste de uma abscissa denominada de abscissa da coordenada inicial da linha horizontal inferior varrida para a esquerda (ACILHIVE) e de uma ordenada denominada de ordenada da coordenada inicial da linha horizontal inferior varrida para a esquerda (OCILHIVE).

- Se foi encontrada a coordenada CILHSVD, é executada a função denominada no programa de horiz-conec, que recebe os parâmetros denominados no programa de "sup" (superior) e "right" (direita), que indicam respectivamente, que a linha horizontal superior está sendo varrida da esquerda para a direita. A função horiz-conec será explicada adiante neste capítulo. Ao executar a função horiz-conec, ela retorna para a função verif-conec-isolado a coordenada CFLHSVD (coordenada final da linha horizontal superior varrida para a direita). A coordenada CFLHSVD (ACFLHSVD, OCFLHSVD) consiste de uma abscissa denominada de abscissa da coordenada final da linha horizontal superior varrida para a direita (ACFLHSVD) e de uma ordenada denominada de ordenada da coordenada final da linha horizontal superior varrida para a direita (OCFLHSVD). A função horiz-conec tem como objetivo encontrar a coordenada final das linhas horizontais superior e inferior dos caracteres conectados e os pontos de cruzamento que se encontram na faixa compreendida entre 80% a 140% da largura do conjunto formado pelos caracteres contidos na nova-matriz. É anotado apenas um ponto de cruzamento para a linha horizontal superior e um ponto de cruzamento para a linha

horizontal inferior, pois a partir desses pontos de cruzamento é possível estabelecer o trecho comum aos caracteres conectados.

- Se foi encontrada a coordenada CILHSVE, é executada a função horiz-conec, que recebe os parâmetros denominados no programa de "sup" (superior) e "left" (esquerda), que indicam respectivamente, que a linha horizontal superior está sendo varrida da direita para a esquerda. Ao executar a função horiz-conec, ela retorna para a função verif-conec-isolado a coordenada CFLHSVE (coordenada final da linha horizontal superior varrida para a esquerda). A coordenada CFLHSVE (ACFLHSVE, OCFLHSVE) consiste de uma abscissa denominada de abscissa da coordenada final da linha horizontal superior varrida para a esquerda (ACFLHSVE) e de uma ordenada denominada de ordenada da coordenada final da linha horizontal superior varrida para a esquerda (OCFLHSVE).

- Se foi encontrada a coordenada CILHIVD, é executada a função horiz-conec, que recebe os parâmetros "inf" (inferior) e "right" (direita), que indicam respectivamente, que a linha horizontal inferior está sendo varrida da esquerda para a direita. Ao executar a função horiz-conec, ela retorna para a função verif-conec-isolado a coordenada CFLHIVD (coordenada final da linha horizontal inferior varrida para a direita). A coordenada CFLHIVD (ACFLHIVD, OCFLHIVD) consiste de uma abscissa denominada de abscissa da coordenada final da linha horizontal inferior varrida para a direita (ACFLHIVD) e de uma ordenada denominada de ordenada da coordenada final da linha horizontal inferior varrida para a direita (OCFLHIVD).

- Se foi encontrada a coordenada CILHIVE, é executada a função horiz-conec, que recebe os parâmetros "inf" (inferior) e "left" (esquerda), que indicam

respectivamente que a linha horizontal inferior está sendo varrida da direita para a esquerda. Ao executar a função horiz-conec, ela retorna para a função verif-conec-isolado a coordenada CFLHIVE (coordenada final da linha horizontal inferior varrida para a esquerda). A coordenada CFLHIVE (ACFLHIVE, OCFLHIVE) consiste de uma abscissa denominada de abscissa da coordenada final da linha horizontal inferior varrida para a esquerda (ACFLHIVE) e de uma ordenada denominada de ordenada da coordenada final da linha horizontal inferior varrida para a esquerda (OCFLHSVE).

As coordenadas CILHSVD, CILHSVE, CFLHSVD, CFLHSVE, CILHIVD, CILHIVE, CFLHIVD e CFLHIVE não têm nenhuma relação com as coordenadas CILHS, CFLHS, CILHI e CFLHI encontradas, respectivamente, na delimitação da linha horizontal superior e inferior de um caractere isolado.

- Se ((não foi encontrada a coordenada CILHSVD ou a coordenada CILHSVE) ou (o valor absoluto da diferença entre as abscissas ACILHSVD e ACFLHSVE é menor ou igual a 4 e o valor absoluto da diferença entre as abscissas ACFLHSVD e ACILHSVE é menor ou igual a 4)), a variável denominada no programa de **supe** é igualada a -1, indicando que provavelmente o caractere possui apenas uma linha horizontal superior. A variável **supe** indica que foi possível delimitar no caractere uma ou duas linhas horizontais superiores.

- Se ((não foi encontrada a coordenada CILHIVD ou a coordenada CILHIVE) ou (o valor absoluto da diferença entre as abscissas ACILHIVD e ACFLHIVE é menor ou igual a 4 e o valor absoluto da diferença entre as abscissas ACFLHIVD e ACILHIVE é menor ou igual a 4 e o valor absoluto da diferença entre as ordenadas OCFLHIVD e OCILHIVE é menor ou igual a 3)), a variável denominada no programa de **infe** é igualada a -1, indicando que

provavelmente o caractere possui apenas uma linha horizontal inferior. A variável **infe** indica que foi possível delimitar no caractere uma ou duas linhas horizontais inferiores.

- Se os valores das variáveis **supe** e **infe** são iguais a -1, a variável denominada no programa de **si** é igualada a 1, indicando que provavelmente o caractere possui uma linha horizontal superior e uma linha horizontal inferior. Porém, se os valores dessas variáveis são diferentes de -1, a variável **si** é igualada a 4, indicando que provavelmente o caractere possui duas linhas horizontais superior e duas linhas horizontais inferior. Se apenas o valor da variável **infe** é igual a -1, a variável **si** é igualada a 2, indicando que provavelmente o caractere possui duas linhas horizontais superior e uma linha horizontal inferior. Se os valores das variáveis **supe** e **infe** não satisfazem a nenhuma dessas condições, a variável **si** é igualada a 3, indicando que provavelmente o caractere possui uma linha horizontal superior e duas linhas horizontais inferior. Através da variável **si** é possível determinar se o caractere que está presente na nova-matriz é um caractere isolado ou são caracteres conectados.

- Se quando a função horiz-conec foi executada, foi armazenada a coordenada de um ponto de cruzamento na linha horizontal superior e a coordenada de um ponto de cruzamento na linha horizontal inferior, a função denominada no programa de ver-conec-00 é executada, usando como parâmetros a largura média dos caracteres contidos na nova-matriz e a variável **si**. A função ver-conec-00 será explicada adiante neste capítulo. O objetivo da função ver-conec-00 é verificar se ao varrer a nova-matriz para baixo, a partir do ponto de cruzamento da linha horizontal superior encontrado pela função horiz-conec, é possível alcançar o ponto de cruzamento da linha horizontal inferior encontrado pela função horiz-conec. Ao executar a função ver-conec-00, a

nova-matriz é varrida para baixo, até que não existam mais "pixels-on" que possam ser varridos nessa direção ou seja encontrado o ponto de cruzamento da linha horizontal inferior, cuja coordenada foi armazenada. Enquanto a nova-matriz é varrida para baixo, se é encontrado o ponto de cruzamento da linha horizontal inferior, cuja coordenada foi armazenada, a variável **si** é igualada a 4. Caso contrário, a variável **si** é igualada a 1.

- Se ((o valor da variável **si** é igual a 2 ou 3) ou (o valor da variável **si** é igual a 4 e o valor absoluto da diferença entre as abscissas ACFLHSVD e ACILHSVD é menor ou igual a 2 e o valor absoluto da diferença entre as abscissas ACFLHSVE e ACILHSVE é menor ou igual a 2)), o programa verifica se o valor da variável **si** é igual a 2 ou 4 (este passo será referenciado adiante como PASSO 1). Em caso afirmativo, a função isolad-conec é executada. Esta função será explicada adiante neste capítulo. O objetivo da função isolad-conec é verificar se é possível nos caracteres cujo valor da variável **si** é igual a 2 ou 4, partir da coordenada CILHIVD, varrer a nova-matriz para cima e encontrar as coordenadas CILHSVD e CILHSVE (este passo será referenciado adiante como PASSO 2) e partir da coordenada CILHSVD, varrer a nova-matriz para a direita e encontrar a coordenada CILHSVE (este passo será referenciado adiante como PASSO 3). Se na função isolad-conec o PASSO 2 foi satisfeito e o PASSO 3 não foi satisfeito, a variável **si** é igualada a 1. Caso contrário, a variável **si** é igualada a 2. Se o PASSO 1 não foi satisfeito, o programa verifica se o valor da variável **si** é igual a 3. Em caso afirmativo, a função isolad-conec é executada com o objetivo de partir da coordenada CILHIVE, varrer a nova-matriz para cima, encontrar a coordenada CILHSVD e um extremo do caractere, denominado no programa de **extr**. O programa considera que existe um extremo no caractere, quando não for possível varrer mais nenhum "pixel-on" nesse caractere, na direção em que ele estava sendo

varrido. Após varrer a nova-matriz para cima, o programa retorna a coordenada CILHSVD e varre a nova-matriz novamente para cima. Ao varrer a nova-matriz para cima, o programa verifica se é possível encontrar o extremo extr. Em caso afirmativo, a variável **si** é igualada a 2. Caso contrário, a variável **si** é igualada a 1.

Ao terminar a execução da função verif-conec-isolado, esta função retorna à função que a chamou o valor da variável **si**, que indica se na nova-matriz existe um caractere isolado ou caracteres conectados.

As funções horiz-conec, ver-conec-00 e isolad-conec citadas acima, são explicadas a seguir e também são explicadas as funções verif-HS-HI e verif-pc-HS-HI.

Horiz-Conec

A função horiz-conec tem como objetivo encontrar a coordenada final das linhas horizontais superior e inferior dos caracteres conectados e os pontos de cruzamento que se encontram na faixa compreendida entre 80% e 140% da largura média dos caracteres contidos na nova-matriz. A largura média dos caracteres contidos na nova-matriz equivale a metade da largura dos caracteres contidos nessa matriz, isto é, a metade da diferença entre a coluna final e a coluna inicial dos caracteres contidos na nova-matriz. A nova-matriz armazena os "pixels-on" e os "pixels-off" do esqueleto da imagem de um caractere isolado ou o esqueleto da imagem de caracteres conectados.

Para facilitar o entendimento de todo o processo utilizado na função horiz-conec, optou-se por explicá-lo no texto e esboçar o seu fluxograma. Aconselha-se ao leitor que para sua melhor clareza, leia o texto que descreve a função horiz-conec e acompanhe ao mesmo tempo o seu fluxograma. A figura 5.7 que se

encontra nas páginas 617 a 631 do 2º volume deste trabalho, ilustra o fluxograma da função horiz-conec. Nesta figura, alguns comentários que se encontram no texto não constam do fluxograma. Devido à extensão do fluxograma ele foi dividido em 14 partes.

Se a função horiz-conec recebe os parâmetros "left" (esquerda) e "sup" (superior), a nova-matriz é varrida para a esquerda, a partir da coordenada CILHSVE (coordenada inicial da linha horizontal superior varrida para a esquerda). A nova-matriz é varrida para a esquerda, até que não existam mais "pixels-on" que possam ser varridos nessa direção. Ao encontrar na nova-matriz o último "pixel-on" varrido para a esquerda, o programa armazena sua coordenada denominando-a de CFLHSVE (coordenada final da linha horizontal superior varrida para a esquerda).

Se a função horiz-conec recebe os parâmetros "left" (esquerda) e "inf" (inferior), a nova-matriz é varrida para a esquerda, a partir da coordenada CILHIVE (coordenada inicial da linha horizontal inferior varrida para a esquerda). A nova-matriz é varrida para a esquerda, até que não existam mais "pixels-on" que possam ser varridos nessa direção. Ao encontrar na nova-matriz o último "pixel-on" varrido para a esquerda, o programa armazena sua coordenada denominando-a de CFLHIVE (coordenada final da linha horizontal inferior varrida para a esquerda).

Se a função horiz-conec recebe os parâmetros "right" (direita) e "sup" (superior), a nova-matriz é varrida para a direita, a partir da coordenada CILHSVD (coordenada inicial da linha horizontal superior varrida para a direita). A nova-matriz é varrida para a direita, até que não existam mais "pixels-on" que possam ser varridos nessa direção. Ao encontrar na nova-matriz o último "pixel-on" varrido para a direita, o programa armazena sua coordenada denominando-a de CFLHSVD (coordenada final da linha horizontal superior varrida para a direita).

Se a função horiz-conec recebe os parâmetros "right" (direita) e "inf" (inferior), a nova-matriz é varrida para a direita, a partir da coordenada CILHIVD (coordenada inicial da linha horizontal inferior varrida para a direita). A nova-matriz é varrida para a direita, até que não existam mais "pixels-on" que possam ser varridos nessa direção. Ao encontrar na nova-matriz o último "pixel-on" varrido para a direita, o programa armazena sua coordenada denominando-a de CFLHIVD (coordenada final da linha horizontal inferior varrida para a direita).

Se ((um dos parâmetros de entrada da função horiz-conec é "left" e a coordenada do "pixel-on" que está sendo varrido coincide com a coordenada CILHSVE ou CILHIVE) ou (um dos parâmetros de entrada da função horiz-conec é "right" e a coordenada do "pixel-on" que está sendo varrido coincide com a coordenada CILHSVD ou CILHIVD)), o programa verifica:

- Se não foi armazenado nas variáveis **LF** e **CF**, respectivamente, a ordenada e a abscissa de algum "pixel-on" varrido, o programa armazena nestas variáveis a ordenada e a abscissa do "pixel-on" que está sendo varrido.

- Se ((não foi encontrado ponto de cruzamento) e ((um dos parâmetros de entrada da função horiz-conec é "left" e a abscissa do "pixel-on" que está sendo varrido está acima de 80% da largura média dos caracteres contidos na nova-matriz) ou (um dos parâmetros de entrada da função horiz-conec é "right" e a abscissa do "pixel-on" que está sendo varrido está abaixo de 140% da largura média dos caracteres contidos na nova-matriz))), o programa continua a varrer a nova-matriz para a esquerda, se o parâmetro de entrada na função horiz-conec é "left", ou para a direita, se o parâmetro de entrada nessa função é "right". Caso contrário, o programa finaliza o processo de varrer a nova-matriz para a esquerda, se o parâmetro de entrada na função horiz-conec é "left", ou para a direita, se o parâmetro de entrada nessa função é "right".

Se os parâmetros de entrada na função horiz-conec são "inf" (inferior) e "left" (esquerda), enquanto a nova-matriz é varrida para a esquerda é encontrado ponto de cruzamento ou o "pixel-on" que está sendo varrido é o primeiro "pixel-on" a ser varrido na função horiz-conec e possui "pixels-on" vizinhos que se encontram nas direções (3 e 5) ou (4 e 5) ou (3 e 4), o programa armazena nas variáveis **L1** e **C1**, respectivamente, a ordenada e a abscissa do próximo "pixel-on" a ser varrido e realiza as análises descritas a seguir, para o "pixel-on" denominado no programa de *"pixel-analisado"*. Este "pixel-on" será o ponto de cruzamento encontrado ou o "pixel-on" que está sendo varrido se este é o primeiro "pixel-on" a ser varrido pela função horiz-conec. Caso contrário, se os parâmetros de entrada na função horiz-conec são "inf" (inferior) e "right" (direita), enquanto a nova-matriz é varrida para a direita é encontrado ponto de cruzamento ou o "pixel-on" que está sendo varrido é o primeiro "pixel-on" a ser varrido na função horiz-conec e possui "pixels-on" vizinhos que se encontram nas direções (1 e 0) ou (1 e 7) ou (0 e 7), o programa armazena nas variáveis **L1** e **C1**, respectivamente, a ordenada e a abscissa do próximo "pixel-on" a ser varrido e realiza as análises descritas a seguir para o *"pixel-analisado"*. As análises realizadas para o *"pixel-analisado"* são:

- Se um dos parâmetros de entrada da função horiz-conec é "inf" e já foi encontrado um ponto de cruzamento na linha horizontal inferior e um ponto de cruzamento na linha horizontal superior, o programa calcula o valor absoluto da diferença entre as abscissas do ponto de cruzamento encontrado na linha horizontal inferior e do ponto de cruzamento encontrado na linha horizontal superior e armazena o resultado deste cálculo na variável denominada de **d-pci-pcs**. O programa também calcula o valor absoluto da diferença entre a abscissa do *"pixel-analisado"* e a abscissa do ponto de cruzamento encontrado na linha horizontal superior e armazena o resultado deste cálculo na

variável denominada de **d-pca-pcs**. A variável **d-pci-pcs** armazena a distância entre as abscissas do ponto de cruzamento encontrado na linha horizontal inferior e do ponto de cruzamento encontrado na linha horizontal superior e a variável **d-pca-pcs** armazena a distância entre a abscissa do *"pixel-analisado"* e a abscissa do ponto de cruzamento encontrado na linha horizontal superior.

Se ((o valor contido na variável **d-pca-pcs** é menor do que o valor contido na variável **d-pci-pcs**) ou ((o valor contido na variável **d-pca-pcs** é igual ao valor contido na variável **d-pci-pcs** e (o *"pixel-analisado"* encontra-se a direita do ponto de cruzamento encontrado na linha horizontal superior))), a variável denominada no programa de **ind-pc** é igualada a 1. A variável **ind-pc** indica que a coordenada do *"pixel-analisado"* deve ser armazenada como sendo a coordenada do ponto de cruzamento da linha horizontal superior (se o parâmetro de entrada é "sup") ou do ponto de cruzamento da linha horizontal inferior (se o parâmetro de entrada é "inf").

- Se um dos parâmetros de entrada da função horiz-conec é "sup" e o *"pixel-analisado"* encontra-se a esquerda do ponto de cruzamento encontrado na linha horizontal superior, a variável **ind-pc** é igualada a 1.

- Se um dos parâmetros de entrada da função horiz-conec é "sup" e ((a abscissa do *"pixel-analisado"* encontra-se na faixa compreendida entre 80% e 140% da largura dos caracteres contidos na nova-matriz) e ((ainda não foi encontrado ponto de cruzamento na linha horizontal superior) ou (o valor da variável **ind-pc** é igual a 1)) e (o *"pixel-analisado"* não é o primeiro "pixel-on" a ser varrido pela função horiz-conec)), o programa armazena a coordenada do *"pixel-analisado"* como sendo a coordenada do ponto de cruzamento encontrado na linha horizontal superior.

- Se um dos parâmetros de entrada da função horiz-conec é "inf" e (((a coordenada do *"pixel-analisado"* não coincide com a coordenada do ponto de cruzamento encontrado na linha horizontal superior) e (a abscissa do *"pixel-analisado"* encontra-se na faixa compreendida entre 80% e 140% da largura dos caracteres contidos na nova-matriz)) e ((ainda não foi encontrado ponto de cruzamento na linha horizontal inferior) ou (o valor da variável **ind-pc** é igual a 1)) e (o *"pixel-analisado"* não é o primeiro "pixel-on" a ser varrido pela função horiz-conec)), o programa armazena a coordenada do *"pixel-analisado"* como sendo a coordenada do ponto de cruzamento encontrado na linha horizontal inferior.

- Se a função horiz-conec recebe como parâmetros de entrada "right" e "sup", o programa verifica se ((o *"pixel-analisado"* possui um "pixel-on" vizinho que se encontra na direção 1, 0 ou 7) ou (a função horiz-conec recebe como parâmetros de entrada "right" e "inf")), o programa verifica se o *"pixel-analisado"* possui um "pixel-on" vizinho que se encontra na direção 7, 0 ou 1, o programa considera o "pixel-on" vizinho como o próximo "pixel-on" a ser varrido pela função horiz-conec.

- Se a função horiz-conec recebe como parâmetros de entrada "left" e "sup", o programa verifica se ((o *"pixel-analisado"* possui um "pixel-on" vizinho que se encontra na direção 3, 4 ou 5) ou (a função horiz-conec recebe como parâmetros de entrada "left" e "inf")), o programa verifica se o *"pixel-analisado"* possui um "pixel-on" vizinho que se encontra na direção 5, 4 ou 3)), o programa considera o "pixel-on" vizinho como o próximo "pixel-on" a ser varrido pela função horiz-conec.

- Se o *"pixel-analisado"* não possui "pixel-on" vizinho que se encontra na

direção (1, 0 ou 7) ou (3, 4 ou 5), o programa retorna ao "pixel-on" cuja coordenada foi armazenada nas variáveis **L1** e **C1** e este "pixel-on" será o próximo "pixel-on" a ser varrido pela função horiz-conec.

Se (((não existem mais "pixels-on" que possam ser varridos na nova-matriz) ou (o processo de varrer a nova-matriz para a esquerda (parâmetro de entrada da função horiz-conec "left") ou para a direita (parâmetro de entrada da função horiz-conec "right") foi finalizado)) e (não foi armazenado nas variáveis **LF** e **CF**, respectivamente, a ordenada e a abscissa de algum "pixel-on" varrido)), o programa armazena nestas variáveis, respectivamente, a ordenada e a abscissa do último "pixel-on" varrido.

Se ((não foi encontrado ponto de cruzamento) e ((um dos parâmetros de entrada da função horiz-conec é "left" e a abscissa do "pixel-on" que está sendo varrido está acima de 80% da largura média dos caracteres contidos na nova-matriz) ou (um dos parâmetros de entrada da função horiz-conec é "right" e a abscissa do "pixel-on" que está sendo varrido está abaixo de 140% da largura média dos caracteres contidos na nova-matriz))), o programa a partir do último "pixel-on" varrido continua a varrer a nova-matriz para a esquerda, se o parâmetro de entrada da função horiz-conec é "left", ou para a direita, se o parâmetro de entrada dessa função é "right".

Se ((foi encontrado ponto de cruzamento) ou ((um dos parâmetros de entrada da função horiz-conec é "left" e a abscissa do "pixel-on" que está sendo varrido está abaixo de 80% da largura média dos caracteres contidos na nova-matriz) ou (um dos parâmetros de entrada da função horiz-conec é "right" e a abscissa do "pixel-on" que está sendo varrido está acima de 140% da largura média dos caracteres contidos na nova-matriz))), todo o processo de varrer a nova-matriz é finalizado.

Ao terminar o processo de varrer a nova-matriz, a coordenada armazenada

nas variáveis **LF** e **CF** determina a coordenada final da linha horizontal superior ou a coordenada final da linha horizontal inferior. Com o objetivo de determinar qual das linhas horizontais a coordenada armazenada nas variáveis **LF** e **CF** pertence, o programa realiza as análises descritas a seguir:

- Se um dos parâmetros de entrada da função horiz-conec é "right", o programa verifica se outro parâmetro de entrada dessa função é "sup". Em caso afirmativo, a abscissa e a ordenada armazenadas, respectivamente, nas variáveis **LF** e **CF**, são armazenadas na coordenada CFLHSVD. Caso contrário, a abscissa e a ordenada armazenadas, respectivamente, nas variáveis **LF** e **CF**, são armazenadas na coordenada CFLHIVD.

- Se um dos parâmetros de entrada da função horiz-conec é "left", o programa verifica se outro parâmetro de entrada dessa função é "sup". Em caso afirmativo, a abscissa e a ordenada armazenadas, respectivamente, nas variáveis **LF** e **CF**, são armazenadas na coordenada CFLHSVE. Caso contrário, a abscissa e a ordenada armazenadas, respectivamente, nas variáveis **LF** e **CF**, são armazenadas na coordenada CFLHIVE.

Ver-Conec-00

O objetivo da função ver-conec-00 é verificar se ao varrer a nova-matriz para baixo, a partir do ponto de cruzamento da linha horizontal superior encontrado pela função horiz-conec, é possível alcançar o ponto de cruzamento da linha horizontal inferior encontrado pela função horiz-conec. A nova-matriz armazena os "pixels-on" e os "pixels-off" do esqueleto da imagem de um caractere isolado ou o esqueleto da imagem de caracteres conectados.

Para facilitar o entendimento de todo o processo utilizado na função ver-

conec-00, optou-se por explicá-lo no texto e esboçar o seu fluxograma. Aconselha-se ao leitor que para sua melhor clareza, leia o texto que descreve a função ver-conec-00 e acompanhe ao mesmo tempo o seu fluxograma. A figura 5.8 que se encontra nas páginas 632 a 641 do 2º volume deste trabalho, ilustra o fluxograma da função ver-conec-00. Nesta figura, alguns comentários que se encontram no texto não constam no fluxograma. Devido à extensão do fluxograma ele foi dividido em 9 partes.

Ao iniciar a função ver-conec-00, o programa verifica se o ponto de cruzamento da linha horizontal superior está situado acima do ponto de cruzamento da linha horizontal inferior. Em caso afirmativo, o programa denomina o ponto de cruzamento da linha horizontal superior de *pcs* e o ponto de cruzamento da linha horizontal inferior de *pci*. Caso contrário, o programa denomina o ponto de cruzamento da linha horizontal inferior de *pcs* e o ponto de cruzamento da linha horizontal superior de *pci*.

A função ver-conec-00 recebe como parâmetro a largura média dos caracteres contidos na nova-matriz e esta largura média é armazenada na variável **med**. Se o valor da variável **med** é diferente de -1, a função ver-conec-00 deve varrer para baixo, apenas a parte dos caracteres contidos na nova-matriz que se encontra na faixa compreendida entre 80% e 140% da largura média desses caracteres. Caso contrário, a função ver-conec-00 deve varrer para baixo, todo o esqueleto da imagem dos caracteres contidos na nova-matriz.

Se o valor da variável **med** é diferente de -1 e o ponto de cruzamento *pcs* é igual ao ponto de cruzamento *pci*, a variável **si** é igualada a 11. Caso contrário, a nova-matriz é varrida para baixo, a partir do ponto de cruzamento *pcs*. A nova-matriz é varrida para baixo, até que não existam mais "pixels-on" que possam ser varridos nessa direção ou a variável **si** seja igualada a 10. Quando a variável **si** é igualada a 11, os pontos de cruzamentos *pcs* e *pci* por serem iguais devem ser desconsiderados quando for realizado o processo de desconexão dos caracteres e

a função conectados deve ser executada, pois o programa considera que na nova-matriz existem 2 linhas horizontais superiores e 2 linhas horizontais inferiores, ou seja, nessa matriz existem 2 ou mais caracteres conectados. Quando a variável **si** é igualada a 10, os pontos de cruzamento *pcs* e *pci* devem ser considerados quando for realizado o processo de desconexão dos caracteres e na nova-matriz existem 2 linhas horizontais superiores e 2 linhas horizontais inferiores, ou seja, nessa matriz existem 2 ou mais caracteres conectados.

Ao varrer a nova-matriz para baixo, os seguintes procedimentos são realizados:

- Ao terminar de varrer a nova-matriz para baixo, se durante o processo de varrer essa matriz nessa direção foi armazenada a coordenada de um ou mais pontos de cruzamento, o programa continua a varrer a nova-matriz para baixo, a partir de um dos pontos de cruzamento cuja coordenada foi armazenada.

- Se a coordenada de um "pixel-on" que está sendo varrido na nova-matriz coincide com a coordenada do ponto de cruzamento *pci*, a variável **si** é igualada a 10.

- Se ((o valor da variável **"condição-de-folga"** é diferente de 0) e ((o "pixel" (denominado no programa de *"primeiro-pixel"* (este "pixel" pode ser "on" ou "off")) que está sendo varrido possui "pixel-on" vizinho que se encontra na direção 0, 1, 2, 3 ou 4) e (o número de "pixels-on" varridos que se encontram nas direções 0, 1, 2, 3 e 4 é menor do que 3))), a variável **"condição-de-folga"** é igualada a 0 e o programa continua a varrer a nova-matriz para baixo, a partir do "pixel" (este "pixel" pode ser "on" ou "off") cuja a abscissa é igual a abscissa do *"primeiro-pixel"* e a ordenada é a ordenada do *"primeiro-pixel"* decrementada de 2 unidades.

- Se (((é encontrado um ponto de cruzamento) ou (o "pixel-on" que está sendo varrido é o primeiro "pixel-on" a ser varrido)) e (o valor da variável **si** é diferente de 10)), o programa realiza os seguintes procedimentos:

  - Armazena respectivamente nas variáveis **L1** e **C1**, a ordenada e a abscissa do "pixel-on" que está sendo varrido e retorna ao ponto de cruzamento, se existir ponto de cruzamento ou retorna ao primeiro "pixel-on" que foi varrido para baixo. O "pixel-on" que o programa retorna é denominado de *"pixel-on em análise"*.

  - Se ((o *"pixel-on em análise"* é o primeiro "pixel-on" a ser varrido) ou (a ordenada e a abscissa do *"pixel-on em análise"* é diferente da ordenada e da abscissa armazenadas, respectivamente, nas variáveis **L1** e **C1)**), o programa verifica se o *"pixel-on em análise"* encontra-se a direita do ponto de cruzamento *pci* e possui "pixel-on" vizinho que se encontra na direção 7. Em caso afirmativo, o próximo "pixel-on" a ser varrido para baixo é o "pixel-on" vizinho que se encontra na direção 7. Se o *"pixel-on em análise"* não possui "pixel-on" vizinho que se encontra na direção 7, o programa verifica se o *"pixel-on" em análise"* possui "pixel-on" vizinho que se encontra na direção 0. Em caso afirmativo, o próximo "pixel-on" a ser varrido para baixo é o "pixel-on" vizinho que se encontra na direção 0.

  - Se o *"pixel-on em análise"* encontra-se a esquerda do ponto de cruzamento *pci* e possui "pixel-on" vizinho que se encontra na direção 5, o próximo "pixel-on" a ser varrido para baixo é o "pixel-on" vizinho que se encontra na direção 5. Se o *"pixel-on em análise"* não possui "pixel-on" vizinho que se encontra na direção 5,

o programa verifica se o *"pixel-on em análise"* possui "pixel-on" vizinho que se encontra na direção 4. Em caso afirmativo, o próximo "pixel-on" a ser varrido para baixo é o "pixel-on" vizinho que se encontra na direção 4. Se o *"pixel-on em análise"* não possui "pixel-on" vizinho que se encontra na direção 4, o programa retorna ao "pixel-on" cuja a abscissa e a ordenada foram armazenadas respectivamente, nas variáveis **L1** e **C1** e continua a varrer a nova-matriz para baixo. A nova-matriz é varrida para baixo, até que não existam mais "pixels-on" que possam ser varridos nessa direção.

Após terminar de varrer a nova-matriz para baixo, se ((o valor da variável **med** é diferente de -1) e (o valor da variável **si** é igual a 10) e (o valor da variável denominada na função verif-conec-isolado de **si** é igual a 1)), o programa verifica se ao executar a função verif-conec-isolado foi encontrada a coordenada CILHSVD. Em caso afirmativo, a função verif-HS-HI é executada. Caso contrário, a função verif-pc-HS-HI é executada. As funções verif-HS-HI e verif-pc-HS-HI serão explicadas adiante neste capítulo. Com o objetivo de executar as funções verif-HS-HI e verif-pc-HS-HI, o trecho da nova-matriz que foi varrido para baixo, enquanto a função ver-conec-00 é executada é reposto. Após a execução dessas funções, o valor da variável **si** é alterado. O novo valor que a variável **si** recebe, indica se os pontos de cruzamento encontrados ao varrer a nova-matriz devem ser analisados ou não e indica o número de linhas horizontais superiores e linhas horizontais inferiores presentes nos caracteres que se encontram na nova-matriz.

Se o valor da variável **med** é diferente de -1, o programa verifica se o valor da variável **si** é igual a 10. Em caso afirmativo, na nova-matriz existem 2 ou mais caracteres conectados e quando for realizada a desconexão desses caracteres os pontos de cruzamento *pcs* e *pci* devem ser considerados. Se o valor da variável **si**

é igual a 11, na nova-matriz existem 2 ou mais caracteres conectados e quando for realizada a desconexão desses caracteres, os pontos de cruzamento *pcs* e *pci* não devem ser considerados. Se o valor da variável **si** é diferente de 10 ou 11, o programa considera que na nova-matriz existe apenas um caractere isolado.

Verif-HS-HI

A função verif-HS-HI tem como objetivo verificar se a partir do ponto de cruzamento encontrado na linha horizontal inferior é possível encontrar as coordenadas CILHSVD, CFLHSVD, CILHSVE ou CFLHSVE.

Para facilitar o entendimento de todo o processo utilizado na função verif-HS-HI, optou-se por explicá-lo no texto e esboçar o seu fluxograma. Aconselha-se ao leitor que para sua melhor clareza, leia o texto que descreve a função verif-HS-HI e acompanhe ao mesmo tempo o seu fluxograma. A figura 5.9 que se encontra nas páginas 642 a 644 do 2º volume deste trabalho, ilustra o fluxograma da função verif-HS-HI. Nesta figura, alguns comentários que se encontram no texto não constam no fluxograma. Devido à extensão do fluxograma ele foi dividido em 2 partes.

O programa inicia a função verif-HS-HI, varrendo a nova-matriz para cima, para baixo, para a esquerda ou para a direita, conforme os "pixels-on" estejam dispostos, a partir do ponto de cruzamento da linha horizontal inferior. A nova-matriz é varrida para cima, para baixo, para a esquerda ou para a direita, até que não existam mais "pixels-on" que possam ser varridos em nenhuma dessas direções. A nova-matriz armazena os "pixels-on" e os "pixels-off" do esqueleto da imagem de um caractere isolado ou o esqueleto da imagem de caracteres conectados.

O programa verifica se foi armazenada a coordenada de um ou mais pontos de cruzamento, quando não existirem mais "pixels-on" na nova-matriz que

possam ser varridos para cima, para baixo, para a esquerda ou para a direita. Em caso afirmativo, o programa retorna a um dos pontos de cruzamento cujas as coordenadas foram armazenadas e continua a varrer a nova-matriz para cima, para baixo, para a esquerda ou para a direita. Caso contrário, o processo de varrer a nova-matriz para cima, para baixo, para a esquerda ou para a direita é finalizado.

Após finalizar o processo de varrer a nova-matriz para cima, para baixo, para a esquerda ou para a direita, o programa verifica se foi encontrada a coordenada CILHSVD (coordenada inicial da linha horizontal superior varrida para a direita), CFLHSVD (coordenada final da linha horizontal superior varrida para a direita), CILHSVE (coordenada inicial da linha horizontal superior varrida para a esquerda), ou CFLHSVE (coordenada final da linha horizontal superior varrida para a esquerda). Em caso afirmativo, a variável denominada na função ver-conec-00 de **si** é igualada a 10. Caso contrário, a variável denominada na função ver-conec-00 de **si** é igualada a 1.

Verif-pc-HS-HI

A função verif-pc-HS-HI tem como objetivo verificar se a partir do ponto de cruzamento da linha horizontal inferior encontrado na função horiz-conec, é possível varrer a nova-matriz para a direita, encontrar o ponto de cruzamento da linha horizontal superior encontrado na função horiz-conec, retornar ao ponto de cruzamento da linha horizontal inferior, varrer a nova-matriz para a esquerda e novamente encontrar o ponto de cruzamento da linha horizontal superior. A nova-matriz armazena os "pixels-on" e os "pixels-off" que constituem o esqueleto da imagem de um caractere isolado ou de caracteres conectados.

Os pontos de cruzamento da linha horizontal superior e da linha horizontal inferior, encontrados quando foi executada a função horiz-conec são

denominados na função verif-pc-HS-HI, respectivamente de *pcsup* e *pcinf*.

Para facilitar o entendimento de todo o processo utilizado na função verif-pc-HS-HI, optou-se por explicá-lo no texto e esboçar o seu fluxograma. Aconselha-se ao leitor que para sua melhor clareza, leia o texto que descreve a função verif-pc-HS-HI e acompanhe ao mesmo tempo o seu fluxograma. A figura 5.10 que se encontra nas páginas 645 a 652 do 2º volume deste trabalho, ilustra o fluxograma da função verif-pc-HS-HI. Nesta figura, alguns comentários que se encontram no texto não constam do fluxograma. Devido à extensão do fluxograma ele foi dividido em 7 partes.

Ao executar a função verif-pc-HS-HI, o programa inicializa as variáveis **ud** e **ind** com os valores "D" (para baixo) e 0, respectivamente. As variáveis **ud** e **ind** indicam a direção que a nova-matriz está sendo varrida e se foi encontrado o ponto de cruzamento *pcsup*, respectivamente.

A nova-matriz é varrida a partir do ponto de cruzamento *pcinf*, inicialmente para a direita, para cima ou para baixo, conforme os "pixels-on" estejam dispostos. A nova-matriz é varrida inicialmente para a direita, para cima ou para baixo, até que não existam mais "pixels-on" que possam ser varridos em nenhuma dessas direções ou o ponto de cruzamento *pcsup* seja encontrado ao varrer a nova-matriz inicialmente para a esquerda, para cima ou para baixo.

Enquanto a nova-matriz é varrida para a direita, para cima ou para baixo, se é encontrado o ponto de cruzamento *pcsup*, o programa verifica se o valor da variável **ud** é igual a "D". Em caso afirmativo, a variável **ind** é igualada a 1, indicando que foi encontrado o ponto de cruzamento *pcsup*, a variável **ud** é igualada a "U" (para cima), todos os pontos de cruzamento encontrados enquanto a nova-matriz era varrida para direita, para cima ou para baixo são desconsiderados, o programa retorna ao ponto de cruzamento *pcinf* e varre a nova-matriz inicialmente para a esquerda, para cima ou para baixo, conforme os "pixels-on" estejam dispostos. A nova-matriz é varrida inicialmente para a

esquerda, para cima ou para baixo, até que não existam mais "pixels-on" que possam ser varridos em nenhuma dessas direções ou o ponto de cruzamento *pcsup* seja encontrado novamente.

Enquanto a nova-matriz é varrida para a direita, para cima ou para baixo, o programa verifica se é encontrado um ponto de cruzamento. Em caso afirmativo, o programa armazena nas variáveis **L1** e **C1**, respectivamente, a ordenada e a abscissa do ponto de cruzamento encontrado e se o valor da variável **ud** é igual a "D", é verificado se o ponto de cruzamento encontrado possui "pixel-on" vizinho que se encontra na direção 6, 7 ou 5. Se ocorrer "pixel-on" vizinho em uma dessas direções, o programa continua a varrer a nova-matriz para a direita, para cima ou para baixo, a partir do "pixel-on" vizinho encontrado. Se não existir "pixel-on" vizinho em nenhuma dessas direções, o programa continua a varrer a nova-matriz para a direita, para cima ou para baixo, a partir do "pixel-on" cuja ordenada e a abscissa foram armazenadas, respectivamente, nas variáveis **L1** e **C1**.

Se não existem mais "pixels-on" na nova-matriz que possam ser varridos para a direita, para cima ou para baixo, o programa verifica se os pontos de cruzamento cujas coordenadas foram armazenadas anteriormente, não foram desconsiderados. Em caso afirmativo, o programa continua a varrer a nova-matriz para direita, para cima ou para baixo, a partir de um desses pontos de cruzamento. Caso contrário, o programa finaliza o processo de varrer a nova-matriz para a direita, para cima ou para baixo.

Se ao varrer a nova-matriz para a direita, para cima ou para baixo, foi encontrado o ponto de cruzamento *pcsup*, o programa varre a nova-matriz inicialmente para a esquerda, para cima ou para baixo, a partir do ponto de cruzamento *pcinf*. Enquanto a nova-matriz é varrida para a esquerda, para cima ou para baixo, o programa verifica se é encontrado novamente o ponto de cruzamento *pcsup* e se o valor da variável **ind** é igual a 1. Em caso afirmativo, a

variável **ind** é igualada a 2 e o programa finaliza o processo de varrer a nova-matriz para a esquerda, para cima ou para baixo.

Enquanto a nova-matriz é varrida para a esquerda, para cima ou para baixo, o programa verifica se é encontrado um ponto de cruzamento. Em caso afirmativo, o programa armazena nas variáveis **L1** e **C1**, respectivamente, a ordenada e a abscissa do ponto de cruzamento encontrado. Se o ponto de cruzamento encontrado possui "pixel-on" vizinho que se encontra na direção 2, 1 ou 3, o programa continua a varrer a nova-matriz para a esquerda, para cima ou para baixo, a partir do "pixel-on" vizinho encontrado. Se não existir "pixel-on" vizinho em nenhuma dessas direções, o programa continua a varrer a nova-matriz para a esquerda, para cima ou para baixo, a partir do "pixel-on" cuja ordenada e a abscissa foram armazenadas, respectivamente, nas variáveis **L1** e **C1**.

Se não existem mais "pixels-on" na nova-matriz que possam ser varridos para a esquerda, para cima ou para baixo, o programa verifica se os pontos de cruzamento cujas coordenadas foram armazenadas anteriormente, não foram desconsiderados. Em caso afirmativo, o programa continua a varrer a nova-matriz para a esquerda, para cima ou para baixo, a partir de um desses pontos de cruzamento. Caso contrário, o programa finaliza o processo de varrer a nova-matriz para a esquerda, para cima ou para baixo.

Ao terminar de varrer a nova-matriz ((para a direita, para cima ou para baixo) ou (para a esquerda, para cima ou para baixo)), o programa verifica se o valor da variável **ind** é igual a 2. Em caso afirmativo, a variável denominada na função ver-conec-00 de **si** é igualada a 10. Caso contrário, a variável denominada na função ver-conec-00 de **si** é igualada a 1.

Isolad-conec

O objetivo da função isolad-conec é verificar se é possível nos caracteres

cujo valor da variável **si** é igual a 2 ou 4, partir da coordenada CILHIVD ou CILHIVE, encontrar as coordenadas (CILHSVD ou CFLHSVD) e (CILHSVE ou CFLHSVE). E nos caracteres cujo valor da variável **si** é diferente de 2 e 4, partir da coordenada CILHIVE, encontrar as coordenadas CILHIVD e (CILHSVD ou CILHSVE). As coordenadas CILHIVD, CILHSVD, CFLHSVD, CILHIVE, CILHSVE e CFLHSVE foram encontradas na função verif-conec-isolado. A variável **si** foi definida na função verif-conec-isolado.

Para facilitar o entendimento de todo o processo utilizado na função isolad-conec, optou-se por explicá-lo no texto e esboçar o seu fluxograma. Aconselha-se ao leitor que para sua melhor clareza, leia o texto que descreve a função isolad-conec e acompanhe ao mesmo tempo o seu fluxograma. A figura 5.11 que se encontra nas páginas 653 a 667 do 2º volume deste trabalho, ilustra o fluxograma da função isolad-conec. Nesta figura, alguns comentários que se encontram no texto não constam no fluxograma. Devido à extensão do fluxograma ele foi dividido em 14 partes.

Se o valor da variável **si** é igual a 3 ou 4, o programa varre a nova-matriz para cima, a partir da coordenada CILHIVE. Se o valor da variável **si** é igual a 2, o programa varre a nova-matriz para cima, a partir da coordenada CILHIVD, se ao executar a função horiz-conec esta coordenada foi encontrada, ou a partir da coordenada CILHIVE encontrada nessa função. Independente do valor da variável **si** ser igual a 2, 3 ou 4, a nova-matriz é varrida para cima, até que não existam mais "pixels-on" que possam ser varridos nessa direção. As variáveis **sup0** e **sup1**, definidas adiante, são inicializadas com o valor 0. A nova-matriz armazena os "pixels-on" e os "pixels-off" que constituem o esqueleto da imagem de um caractere isolado ou de caracteres conectados.

Enquanto a nova-matriz é varrida para cima, se são encontrados pontos de cruzamento, o programa armazena as coordenadas destes pontos de cruzamento. Se o valor da variável **si** é igual a 2 ou 4, o programa verifica se ((o valor da

variável **sup0** é igual a 0) e (foi encontrada a coordenada CILHSVD ou CFLHSVD)). Em caso afirmativo, a variável **sup0** é igualada a 1 e o processo de varrer a nova-matriz para cima é finalizado. Se as coordenadas CILHSVD e CFLHSVD não foram encontradas, o programa verifica se ((o valor da variável **sup1** é igual a 0) e ((foi encontrada a coordenada CILHSVE ou CFLHSVE) ou ((não existem mais "pixels-on" na nova-matriz que possam ser varridos para cima) e (a coordenada do último "pixel-on" varrido para cima está a uma distância menor ou igual a 4 "pixels-on" da coordenada CFLHSVE)))), a variável **sup1** é igualada a 1 e o processo de varrer a nova-matriz para cima é finalizado. A variável **sup0** indica se foi encontrada a coordenada CILHSVD ou CFLHSVD e a variável **sup1** indica se a coordenada do último "pixel-on" varrido para cima está a uma distância menor ou igual a 4 "pixels-on" da coordenada CFLHSVE ou foi encontrada a coordenada CILHSVE ou CFLHSVE.

Se o valor da variável **si** é diferente de 2 e 4, o programa verifica se ((o valor da variável **sup0** é igual a 0) e (foi encontrada a coordenada CILHIVD e foi encontrado algum ponto de cruzamento, enquanto a nova-matriz era varrida para cima)), a variável **sup0** é igualada a 1. Após isto, o programa verifica se ((o valor da variável **sup1** é igual a 0) e ((foi encontrada a coordenada CILHSVD ou CILHSVE) e (foi encontrado ponto de cruzamento, enquanto a nova-matriz era varrida para cima))), a variável **sup1** é igualada a 1 e o processo de varrer a nova-matriz para cima é finalizado.

Se ((o processo de varrer a nova-matriz para cima foi finalizado) ou (não existem mais "pixels-on" que possam ser varridos para cima na nova-matriz)), o programa verifica se ((o valor da variável **si** é igual a 3) e ((a coordenada do último "pixel-on" varrido para cima é diferente da coordenada CILHSVD) e (esse "pixel-on" encontra-se acima do "pixel-on" cuja coordenada foi armazenada na variável **extr**))), o programa armazena na variável **extr** a coordenada do último "pixel-on" varrido para cima. A variável **extr** foi definida na função elim-virg.

Após verificar se o valor da variável **si** é igual a 3, o programa verifica se enquanto a nova-matriz era varrida para cima foram armazenadas coordenadas de pontos de cruzamento. Em caso afirmativo, o programa retorna a um dos pontos de cruzamento cuja coordenada foi armazenada e continua a varrer a nova-matriz para cima.

Se ao executar a função horiz-conec foi encontrada a coordenada CILHSVD, o programa varre a nova-matriz para baixo, a partir dessa coordenada. A nova-matriz é varrida para baixo, até que não existam mais "pixels-on" que possam ser varridos nessa direção ou seja encontrado um ponto de cruzamento. Ao varrer a nova-matriz para baixo, a variável **sup2** é igualada a 0 e os "pixels-on" varridos são marcados, com o objetivo de que estes "pixels-on" não sejam varridos novamente, quando o programa retornar a coordenada CILHSVD e varrer a nova-matriz para a direita, para cima ou para baixo. Ao terminar de varrer a nova-matriz para baixo, o programa retorna a coordenada CILHSVD e a partir desta coordenada varre a nova-matriz para a direita, para cima ou para baixo, se existirem "pixels-on" que possam ser varridos em alguma dessas direções. A nova-matriz é varrida para a direita, para cima ou para baixo, até que não existam mais "pixels-on" que possam ser varridos em nenhuma dessas direções. A variável **sup2** é definida no próximo parágrafo.

Ao varrer a nova-matriz para a direita, para cima ou para baixo, se o valor da variável **si** é igual a 2 ou 4, é verificado se é encontrada a coordenada CILHSVE ou CFLHSVE. Em caso afirmativo, o programa iguala a variável **sup2** a 1 e o processo de varrer a nova-matriz para a direita, para cima ou para baixo é finalizado. Se o valor da variável **si** é diferente de 2 e 4, o programa verifica se ao executar a função horiz-conec foi encontrada a coordenada CILHSVE. Em caso afirmativo, o programa verifica se ao varrer a nova-matriz para a direita, para cima ou para baixo, a coordenada CILHSVE ou CFLHSVE foi encontrada. Se foi encontrada uma destas coordenadas, a variável **sup2** é igualada a 1 e o programa

finaliza o processo de varrer a nova-matriz para a direita, para cima ou para baixo. Se ao executar a função horiz-conec não foi encontrada a coordenada CILHSVE, o programa verifica para cada "pixel-on" que está sendo varrido, se o valor absoluto da diferença entre a coordenada do "pixel-on" que está sendo varrido e a coordenada armazenada na variável **extr** é menor ou igual a 4. Em caso afirmativo, a variável **sup2** é igualada a 1 e o programa finaliza o processo de varrer a nova-matriz para a direita, para cima ou para baixo. Caso contrário, o programa continua a varrer a nova-matriz para a direita, para cima ou para baixo. A variável **sup2** indica se foi encontrada a coordenada CILHSVE ou CFLHSVE ou se o "pixel-on" varrido está próximo (diferença menor ou igual a 4 "pixels-on") da coordenada armazenada na variável **extr**.

Se ((o processo de varrer a nova-matriz para direita, para cima ou para baixo foi finalizado) ou (não existem mais "pixels-on" que possam ser varridos para a direita, para cima ou para baixo na nova-matriz) ou (após varrer a nova-matriz para cima, foi verificado que ao executar a função horiz-conec não foi encontrada a coordenada CILHSVD)), o programa verifica se ((o valor da variável **si** é igual a 2 ou 4) e (ao executar a função horiz-conec foi encontrada a coordenada CILHIVD)). Em caso afirmativo, o programa varre a nova-matriz para a direita, para cima ou para baixo, a partir da coordenada CILHIVD. A nova-matriz é varrida para a direita, para cima ou para baixo, até que não existam mais "pixels-on" que possam ser varridos em nenhuma dessas direções.

Ao varrer a nova-matriz para a direita, para cima ou para baixo, se é encontrado um ponto de cruzamento, o programa realiza os seguintes procedimentos:

- Armazena a ordenada e a abscissa do "pixel-on" que está sendo varrido, nas variáveis **L1** e **C1**, respectivamente.

- Se o ponto de cruzamento encontrado situa-se no intervalo compreendido entre 80% e 140% da largura média dos caracteres contidos na nova-matriz, a variável **pix** é igualada a 0 e se o ponto de cruzamento encontrado é o primeiro ponto de cruzamento a ser encontrado nesse intervalo, sua coordenada é armazenada na variável **pc1**. A variável **pix** armazena a quantidade de "pixels-on" varridos após ser encontrado um ponto de cruzamento que se encontra no intervalo compreendido entre 80% e 140% da largura média dos caracteres contidos na nova-matriz e a variável **pc1** armazena a coordenada do primeiro ponto de cruzamento a ser encontrado no intervalo compreendido entre 80% e 140% da largura média dos caracteres contidos na nova-matriz.

- Se o ponto de cruzamento encontrado possui "pixel-on" vizinho que se encontra na direção 7, 6, 0 ou 1, nesta ordem, o programa continua a varrer a nova-matriz para a direita, para cima ou para baixo, a partir do "pixel-on" vizinho encontrado. Caso contrário, o programa retorna a coordenada armazenada nas variáveis **L1** e **C1** e continua a varrer a nova-matriz para a direita, para cima ou para baixo.

Se ao varrer a nova-matriz para a direita, para cima ou para baixo, o "pixel-on" que está sendo varrido é o primeiro "pixel-on" a ser varrido cuja coordenada coincide com a coordenada CFLHIVD, o programa verifica se foi encontrado ponto de cruzamento no intervalo compreendido entre 80% e 140% da largura média dos caracteres contidos na nova-matriz. Em caso afirmativo, a variável **HIF** é igualada a "D". Caso contrário, a variável **HIF** é igualada a "A". A variável **HIF** indica se foi encontrada a coordenada CFLHIVD antes ou depois de ser encontrado o primeiro ponto de cruzamento situado no intervalo compreendido entre 80% e 140% da largura média dos caracteres contidos na nova-matriz.

Se ((((o valor da variável **pix** é maior ou igual a 4) e (o valor da variável

HIF é igual a "A" ou a "D")) ou (não existem mais "pixels-on" que possam ser varridos para a direita, para cima ou para baixo)) e (o valor da variável denominada no programa de **pix-r** é igual a -1)), o valor da variável **pix-r** é igualado ao valor da variável **pix** e esta última variável é igualada a 0 e o programa retorna ao ponto de cruzamento cuja coordenada foi armazenada na variável **pc1**. Se existem "pixels-on" que podem ser varridos para a direita, para cima ou para baixo, o programa continua a varrer a nova-matriz em alguma dessas direções. A variável **pix-r** armazena a quantidade de "pixels-on" varridos entre a coordenada CFLHIVD e o ponto de cruzamento situado no intervalo compreendido entre 80% e 140% da largura média dos caracteres contidos na nova-matriz.

Quando não existem mais "pixels-on" que possam ser varridos para a direita, para cima ou para baixo, o programa verifica se ((os valores das variáveis **sup0** e **sup1** são iguais a 1) e (o valor da variável **sup2** é diferente de 1)), a função isolad-conec retorna à função verif-conec-isolado a variável **si** com valor igual a 1. Caso contrário, a função isolad-conec retorna à função verif-conec-isolado a variável **si** com valor igual a 2.

Se ((os valores das variáveis **pix** e **pix-r** são maiores ou iguais a 4) ou (o valor da variável **HIF** é igual a "D")), a função isolad-conec retorna à função verif-conec-isolado a variável **si** com valor igual a 3.

5.4.2 - Classificação e Segmentação dos Caracteres Conectados

Após ser verificado que os caracteres que constituem a nova-matriz são conectados, o programa executa as funções denominadas de conectados e elim-conec. A função conectados classifica a maneira como os caracteres conectados serão desconectados e a função elim-conec realiza a desconexão dos caracteres conectados.

A seguir são explicadas as funções conectados e elim-conec:

Conectados

O objetivo da função conectados é classificar a maneira como os caracteres conectados que constituem a nova-matriz serão desconectados. A nova-matriz armazena os "pixels-on" e os "pixels-off" que constituem o esqueleto da imagem de caracteres conectados.

Com o objetivo de verificar se é encontrado ponto de cruzamento no intervalo compreendido entre 80% e 140% da largura média dos caracteres contidos na nova-matriz, a função conectados varre esta matriz da esquerda para a direita. Ao varrer a nova-matriz da esquerda para a direita, se é encontrado ponto de cruzamento no intervalo compreendido entre 80% e 140% da largura média dos caracteres contidos na nova-matriz, estes caracteres devem ser separados pela função elim-conec, a partir do ponto de cruzamento encontrado. Caso contrário, os "pixels-on" presentes nesse intervalo devem fazer parte de cada caractere separado pela função elim-conec.

Para facilitar o entendimento de todo o processo utilizado na função conectados, optou-se por explicá-lo no texto e esboçar o seu fluxograma. Aconselha-se ao leitor que para sua melhor clareza, leia o texto que descreve a função conectados e acompanhe ao mesmo tempo o seu fluxograma. A figura 5.12 que se encontra nas páginas 668 a 675 do 2º volume deste trabalho, ilustra o fluxograma da função conectados. Nesta figura, alguns comentários que se encontram no texto não constam do fluxograma. Devido à extensão do fluxograma ele foi dividido em 7 partes.

O programa varre a nova-matriz da esquerda para a direita, a partir da coluna cujo valor é igual a 80% da largura média dos caracteres contidos na nova-matriz. A nova-matriz é varrida da esquerda para a direita, até que seja

encontrado um "pixel-on". O programa denomina este "pixel-on" de *"pixel-inicial"*.

A partir do *"pixel-inicial"*, o programa varre a nova-matriz inicialmente para a direita. Se não for possível varrer a nova-matriz nessa direção, o programa varre esta matriz para cima, para baixo ou para a esquerda, conforme os "pixels-on" estejam dispostos. A nova-matriz é varrida para a direita, para cima, para baixo ou para a esquerda, até que não existam mais "pixels-on" que possam ser varridos em nenhuma dessas direções, seja encontrado um ponto de cruzamento no intervalo compreendido entre 80% e 140% da largura média dos caracteres contidos na nova-matriz ou seja varrido um "pixel-on" cujo o valor da abscissa é maior do que 140% da largura média dos caracteres contidos na nova-matriz. Ao varrer a nova-matriz para a direita, para cima, para baixo ou para a esquerda, o programa sempre analisa a possibilidade de varrer essa matriz inicialmente para a direita, pois apenas deste modo é possível verificar se os caracteres conectados apresentam ponto de cruzamento que se encontra no intervalo compreendido entre 80% e 140% da largura média dos caracteres contidos na nova-matriz. Enquanto a nova-matriz é varrida para a direita, para cima, para baixo ou para a esquerda, o programa armazena a quantidade de "pixels-on" varridos nessas direções.

Se ((não existem na nova-matriz mais "pixels-on" que possam ser varridos para a direita, para cima, para baixo ou para a esquerda) e (a quantidade de "pixels-on" varridos nessas direções é menor ou igual a 7)), o programa retorna ao *"pixel-inicial"* e verifica se este *"pixel-inicial"* possui algum "pixel-on" vizinho. Em caso afirmativo, o programa continua a varrer a nova-matriz para a direita, para cima, para baixo ou para a esquerda, conforme os "pixels-on" estejam dispostos, a partir do *"pixel-inicial"*.

Enquanto a nova-matriz é varrida para a direita, para cima, para baixo ou para a esquerda, se a abscissa do "pixel-on" que está sendo varrido é maior do

que 140% da largura média dos caracteres contidos na nova-matriz, o processo de varrer esta matriz para a direita, para cima, para baixo ou para a esquerda é finalizado.

Se a abscissa do "pixel-on" que está sendo varrido é menor ou igual a 140% da largura média dos caracteres contidos na nova-matriz, o programa verifica se ((o "pixel-on" que está sendo varrido é um ponto de cruzamento denominado no programa de *"pixel-retorno"*) e (não possui "pixel-on" vizinho que se encontra na direção 1, 0 ou 7 (para a direita))). Em caso afirmativo, o programa verifica se algum dos pontos de cruzamento encontrados, enquanto a nova-matriz era varrida para a direita, para cima, para baixo ou para a esquerda possui "pixel-on" vizinho que se encontra na direção 1, 0 ou 7. Se isto ocorrer, o programa continua a varrer a nova-matriz para a direita, para cima, para baixo ou para a esquerda, a partir do ponto de cruzamento encontrado. Caso contrário, o programa verifica se algum dos pontos de cruzamento encontrados possui "pixel-on" vizinho que se encontra na direção 3, 4, 5 ou 6. Em caso afirmativo, o programa continua a varrer a nova-matriz para a direita, para cima, para baixo ou para a esquerda, a partir do "pixel-on" vizinho ao ponto de cruzamento encontrado. Caso contrário, o programa verifica se o *"pixel-retorno"* possui "pixel-on" vizinho que se encontra na direção 6, 5, 4 ou 3. Se isto ocorrer, o programa continua a varrer a nova-matriz para a direita, para cima, para baixo ou para a esquerda, a partir do "pixel-on" vizinho ao *"pixel-retorno"*. Se o *"pixel-retorno"* não possui "pixel-on" vizinho que se encontra na direção 6, 5, 4 ou 3, o programa continua a varrer a nova-matriz para a direita, para cima, para baixo ou para a esquerda, a partir do *"pixel-retorno"*.

Se ((o *"pixel-retorno"* possui "pixel-on" vizinho que se encontra na direção 0, 1 ou 7) e (sua abscissa é menor do que 80% da largura média dos caracteres contidos na nova-matriz)), o programa desconsidera que o "pixel-retorno" seja um ponto de cruzamento e continua a varrer a nova-matriz para a

direita, para cima, para baixo ou para a esquerda. Caso contrário, o programa finaliza o processo de varrer a nova-matriz para a direita, para cima, para baixo ou para a esquerda.

Enquanto a nova-matriz é varrida para a direita, para cima, para baixo ou para a esquerda, o programa verifica se o "pixel-on" que está sendo varrido encontra-se à esquerda da coluna cujo valor é igual a largura média dos caracteres contidos na nova-matriz. Em caso afirmativo, o "pixel-on" que está sendo varrido é marcado com '**x**', a sua coordenada é considerada como sendo a coordenada CILHIVD (coordenada inicial da linha horizontal inferior varrida para a direita) e o programa não considera este "pixel-on" na contagem da quantidade de "pixels-on" varridos para a direita, para cima, para baixo ou para a esquerda. Se o "pixel-on" que está sendo varrido não se encontra a esquerda da coluna cujo o valor é igual a largura média dos caracteres contidos na nova-matriz, o "pixel-on" que está sendo varrido é marcado com '*' e o programa o considera na contagem da quantidade de "pixels-on" varridos para a direita, para cima, para baixo ou para a esquerda. Essa verificação foi realizada com o objetivo de marcar os "pixels-on" que não devem ser analisados no processo de desconexão dos caracteres conectados.

Ao terminar de varrer a nova-matriz para a direita, para cima, para baixo ou para a esquerda, o programa verifica se foi encontrado pelo menos um ponto de cruzamento que se encontra no intervalo compreendido entre 80% e 140% da largura média dos caracteres contidos na nova-matriz. Em caso afirmativo, apenas os "pixels-on" que foram marcados com '*' não serão utilizados no processo de desconexão dos caracteres conectados e a coordenada CILHIVD será a coordenada do último ponto de cruzamento encontrado nesse intervalo. Caso contrário, os "pixels-on" marcados com '**x**' não serão utilizados no processo de desconexão dos caracteres conectados.

Após ser classificada a maneira como os caracteres conectados serão

desconectados e os "pixels-on" que não farão parte desse processo, o programa executa a função elim-conec, que tem como objetivo realizar a desconexão dos caracteres conectados.

Elim-Conec

O objetivo da função elim-conec é realizar a desconexão dos caracteres conectados.

Para facilitar o entendimento de todo o processo utilizado na função elim-conec, optou-se por explicá-lo no texto e esboçar o seu fluxograma. Aconselha-se ao leitor que para sua melhor clareza, leia o texto que descreve a função elim-conec e acompanhe ao mesmo tempo o seu fluxograma. A figura 5.13 que se encontra nas páginas 676 a 684 do 2º volume deste trabalho, ilustra o fluxograma da função elim-conec. Nesta figura, alguns comentários que se encontram no texto não constam do fluxograma. Devido à extensão do fluxograma ele foi dividido em 8 partes.

A função elim-conec varre a nova-matriz, inicialmente para a direita, para cima, para baixo ou para a esquerda, conforme os "pixels-on" estejam dispostos, a partir da coordenada CILHIVD (coordenada inicial da linha horizontal inferior varrida para a direita). A nova-matriz é varrida para a direita, para cima, para baixo ou para a esquerda, até que não existam mais "pixels-on" que possam ser varridos em nenhuma dessas direções. Ao varrer a nova-matriz para a direita, para cima, para baixo ou para a esquerda, os "pixels-on" e os "pixels-off" que são varridos são copiados para a matriz denominada no programa de matriz-final. O programa sempre analisa a possibilidade de varrer a nova-matriz inicialmente para a direita, pois deste modo, ao ser finalizado o processo de desconexão dos caracteres conectados, a matriz-final conterá o caractere da nova-matriz que se encontra mais a direita. A nova-matriz armazena os "pixels-on" e os "pixels-off"

que constituem o esqueleto da imagem de caracteres conectados e a matriz-final armazena os "pixels-on" e os "pixels-off" do esqueleto da imagem de um caractere isolado.

À medida que a nova-matriz é varrida para a direita, para cima, para baixo ou para a esquerda, os "pixels-on" e os "pixels-off" varridos são copiados para a matriz-final, nas posições correspondentes a que estes "pixels-on" e "pixels-off" ocupavam na nova-matriz. Deve-se observar que de acordo com as condições estabelecidas na função conectados, os "pixels-on" marcados com '**x**' ou os "pixels-on" marcados com '*' não serão copiados para a matriz-final.

Enquanto a nova-matriz é varrida para a direita, para cima, para baixo ou para a esquerda, se não forem encontrados mais "pixels-on" que possam ser varridos em alguma dessas direções, ou seja, o valor da variável **"condição-de-folga"** é diferente de 0, o programa verifica se o último "pixel-on" varrido para a direita, para cima, para baixo ou para a esquerda encontra-se a uma distância menor ou igual a 1 da coordenada CILHIVD. Em caso afirmativo, a variável **fga** é igualada a 1 e o programa verifica se enquanto a nova-matriz era varrida para a direita, para cima, para baixo ou para a esquerda, o ponto de cruzamento da linha horizontal inferior *pci* foi encontrado (o ponto de cruzamento *pci* foi encontrado na função ver-conec-00). Se foi encontrada a coordenada do ponto de cruzamento *pci*, o programa retorna ao último ponto de cruzamento encontrado ao varrer a nova-matriz para a direita, para cima, para baixo ou para a esquerda e verifica se à direita desse ponto de cruzamento existe algum "pixel-on". Em caso afirmativo, o "pixel-on" encontrado é considerado como um ponto de cruzamento, o programa armazena sua coordenada e continua a varrer a nova-matriz para a direita, para cima, para baixo ou para a esquerda, a partir desse ponto de cruzamento. Caso contrário, a variável **fim** é igualada a 1. A variável **fga** tem o objetivo de indicar ao programa que próximo a coordenada CILHIVD existem "pixels-on" que possam ser varridos, segundo as condições que serão

mencionadas a seguir. A variável **fim** é utilizada para indicar quando o processo de varrer a nova-matriz deve ser finalizado.

Se o valor da variável **fga** é igual a 1, o programa iguala esta variável a 0 e verifica se o primeiro "pixel-on" varrido após a variável **fga** ter sido igualada a 1 possui "pixels-on" vizinhos que se encontram nas direções ((4, 5 ou 6) e (6, 7 ou 0)). Em caso afirmativo, o programa continua a varrer a nova-matriz para a direita, para cima, para baixo ou para a esquerda. Caso contrário, o processo de varrer a nova-matriz para a direita, para cima, para baixo ou para a esquerda é finalizado.

Se ((não forem encontrados mais "pixels-on" que possam ser varridos para a direita, para cima, para baixo ou para a esquerda) ou (o valor da variável **fim** é igual a 1)), o programa armazena nas variáveis **L1** e **C1**, respectivamente, a ordenada e a abscissa do último "pixel-on" varrido para a direita, para cima, para baixo ou para a esquerda e verifica se a coordenada de algum ponto de cruzamento foi armazenada. Em caso afirmativo, o programa verifica se ((o valor absoluto da diferença entre a coordenada do ponto de cruzamento armazenada e a coordenada armazenada nas variáveis **L1** e **C1** é menor ou igual a 1) e ((não foi desconsiderado nenhum ponto de cruzamento) ou (o número de "pixels-on" varridos após um ponto de cruzamento desconsiderado é maior do que 3))). Se isto ocorrer, a variável **fim** é igualada a 1, finalizando o processo de varrer a nova-matriz para a direita, para cima, para baixo ou para a esquerda. Caso contrário, o programa verifica se o ponto de cruzamento cuja coordenada foi armazenada possui "pixel-on" vizinho que se encontra na direção 6, 7, 0 ou 1. Em caso afirmativo, as variáveis **"condição-de-folga"** e **fga** são igualadas a 0 e o programa continua a varrer a nova-matriz para a direita, para cima, para baixo ou para a esquerda. Se o ponto de cruzamento cuja coordenada foi armazenada não possui "pixel-on" vizinho que se encontra na direção 6, 7, 0 ou 1, o programa continua a varrer a nova-matriz para a direita, para cima, para baixo ou para a

esquerda, a partir desse ponto de cruzamento.

Se ((não forem encontrados mais "pixels-on" que possam ser varridos para a direita, para cima, para baixo ou para a esquerda) ou (o valor da variável **fim** é igual a 1)), o programa verifica se não foi armazenada a coordenada de algum ponto de cruzamento enquanto a nova-matriz era varrida para a direita, para cima, para baixo ou para a esquerda. Em caso afirmativo, a variável **fim** é igualada a 1, finalizando o processo de varrer a nova-matriz para a direita, para cima, para baixo ou para a esquerda.

Enquanto a nova-matriz é varrida para a direita, para cima, para baixo ou para a esquerda, se não passou pelo ponto de cruzamento *pci* e é encontrado um ponto de cruzamento, a ordenada e a abscissa do ponto de cruzamento encontrado são armazenadas, respectivamente, nas variáveis **L1** e **C1** e é verificado se ((a coordenada do ponto de cruzamento encontrado coincide com a coordenada do ponto de cruzamento *pci* e este ponto de cruzamento possui "pixel-on" vizinho que se encontra na direção 6) ou (o ponto de cruzamento encontrado encontra-se a esquerda do ponto de cruzamento *pci* e possui "pixel-on" vizinho que se encontra na direção (5 ou 4)) ou (o ponto de cruzamento encontrado encontra-se a direita do ponto de cruzamento *pci* e possui "pixel-on" vizinho que se encontra na direção (7 ou 0))). Se for encontrado "pixel-on" vizinho que se encontra na direção 6 ou (5 ou 4) ou (7 ou 0), o programa continua a varrer a nova-matriz para a direita, para cima, para baixo ou para a esquerda, a partir do "pixel-on" vizinho encontrado. Caso contrário, o programa continua a varrer a nova-matriz para a direita, para cima, para baixo ou para a esquerda, a partir do "pixel-on" cuja ordenada e a abscissa foram armazenadas, respectivamente, nas variáveis **L1** e **C1**.

Enquanto a nova-matriz é varrida para a direita, para cima, para baixo ou para a esquerda, se a variável **fim** é igualada a 1, o programa finaliza o processo de varrer a nova-matriz para a direita, para cima, para baixo ou para a esquerda.

Caso contrário, o programa continua a varrer a nova-matriz para a direita, para cima, para baixo ou para a esquerda.

Após finalizar o processo de desconexão dos caracteres conectados, a matriz-final conterá os "pixels-on" e os "pixels-off" que constituem o esqueleto da imagem de um caractere, agora isolado. Essa matriz será utilizada no processo de reconhecimento desse caractere. O processo de reconhecimento foi explicado no capítulo 4 deste trabalho e neste processo a matriz-final foi denominada de matriz-original.

A figura 5.14 ilustra alguns números conectados que foram desconectados e reconhecidos pelo algoritmo.

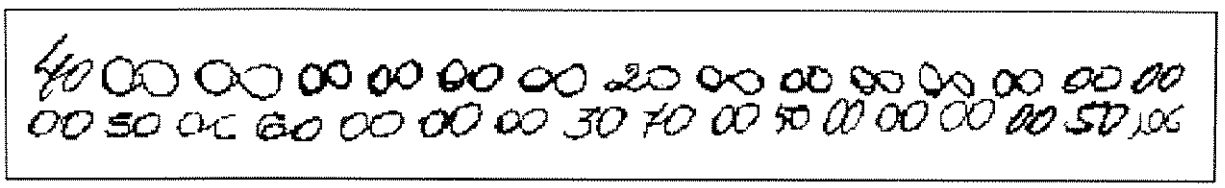


Figura 5.14 - Alguns dos números conectados que foram desconectados e reconhecidos pelo algoritmo.

5.5 - ELIMINAÇÃO DO CAMPO QUE ENVOLVE A QUANTIA NUMÉRICA DE CHEQUES BANCÁRIOS

Em cada folha de cheque bancário existe uma área reservada em que o valor numérico deve ser escrito. Nesta área existe um campo que envolve a quantia numérica. Este campo pode ser composto de linhas verticais e/ou linhas horizontais. A figura 5.15 mostra os tipos de campos encontrados na área reservada a escrita da quantia numérica.

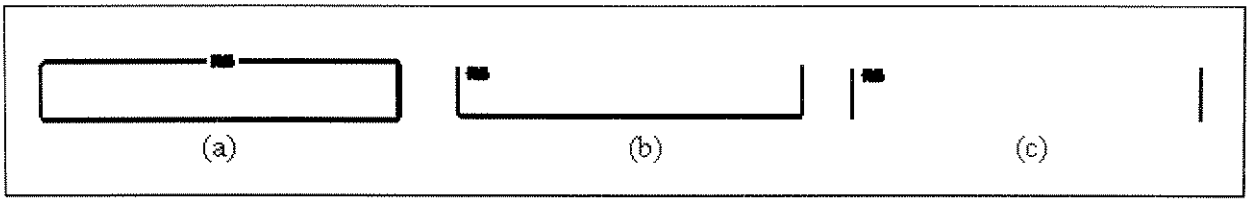


Figura 5.15 - Tipos de campos encontrados na área reservada a escrita da quantia numérica.

Neste trabalho a linha do campo que envolve a quantia numérica que se encontra na vertical é denominada de tarja vertical, enquanto a linha deste campo que se encontra na horizontal é denominada de tarja horizontal. A figura 5.15(a) ilustra o campo que envolve a quantia numérica de um cheque. Este campo possui a tarja horizontal superior, situada na parte superior da matriz-imagem; a tarja vertical esquerda, situada no lado esquerdo dessa matriz; a tarja horizontal inferior, situada na parte inferior da matriz-imagem e a tarja vertical direita, situada no lado direito dessa matriz. A figura 5.15(b) ilustra o campo que envolve a quantia numérica de um cheque. Este campo possui a tarja vertical esquerda, a tarja horizontal inferior e a tarja vertical direita. A figura 5.15(c) ilustra o campo que envolve a quantia numérica de um cheque, onde são encontradas a tarja vertical esquerda e a tarja vertical direita. A matriz-imagem armazena os "pixels-on" e os "pixels-off" do esqueleto da imagem da área reservada à escrita da quantia numérica de um cheque bancário.

Se ao realizar o processo de segmentação e reconhecimento da quantia numérica de um cheque bancário, é verificado que a área reservada à escrita dessa quantia possui as tarjas horizontais e/ou as tarjas verticais, o programa elimina estas tarjas, com o objetivo de que elas não interfiram no processo de segmentação e reconhecimento dos caracteres.

Para verificar se na área reservada à escrita da quantia numérica de um cheque existem as tarjas horizontais e/ou as tarjas verticais, o programa executa as funções tarja-horiz-sup, tarja-horiz-inf e tarja-vertical. A função tarja-horiz-sup tem como objetivo verificar se na matriz-imagem existe a tarja horizontal

superior, a função tarja-horiz-inf tem como objetivo verificar se na matriz-imagem existe a tarja horizontal inferior e a função tarja-vertical tem como objetivo verificar se na matriz-imagem existe a tarja vertical esquerda e/ou a tarja vertical direita. As funções tarja-horiz-sup, tarja-horiz-inf e tarja-vertical são descritas a seguir. Além destas funções também é descrita a função verifica-tarja, que tem como objetivo evitar que os trechos dos caracteres que se encontram conectados à tarja horizontal inferior sejam apagados, quando esta tarja for apagada da matriz-imagem.

Tarja-horiz-sup

A função tarja-horiz-sup verifica se na matriz-imagem existe a tarja horizontal superior. Esta tarja está localizada na parte superior da matriz-imagem. A matriz-imagem armazena os "pixels-on" e os "pixels-off" do esqueleto da imagem da área reservada à escrita da quantia numérica de um cheque bancário.

Para facilitar o entendimento da função tarja-horiz-sup, optou-se por explicá-la no texto e esboçar o seu fluxograma. Aconselha-se ao leitor que para sua melhor clareza, leia o texto que descreve a função tarja-horiz-sup e acompanhe ao mesmo tempo o seu fluxograma. A figura 5.16 que se encontra nas páginas 685 a 689 do 2º volume deste trabalho, ilustra o fluxograma da função tarja-horiz-sup. Nesta figura, alguns comentários que se encontram no texto não constam do fluxograma. Devido à extensão do fluxograma ele foi dividido em 4 partes.

Para verificar se na matriz-imagem existe a tarja horizontal superior, o programa varre essa matriz de cima para baixo, da esquerda para a direita e da direita para a esquerda.

Neste trabalho, a resolução utilizada no "scanner" foi de 100 dpi. Devido a isto, na imagem do campo que envolve a quantia numérica, a tarja horizontal

superior pode ser descontínua. Com o objetivo de eliminar toda a tarja horizontal superior, a matriz-imagem é varrida da esquerda para a direita e da direita para a esquerda.

A matriz-imagem é varrida da esquerda para a direita e de cima para baixo, até que seja encontrado o primeiro "pixel-on" ou a coordenada cuja abscissa é igual a 40% do número total de colunas da matriz-imagem e a ordenada é igual a 50% do número total de linhas da matriz-imagem. Após este processo, a matriz-imagem é varrida da direita para a esquerda e de cima para baixo, até que seja encontrado o primeiro "pixel-on" ou a coordenada cuja abscissa é igual a 60% do número total de colunas da matriz-imagem e a ordenada é igual a 50% do número total de linhas da matriz-imagem.

Ao varrer a matriz-imagem da esquerda para a direita e de cima para baixo, o programa denomina a coordenada do primeiro "pixel-on" encontrado de *cie*. A coordenada do primeiro "pixel-on" encontrado ao varrer a matriz-imagem da direita para a esquerda e de cima para baixo é denominada de *cid*.

Considerando que podem existir irregularidades na tarja horizontal superior e que com isto as coordenadas *cie* e *cid* podem não corresponder aos extremos desta tarja, o programa retorna a estas coordenadas e varre a matriz-imagem para a esquerda, a partir da coordenada *cie* e para a direita, a partir da coordenada *cid*, até que não existam mais "pixels-on" que possam ser varridos nessas direções. Ao terminar de varrer a matriz-imagem, a coordenada do último "pixel-on" varrido, corresponderá ao extremo da tarja horizontal superior e será considerada como sendo a nova coordenada *cie*, se a matriz-imagem foi varrida para a esquerda e a nova coordenada *cid*, se a matriz-imagem foi varrida para a direita.

Para marcar os "pixels-on" que constituem a tarja horizontal superior, o programa varre a matriz-imagem para a direita, a partir da coordenada *cie*. Enquanto a matriz-imagem é varrida para a direita, o programa marca os "pixels-on" varridos e armazena a quantidade de "pixels-on" que são marcados. A

matriz-imagem é varrida para a direita, até que não existam mais "pixels-on" que possam ser varridos nessa direção.

Ao terminar de varrer a matriz-imagem para a direita, se o número de "pixels-on" marcados é maior ou igual a 40, o programa considera que estes "pixels-on" constituem a tarja horizontal superior ou a parte esquerda dela, caso esta tarja seja descontínua. Os "pixels-on" marcados são apagados da matriz-imagem. Se o número de "pixels-on" marcados é menor do que 40, o programa desmarca estes "pixels-on" e iguala a coordenada *cie* a -1. Ao desmarcar os "pixels-on" marcados e igualar a coordenada *cie* a -1, o programa está considerando que no campo que envolve a quantia numérica não existe a tarja horizontal superior ou a parte esquerda dela, caso esta tarja seja descontínua.

Após varrer a matriz-imagem para a direita, o programa varre esta matriz para a esquerda a partir da coordenada *cid*. Enquanto a matriz-imagem é varrida para a esquerda, o programa marca os "pixels-on" que são varridos e armazena a quantidade de "pixels-on" que são marcados. A matriz-imagem é varrida para a esquerda, até que não existam mais "pixels-on" que possam ser varridos nessa direção.

Se o número de "pixels-on" marcados é maior ou igual a 40, o programa considera que estes "pixels-on" constituem a tarja horizontal superior ou a parte direita dela, caso esta tarja seja descontínua. Os "pixels-on" marcados são apagados da matriz-imagem. Se o número de "pixels-on" marcados é menor do que 40, o programa desmarca estes "pixels-on" e iguala a coordenada *cid* a -1. Ao desmarcar os "pixels-on" marcados e igualar a coordenada *cid* a -1, o programa está considerando que no campo que envolve a quantia numérica não existe a parte direita da tarja horizontal superior, pois se esta tarja fosse contínua, o programa a teria apagado ao varrer a matriz-imagem para a direita.

Tarja-horiz-inf

A função tarja-horiz-inf tem como objetivo verificar se na matriz-imagem existe a tarja horizontal inferior. Esta tarja está localizada na parte inferior da matriz-imagem. A matriz-imagem armazena os "pixels-on" e os "pixels-off" do esqueleto da imagem da área reservada à escrita da quantia numérica de um cheque bancário.

Para facilitar o entendimento da função tarja-horiz-inf, optou-se por explicá-la no texto e esboçar o seu fluxograma. Aconselha-se ao leitor que para sua melhor clareza, leia o texto que descreve a função tarja-horiz-inf e acompanhe ao mesmo tempo o seu fluxograma. A figura 5.17 que se encontra nas páginas 690 a 704 do 2º volume deste trabalho, ilustra o fluxograma da função tarja-horiz-inf. Nesta figura, alguns comentários que se encontram no texto não constam do fluxograma. Devido à extensão do fluxograma ele foi dividido em 14 partes.

Para verificar se na matriz-imagem existe a tarja horizontal inferior, o programa varre esta matriz da esquerda para a direita e de baixo para cima. A matriz-imagem é varrida da esquerda para a direita e de baixo para cima, até que seja encontrado um "pixel-on" ou a coordenada cuja ordenada é menor ou igual a 60% do número total de linhas da matriz-imagem (este passo será referenciado adiante como PASSO 1).

Ao varrer a matriz-imagem da esquerda para a direita e de baixo para cima, se é encontrado uma coordenada cuja a ordenada é menor ou igual a 60% do número total de linhas da matriz-imagem, o programa considera que no campo que envolve a área reservada à escrita da quantia numérica não existe a tarja horizontal inferior. Caso contrário, se ao varrer a matriz-imagem de baixo para cima e da esquerda para a direita é encontrado um "pixel-on", o programa denomina a coordenada deste "pixel-on" de *cee*.

A partir da coordenada *cee*, o programa varre a matriz-imagem para a direita, até que não existam mais "pixels-on" que possam ser varridos nessa direção ou o número de "pixels-on" varridos para a direita seja maior ou igual a 40.

Enquanto a matriz-imagem é varrida para a direita ou para a esquerda, se o programa verificar que (((o último "pixel-on" varrido encontra-se na direção 2 ou 6) e (o "pixel-on" que está sendo varrido encontra-se na direção diagonal (direção 1 ou 7)) ou ((o último "pixel-on" varrido encontra-se na direção 2) e (o valor da variável **"condição-horizontal"** é menor do que o valor que esta variável possuía antes do "pixel-on" que está sendo varrido ser varrido))), o programa armazena na variável **"condição-horizontal"** o valor que esta variável possuía antes do "pixel-on" que está sendo varrido ser varrido, incrementado de uma unidade. Se o valor da variável **"condição-horizontal"** é maior ou igual a 3 e a matriz-imagem está sendo varrida para a direita, o programa finaliza o processo de varrer a matriz-imagem nessa direção.

Se ((não existem mais "pixels-on" que possam ser varridos na matriz-imagem para a direita ou para a esquerda) ou (a matriz-imagem é varrida para a esquerda e o número de "pixels-on" varridos nessa direção é maior ou igual a 40)), o programa verifica se após ter sido varrido um "pixel-on" que se encontra na direção vertical (direção 2 ou 6) foram varridos 2 ou menos "pixels-on" que se encontram na direção horizontal (direção (0, 1 ou 7) ou (4, 3 ou 5)). Em caso afirmativo, o programa retorna ao "pixel-on" varrido que se encontra na direção vertical (direção 2 ou 6), considera este "pixel-on" varrido como sendo o último "pixel-on" varrido para a direita ou para a esquerda e finaliza o processo de varrer a matriz-imagem para a direita ou para a esquerda.

Se o número de "pixels-on" varridos para a direita na matriz-imagem é menor do que 40, o programa considera que os "pixels-on" varridos não constituem a tarja horizontal inferior. Visando encontrar esta tarja na matriz-

imagem, o programa executa novamente o PASSO 1, até encontrar um novo "pixel-on" cuja coordenada é diferente da coordenada *cee*. A coordenada do novo "pixel-on" encontrado é considerada como sendo a nova coordenada *cee* e o programa executa os procedimentos subseqüentes ao PASSO 1. Se o número de "pixels-on" varridos para a direita na matriz-imagem é maior ou igual a 40, o programa considera que na matriz-imagem existe a tarja horizontal inferior, retorna a coordenada *cee* e varre essa matriz para a esquerda, até que não existam mais "pixels-on" que possam ser varridos nessa direção. Com o objetivo de varrer a matriz-imagem para a esquerda, o programa realiza os procedimentos subseqüentes ao PASSO 1.

Ao terminar de varrer a matriz-imagem para a esquerda, a coordenada do último "pixel-on" varrido nessa direção é considerada como sendo a nova coordenada *cee* e o programa calcula e armazena na variável **lmed** a média da ordenada de todos os "pixels-on" varridos na matriz-imagem para a direita ou para a esquerda.

Após verificar que na matriz-imagem existe a tarja horizontal inferior, com o objetivo de apagar esta tarja da matriz-imagem, o programa varre esta matriz para a direita a partir da coordenada *cee*. A matriz-imagem é varrida para a direita, até que não existam mais "pixels-on" que possam ser varridos nessa direção.

Enquanto a matriz-imagem é varrida para a direita, o programa realiza os seguintes procedimentos:

- Se a variável **"condição-de-folga"** possuía valor igual a 0 antes do "pixel-on" que está sendo varrido ser varrido e enquanto este "pixel-on" está sendo varrido os valores das variáveis **"condição-de-folga"** e **"condição-horizontal"** são iguais a 0, o "pixel-on" varrido antes do "pixel-on" que está sendo varrido é marcado com o valor 10, por ser considerado pelo programa como

pertencente a tarja horizontal inferior.

- Se o "pixel-on" que está sendo varrido é um ponto de cruzamento, o programa denomina a coordenada deste ponto de cruzamento de *pc-ini* e executa a função verifica-tarja. Esta função será explicada adiante neste capítulo. A função verifica-tarja tem como objetivo evitar que os caracteres que se encontram conectados a tarja horizontal inferior tenham alguns dos seus trechos apagados, quando esta tarja for apagada da matriz-imagem. A função verifica-tarja retorna à função tarja-horiz-inf a coordenada do último ponto de cruzamento encontrado (denominado de *pce* na função verifica-tarja) e a coordenada do "pixel-on" a partir do qual o programa continuará a varrer a tarja horizontal inferior para a direita. Esse "pixel-on" é denominado de *pcr*. Se o valor da ordenada do último ponto de cruzamento encontrado pela função verifica-tarja é maior ou igual a 90% do valor contido na variável **lmed**, o programa armazena na variável **pc-ret** a coordenada desse ponto de cruzamento. A variável **pc-ret** armazena a coordenada do ponto de cruzamento comum a tarja horizontal inferior e a um caractere que se encontra conectado a esta tarja.

- Quando não existirem mais "pixels-on" que possam ser varridos para a direita, se na variável **pc-ret** foi armazenado a coordenada de um ponto de cruzamento, o programa verifica se ((a abscissa do último "pixel-on" varrido para a direita é menor do que 85% do número total de colunas da matriz-imagem) ou ((a abscissa do último "pixel-on" varrido para a direita é maior ou igual a 95% do número total de colunas da matriz-imagem) e (a ordenada desse "pixel-on" é menor do que 85% do valor contido na variável **lmed**))). Em caso afirmativo, o programa verifica se o ponto de cruzamento cuja coordenada foi armazenada na variável **pc-ret** possui algum "pixel-on" vizinho. Se isto

ocorrer, o programa varre na matriz-imagem a região compreendida entre a coordenada formada pela ordenada do último "pixel-on" varrido para a direita e a abscissa da coordenada do ponto de cruzamento armazenada na variável **pc-ret** e a coordenada formada pela ordenada do ponto de cruzamento armazenada na variável **pc-ret** e a abscissa do último "pixel-on" varrido para a direita (esta região será denominada adiante como REGIÃO 1). Enquanto o programa varre a REGIÃO 1, os "pixels-on" que foram marcados anteriormente com o valor 10 são marcados com o valor 6. O objetivo da utilização do valor 6 será explicado adiante neste capítulo. Após varrer a REGIÃO 1, o programa continua a varrer a matriz-imagem para a direita, a partir da coordenada do ponto de cruzamento armazenada na variável **pc-ret**. A coordenada armazenada na variável **pc-ret** é apagada desta variável, com o objetivo de que seja possível armazenar nesta variável a coordenada de outro ponto de cruzamento.

- Enquanto a matriz-imagem é varrida para a direita, se ((a coordenada do "pixel-on" que está sendo varrido coincide com a coordenada do "pixel-on" *pcr*) e (o "pixel-on" que se encontra a direita do "pixel-on" que está sendo varrido possui "pixels-on" vizinhos que se encontram nas direções 2 e 7)), o "pixel-on" que está sendo varrido é marcado com o valor 8. O objetivo da utilização do valor 8 é explicado adiante nesta função.

- Quando não existirem mais "pixels-on" que possam ser varridos para a direita, se o número de "pixels-on" varridos nessa direção é menor do que 40, os "pixels-on" que foram marcados com o valor 10 são marcados com o valor 5 e o programa continua a varrer a matriz-imagem para a direita, a partir da coordenada *cee*. Se o número de "pixels-on" varridos para a direita é maior ou igual a 40, o program a finaliza o processo de varrer a matriz-imagem nessa

direção.

Após terminar de varrer a matriz-imagem para a direita, o programa varre esta matriz em toda a sua extensão, da esquerda para a direita e de baixo para cima. Enquanto a matriz-imagem é varrida da esquerda para a direita e de baixo para cima, se ((o programa encontrar um "pixel-on" marcado com o valor 8 ou 10) e (o "pixel" (este "pixel" pode ser "on" ou "off") que se encontra abaixo desse "pixel-on" marcado possui valor diferente de 8 ou 10)), o programa apaga todos os "pixels-on" que se encontram na mesma coluna do "pixel-on" marcado e abaixo dele. Se o "pixel-on" marcado possui o valor 10, ele também é apagado da matriz-imagem, pois faz parte da tarja horizontal inferior.

Ao terminar de varrer a matriz-imagem da esquerda para a direita e de baixo para cima, a coordenada do último "pixel-on" marcado com o valor 8 ou 10 e varrido nesse processo é denominada de *cdd* e os "pixels-on" marcados com o valor 5, 6 ou 8 são desmarcados, pois estes "pixels-on" não constituem a tarja horizontal inferior.

Com o objetivo de verificar se na matriz-imagem existe a tarja vertical esquerda, o programa varre essa matriz para cima, a partir da coordenada *cee*. Enquanto a matriz-imagem é varrida para cima, o programa armazena o número de "pixels-on" varridos. A matriz-imagem é varrida para cima, até que não existam mais "pixels-on" que possam ser varridos nessa direção ou o número de "pixels-on" varridos seja maior ou igual a 10.

Com o objetivo de verificar se na matriz-imagem existe a tarja vertical direita, o programa varre essa matriz para cima, a partir da coordenada *cdd*. Enquanto a matriz-imagem é varrida para cima, o programa armazena o número de "pixels-on" varridos. A matriz-imagem é varrida para cima, até que não existam mais "pixels-on" que possam ser varridos nessa direção ou o número de "pixels-on" varridos seja maior ou igual a 10.

Ao varrer a matriz-imagem para cima a partir da coordenada *cee*, se o número de "pixels-on" varridos nessa direção é maior ou igual a 10, o programa considera que provavelmente existe na matriz-imagem a tarja vertical esquerda e que a coordenada *cee* é a coordenada inicial desta tarja. Caso contrário, a coordenada *cee* é igualada a -1, indicando que provavelmente na matriz-imagem não existe a tarja vertical esquerda.

Ao varrer a matriz-imagem para cima a partir da coordenada *cdd*, se o número de "pixels-on" varridos nessa direção é maior ou igual a 10, o programa considera que provavelmente existe na matriz-imagem a tarja vertical direita e que a coordenada *cdd* é a coordenada inicial desta tarja. Caso contrário, a coordenada cdd é igualada a -1, indicando que provavelmente na matriz-imagem não existe a tarja vertical direita.

A seguir é descrito a função verifica-tarja.

Verifica-tarja

A função verifica-tarja tem como objetivo evitar que os caracteres que se encontram conectados à tarja horizontal inferior tenham alguns dos seus trechos apagados, quando esta tarja for apagada da matriz-imagem. A função verifica-tarja retorna à função que a chamou a coordenada do último ponto de cruzamento encontrado e a coordenada do "pixel-on" a partir do qual o programa continuará a varrer a tarja horizontal inferior para a direita. O programa denomina de *pcr* o "pixel-on" a partir do qual a tarja horizontal inferior continuará a ser varrida para a direita. A matriz-imagem armazena os "pixels-on" e os "pixels-off" do esqueleto da imagem da área reservada à escrita da quantia numérica de um cheque bancário.

Para facilitar o entendimento da função verifica-tarja, optou-se por explicá-la no texto e esboçar o seu fluxograma. Aconselha-se ao leitor que para sua

melhor clareza, leia o texto que descreve a função verifica-tarja e acompanhe ao mesmo tempo o seu fluxograma. A figura 5.18 que se encontra nas páginas 705 a 720 do 2º volume deste trabalho, ilustra o fluxograma da função verifica-tarja. Nesta figura, alguns comentários que se encontram no texto não constam do fluxograma. Devido à extensão do fluxograma ele foi dividido em 15 partes.

A partir do ponto de cruzamento encontrado na tarja horizontal inferior, denominado no programa de *pc-ini*, a função verifica-tarja inicia o processo de varrer a tarja horizontal inferior. O objetivo de varrer a tarja horizontal inferior a partir do ponto de cruzamento *pc-ini* é verificar se este ponto de cruzamento pertence a esta tarja ou a um caractere que está conectado a ela.

O processo de varrer a tarja horizontal inferior é realizado inicialmente igualando as variáveis **ud** e **lr** a "up" e "right", respectivamente, as variáveis **"condição-horizontal"**, **"condição-de-folga"** e **conc** a 0 e considerando que podem existir "pixels-on" que se encontram na direção horizontal (direção 7, 0, 1, 3, 4 ou 5) ou na direção vertical (direção 1, 2, 3, 5, 6 ou 7). O objetivo das variáveis **ud**, **lr** e **conc** é explicado adiante neste capítulo.

Ao varrer a tarja horizontal inferior ou o caractere que possui "pixels-on" em comum com esta tarja, o programa verifica se os valores das variáveis **"condição-horizontal"** e **conc** são iguais a 0. Em caso afirmativo, o programa executa a função vertical, explicada no capítulo 4 deste trabalho. A função vertical, varre a matriz-imagem para cima ou para baixo, conforme o valor contido na variável **ud**. Se o valor contido na variável **ud** é igual a "up", a função vertical varre a matriz-imagem para cima e se o valor contido na variável **ud** é "down", o programa varre a matriz-imagem para baixo. A variável **conc** indica se os "pixels-on" que estão sendo varridos pertencem a concavidade do caractere conectado à tarja horizontal inferior ou a esta tarja.

Após o programa executar a função vertical, se o valor da variável **"condição-de-folga"** é diferente de 0, é verificado se o "pixel-on" que foi varrido

nessa função possui algum "pixel-on" vizinho. Em caso afirmativo, a variável **"condição-horizontal"** é igualada a 1 e as variáveis **"condição-de-folga"** e **indi-fim** são igualadas a 0. Se o valor da variável **ud** é igual a "up" (direção 1, 2 ou 3), o programa varre o "pixel" (este "pixel" pode ser "on" ou "off") que possui abscissa igual a abscissa do "pixel-on" que foi varrido pela função vertical e ordenada igual a ordenada deste "pixel-on" incrementada de uma unidade. Caso contrário, o programa varre o "pixel" (este "pixel" pode ser "on" ou "off") que possui abscissa igual a abscissa do "pixel-on" que foi varrido pela função vertical e ordenada igual a ordenada deste "pixel-on" decrementada de uma unidade. Se o "pixel-on" que foi varrido pela função vertical não possui "pixel-on" vizinho, o programa verifica se o valor da variável **indi-fim** é igual a 0 e a função vertical indica que ainda existem "pixels-on" que possam ser varridos. Em caso afirmativo, o valor da variável **indi-fim** é incrementado de uma unidade.

Após o programa executar a função vertical, se o valor da variável **"condição-de-folga"** é igual a 0, o programa verifica se o valor da variável **indi-fim** é diferente de 0 e a função vertical indica que não existem mais "pixels-on" que possam ser varridos, a variável **indi-fim** é igualada a 0. A variável **indi-fim** possibilita ao programa continuar a varrer a tarja horizontal inferior ou o caractere conectado a esta tarja, quando é encontrada uma falha de um "pixel-on".

Ao varrer a tarja horizontal inferior ou o caractere que possui "pixels-on" em comum com esta tarja, se o valor da variável **"condição-horizontal"** ou **conc** é diferente de 0, o programa realiza os procedimentos descritos a seguir:

- A variável **"condição-horizontal"** é igualada a 0 e o programa executa a função horizontal, explicada no capítulo 4 deste trabalho. A função horizontal, varre a matriz-imagem para a esquerda ou para a direita, conforme o valor contido na variável **lr**. Se o valor contido na variável **lr** é igual a "left", a

função horizontal varre a matriz-imagem para a esquerda, se o valor contido na variável **lr** é "right", o programa varre a matriz-imagem para a direita.

- Se o valor da variável **"condição-horizontal"** é diferente de 0, esta variável é igualada a 0 e se o valor da variável **conc** é igual a 0, o programa verifica se o "pixel-on" que foi varrido pela função horizontal encontra-se na direção 6. Em caso afirmativo, a variável **ud** é igualada a "down", indicando que quando a função vertical for executada, esta função deve varrer os "pixels-on" para baixo. Se o "pixel-on" que foi varrido pela função horizontal encontra-se na direção 2, a variável **ud** é igualada a "up", indicando que quando a função vertical for executada, esta função deve varrer os "pixels-on" para cima.

- Se o valor da variável **"condição-de-folga"** é diferente de 0, o valor da variável **indi-fim** é igual a 0 e existem "pixels-on" que podem ser varridos pela função horizontal, as variáveis **"condição-horizontal"** e **indi-fim** são igualadas a 1. Caso contrário, se o valor da variável **indi-fim** é diferente de 0 e não existem mais "pixels-on" que possam ser varridos pela função horizontal, a variável **ud** é igualada a "down", as variáveis **indi-fim** e **"condição-horizontal"** são igualadas a 0 e o programa continua a varrer a tarja horizontal inferior ou o caractere conectado a esta tarja na direção horizontal ou na direção vertical, conforme os "pixels-on" estejam dispostos. A tarja horizontal inferior ou o caractere conectado a esta tarja é varrido na direção horizontal ou na direção vertical, até que não existam mais "pixels-on" que possam ser varridos em nenhuma dessas direções ou o valor da variável **indi-pc** seja diferente de 0. O objetivo da variável **indi-pc** será descrito adiante neste capítulo.

- Se o valor da variável **indi-fim** é igual a 0 e ainda existem "pixels-on" que

possam ser varridos pela função horizontal, o programa verifica se o valor da variável **conc** é diferente de 0. Em caso afirmativo, são realizados os procedimentos descritos a seguir:

- O programa retorna ao "pixel-on" varrido antes do "pixel-on" que foi varrido pela função horizontal e apaga da matriz-imagem os "pixels-on" que se encontram abaixo desse "pixel-on" e na mesma coluna. O programa denomina de *lc* a coordenada do "pixel-on" varrido antes do "pixel-on" que foi varrido pela função horizontal.

- O programa verifica se a coordenada *lc* coincide com a coordenada armazenada na variável **pcm**. Em caso afirmativo, o "pixel-on" cuja coordenada está armazenada na variável **pcm** é considerado como tendo sido varrido pela função horizontal. A variável **pcm** armazenada a coordenada de um ponto de cruzamento que se encontra abaixo de 90% do valor contido na variável **lmed**.

- O programa verifica se a coordenada do "pixel-on" que foi varrido coincide com a coordenada armazenada na variável **pcm**. Em caso afirmativo, a variável **indi-pc** é igualada a 1, o "pixel-on" que foi varrido é marcado com o valor 5, os "pixels-on" que se encontram na mesma coluna e nas linhas abaixo do "pixel-on" que foi varrido são apagados da matriz-imagem e o programa considera que não existem mais "pixels-on" que possam ser varridos na direção horizontal ou na direção vertical. O programa considera que os "pixels-on" marcados com o valor 5 não constituem a tarja horizontal inferior.

- Se apenas a abscissa do "pixel-on" que foi varrido coincide com a

abscissa da coordenada armazenada na variável **pcm**, as variáveis **conc**, **"condição-de-folga"** e **"condição-horizontal"** são igualadas a 0, a variável **ud** é igualada a "down", o "pixel-on" que foi varrido é denominado de *pcr* e o programa retorna a coordenada armazenada na variável **pcm**. O programa denomina de *pcr* o "pixel-on" a partir do qual a tarja horizontal inferior continuará a ser varrida para a direita, após executar a função verifica-tarja.

- Enquanto o programa varre a tarja horizontal inferior ou o caractere conectado a esta tarja utilizando a função horizontal ou a função vertical, se ((o valor da variável **"condição-de-folga"** é igual a 0) ou (o valor que esta variável possuía antes do "pixel-on" que está sendo varrido ser varrido era diferente de 0)), o "pixel-on" que foi varrido antes do "pixel-on" que está sendo varrido é marcado com o valor 5. Se o "pixel-on" que está sendo varrido encontra-se na direção 1, 0 ou 7, o valor da variável **pix** é incrementado de uma unidade. Se o valor armazenado na variável **cmenor** é maior do que o valor da abscissa do "pixel-on" que está sendo varrido, o programa armazena na variável **cmenor** a abscissa desse "pixel-on". A variável **pix** armazena a quantidade de "pixels-on" varridos que se encontram na direção horizontal e a variável **cmenor** armazena a abscissa do "pixel-on" que se encontra no extremo esquerdo da tarja horizontal inferior ou do caractere conectado a esta tarja.

- Enquanto o programa está varrendo a tarja horizontal inferior ou o caractere conectado a esta tarja, se ((é encontrado um ponto de cruzamento, denominado no programa de *pc1*) e ((este ponto de cruzamento está abaixo de 90% do valor contido na variável **lmed)** ou (o valor da variável **conc** é diferente de 0)) e (a coordenada do ponto de cruzamento *pc1* é diferente da coordenada do ponto de cruzamento *pc-ini*)), são executados os procedimentos descritos a

seguir:

- O programa iguala as variáveis **ud** e **"condição-horizontal"** a "down" e a 0, respectivamente.

- Se o valor da variável **conc** é igual a 0, o programa verifica se o ponto de cruzamento *pc1* possui "pixel-on" vizinho que se encontra na direção 6, 5, 7 ou 1. Em caso afirmativo, o programa marca o "pixel-on" do ponto de cruzamento *pc1* com o valor 5. Caso contrário, o programa continua a varrer a tarja horizontal inferior ou o caractere conectado a esta tarja na direção horizontal ou na direção vertical, a partir do ponto de cruzamento *pc1*.

- Se o valor da variável **pix** é maior do que 3, o programa verifica se o ponto de cruzamento *pc1* possui "pixel-on" vizinho que se encontra na direção 6, 5, 7 ou 1, nesta ordem. Em caso afirmativo, o programa continua a varrer a tarja horizontal inferior ou o caractere conectado a esta tarja na direção horizontal ou na direção vertical, a partir do "pixel-on" vizinho encontrado. Caso contrário, o programa continua a varrer a tarja horizontal inferior ou o caractere conectado a esta tarja na direção horizontal ou na direção vertical, a partir do ponto de cruzamento *pc1*.

- Se o valor da variável **pix** é menor ou igual a 3, o programa verifica se o ponto de cruzamento *pc1* possui "pixel-on" vizinho que se encontra na direção 6, 7, 5 ou 1, nesta ordem. Em caso afirmativo, o programa continua a varrer a tarja horizontal inferior ou o caractere conectado a esta tarja na direção horizontal ou na direção vertical, a

partir do "pixel-on" vizinho encontrado. Caso contrário, o programa continua a varrer a tarja horizontal inferior ou o caractere conectado a esta tarja, a partir do ponto de cruzamento *pc1*.

Se o valor da variável **conc** é diferente de 0, o programa verifica se o ponto de cruzamento *pc1* possui "pixel-on" vizinho que se encontra na direção 0, 7 ou 1, nesta ordem. Em caso afirmativo, o programa marca o "pixel-on" do ponto de cruzamento *pc1* com o valor 5 e continua a varrer a tarja horizontal inferior ou o caractere conectado a esta tarja, a partir do "pixel-on" vizinho encontrado. Caso contrário, o programa continua a varrer a tarja horizontal inferior ou o caractere conectado a esta tarja, a partir do ponto de cruzamento *pc1*.

- Enquanto o programa está varrendo a tarja horizontal inferior ou o caractere conectado a esta tarja, se ((o ponto de cruzamento *pc1* está acima de 90% do valor contido na variável **lmed)** e (o valor da variável **conc** é igual a 0)), o programa verifica se o ponto de cruzamento *pc1* encontra-se no intervalo compreendido entre 90% e 110% do valor contido na variável **lmed**, inclusive. Em caso afirmativo, o programa desmarca o "pixel-on" do ponto de cruzamento *pc1* e verifica se ((o valor da variável **pix** é maior do que 3) e ((a abscissa do ponto de cruzamento *pc1* encontra-se a uma distância de no máximo um "pixel-on" da abscissa do "pixel-on" *pcr***)** ou (a ordenada do ponto de cruzamento *pc1* encontra-se a uma distância de no máximo um "pixel-on" da ordenada do "pixel-on" *pcr*))). Se isto ocorrer, as variáveis **lr** e **conc** são igualadas a "right", a coordenada do ponto de cruzamento *pc1* é armazenada na variável **pcm** e o programa continua a varrer a tarja horizontal inferior ou o caractere conectado a esta tarja na direção horizontal ou na direção vertical, a partir do "pixel-on" *pcr*.

- Se ((não existem mais "pixels-on" que possam ser varridos pela função vertical ou pela função horizontal) ou (o valor da variável **indi-pc** é diferente de 0)), o programa finaliza o processo de varrer a tarja horizontal inferior ou o caractere conectado a esta tarja. Caso contrário, o programa continua a varrer a tarja horizontal inferior ou o caractere conectado a esta tarja na direção horizontal ou na direção vertical.

Ao finalizar o processo de varrer a tarja horizontal inferior ou o caractere conectado a esta tarja, se o último "pixel-on" varrido nesse processo possui "pixel-on" vizinho que se encontra na direção 0, 7, 1 ou 6 nesta ordem, o "pixel-on" vizinho encontrado será o "pixel-on" *pcr*, a partir do qual a função tarja-horiz-inf continuará a varrer a tarja horizontal inferior. Caso contrário, o "pixel-on" *pcr* será o último "pixel-on" varrido no processo de varrer a tarja horizontal inferior ou o caractere conectado a esta tarja.

Se o valor da variável **indi-pc** é diferente de 1, o programa denomina a coordenada do ponto de cruzamento *pc-ini* de *pce*, retorna à função tarja-horiz-inf a coordenada *pce* como sendo a coordenada do último ponto de cruzamento encontrado na função verifica-tarja, desmarca os "pixels-on" marcados com o valor 5 que se encontram na região compreendida entre ((a coordenada formada pela ordenada do ponto de cruzamento *pc-ini* dividida por 2 e a abscissa armazenada na variável **cmenor**) e (a coordenada formada pela ordenada cujo valor é igual a 130% do valor contido na variável **lmed** e a coluna final da matriz-imagem)) e o "pixel-on" do ponto de cruzamento *pc-ini* é marcado com o valor 8. O programa considera que os "pixels-on" marcados com o valor 8 não constituem a tarja horizontal inferior.

Se o valor da variável **indi-pc** é igual a 1, a função verifica-tarja retorna à função tarja-horiz-inf a coordenada *pce* como sendo a coordenada do último ponto de cruzamento encontrado e varre a matriz-imagem do seu extremo inicial

ao seu extremo final, marcando o "pixels-on" marcados com o valor 5 com o valor 8.

Tarja-vertical

A função tarja-vertical tem como objetivo verificar se no campo que envolve a área reservada à escrita da quantia numérica existe a tarja vertical esquerda e/ou a tarja vertical direita. As tarjas verticais direita e esquerda estão situadas, respectivamente, no extremo direito e no extremo esquerdo do campo que envolve a área reservada à escrita da quantia numérica.

Para facilitar o entendimento da função tarja-vertical, optou-se por explicá-la no texto e esboçar o seu fluxograma. Aconselha-se ao leitor que para sua melhor clareza, leia o texto que descreve a função tarja-vertical e acompanhe ao mesmo tempo o seu fluxograma. A figura 5.19 que se encontra nas páginas 721 a 733 do 2º volume deste trabalho, ilustra o fluxograma da função tarja-vertical. Nesta figura, alguns comentários que se encontram no texto não constam do fluxograma. Devido à extensão do fluxograma ele foi dividido em 12 partes.

Ao apagar os "pixels-on" da tarja horizontal inferior, se não foi encontrada a coordenada inicial da tarja vertical esquerda denominada na função tarja-horiz-inf de coordenada *cee* ou o valor desta coordenada é igual a -1, o programa varre a linha média da matriz-imagem da esquerda para a direita, até que o primeiro "pixel-on" seja encontrado. Na matriz-imagem são armazenados os "pixels-on" e os "pixels-off" do esqueleto da imagem da área reservada à escrita da quantia numérica de um cheque bancário.

Ao apagar os "pixels-on" da tarja horizontal inferior, se não foi encontrada a coordenada inicial da tarja vertical direita denominada na função tarja-horiz-inf de coordenada *cdd* ou o valor desta coordenada é igual a -1, o programa varre a linha média da matriz-imagem da direita para a esquerda, até que o primeiro

"pixel-on" seja encontrado.

Após encontrar o primeiro "pixel-on" varrendo a matriz-imagem da esquerda para a direita e/ou da direita para a esquerda, o programa varre essa matriz para baixo, até que não existam mais "pixels-on" que possam ser varridos nessa direção. Se a matriz-imagem foi varrida da esquerda para a direita, a coordenada do último "pixel-on" varrido para baixo é denominada de *cee*. Se a matriz-imagem foi varrida da direita para a esquerda, a coordenada do último "pixel-on" varrido para baixo é denominada de *cdd*.

Com o objetivo de delimitar a tarja vertical esquerda, o programa varre a matriz-imagem para cima, a partir da coordenada *cee*, até que não existam mais "pixels-on" que possam ser varridos nessa direção. Enquanto a matriz-imagem é varrida para cima, o programa marca com o valor 5 os "pixels-on" que são varridos, armazena a quantidade de 'pixels-on" varridos e a quantidade de "pixels-on" varridos que se encontram na direção diagonal (direção 1 ou 3). Ao marcar os "pixels-on" com o valor 5, o programa considera que estes "pixels-on" constituem a tarja vertical esquerda.

Ao iniciar o processo de varrer a matriz-imagem para cima, se o "pixel-on" da coordenada *cee* possui "pixel-on" vizinho que se encontra na direção 3, o programa varre a matriz-imagem para cima a partir desse "pixel-on" vizinho.

Se ao varrer a matriz-imagem para cima, ((o último "pixel-on" varrido encontra-se na direção 1 e o "pixel-on" que está sendo varrido encontra-se na direção 3) ou (o último "pixel-on" varrido encontra-se na direção 3 e o "pixel-on" que está sendo varrido encontra-se na direção 1)), o programa considera que o "pixel-on" que está sendo varrido encontra-se na direção vertical (direção 2) e decrementa a quantidade de "pixels-on" varridos na direção diagonal de 2 unidades. Ao decrementar a quantidade de "pixels-on" varridos na direção diagonal de 2 unidades, o programa considera que os 2 "pixels-on" encontrados na direção diagonal correspondem a apenas um "pixel-on" que se encontra na

direção vertical.

Enquanto o programa varre a matriz-imagem para cima, se o "pixel-on" que está sendo varrido encontra-se na direção diagonal (direção 1 ou 3) e o último "pixel-on" varrido encontra-se na direção horizontal (direção 0 ou 4), o programa iguala a variável **"condição-horizontal"** ao valor que esta variável possuía antes do "pixel-on" que está sendo varrido ser varrido, incrementado de uma unidade. Se ((o valor da variável **"condição-horizontal"** é maior ou igual a 3) ou (não existem mais "pixels-on" que possam ser varridos para cima)), o programa finaliza o processo de varrer a tarja vertical esquerda. Caso contrário, o programa continua a varrer a matriz-imagem para cima.

Após finalizar o processo de delimitação da tarja vertical esquerda, o programa inicia o processo de delimitação da tarja vertical direita. Este processo de delimitação da tarja vertical direita utiliza o mesmo procedimento que foi utilizado na delimitação da tarja vertical esquerda, porém, na delimitação da tarja vertical direita, o programa varre a matriz-imagem para cima a partir da coordenada *cdd* e não verifica se esta coordenada possui "pixel-on" vizinho que se encontra na direção 3.

Ao terminar de delimitar a tarja vertical esquerda, se (((a quantidade de "pixels-on" varridos para cima é maior ou igual a 23) e (a quantidade de "pixels-on" varridos que se encontram na direção diagonal é menor do que 40% da quantidade de "pixels-on" varridos para cima)) ou ((a coordenada *cie* encontrada ao delimitar a tarja horizontal superior encontra-se a uma distância menor ou igual a 5 "pixels-on" do último "pixel-on" varrido para cima) ou (o número de "pixels-on" varridos para cima quando o programa varreu a matriz-imagem a partir da coordenada *cee* na função tarja-horiz-inf é maior ou igual a 10))), o programa armazena a coordenada do último "pixel-on" varrido ao delimitar a tarja vertical esquerda como sendo a coordenada de um ponto de cruzamento e executa os procedimentos descritos a seguir para os pontos de cruzamento cujas

coordenadas foram armazenadas quando o programa delimitou a tarja vertical esquerda. Esses procedimentos serão referenciados como PROCESSO 1. Caso contrário, o programa considera que os "pixels-on" que foram varridos para cima não constituem a tarja vertical esquerda.

- Se algum dos pontos de cruzamento cuja coordenada foi armazenada encontra-se a uma distância menor ou igual a 4 "pixels-on" da coordenada do último "pixel-on" varrido na delimitação da tarja vertical esquerda, o programa retorna ao ponto de cruzamento cuja coordenada foi armazenada e varre a matriz-imagem para a esquerda, até que não existam mais "pixels-on" que possam ser varridos nessa direção. Se o número de "pixels-on" varridos para a esquerda é maior do que 3, o programa considera que os "pixels-on" que foram varridos para cima nos procedimentos anteriores ao PROCESSO 1 não constituem uma tarja vertical, mas sim o número 1.

- Se não foi encontrado ponto de cruzamento cuja coordenada foi armazenada e esta coordenada encontra-se a uma distância menor ou igual a 4 "pixels-on" da coordenada do último "pixel-on" varrido na delimitação da tarja vertical esquerda, o programa retorna a cada um dos pontos de cruzamento cuja coordenada foi armazenada e varre a matriz-imagem para a esquerda (direção 3, 4 ou 5) e/ou para cima (direção 1, 2 ou 3). A matriz-imagem é varrida para a esquerda e/ou para cima, até que não existam mais "pixels-on" que possam ser varridos nessas direções. Se ((o número de "pixels-on" varridos para a esquerda é maior ou igual a 4) e (o número de "pixels-on" varridos para cima é maior ou igual a 4)), o programa considera que os "pixels-on" que foram varridos para cima nos procedimentos anteriores ao PROCESSO 1 não constituem uma tarja vertical, mas sim o número 4.

Ao terminar de delimitar a tarja vertical direita, se (((a quantidade de "pixels-on" varridos para cima é maior ou igual a 23) e (a quantidade de "pixels-on" varridos que se encontram na direção diagonal é menor do que 40% da quantidade de "pixels-on" varridos para cima)) ou ((a coordenada *cid* encontrada ao delimitar a tarja horizontal superior encontra-se a uma distância menor ou igual a 5 "pixels-on" do último "pixel-on" varrido para cima) ou (o número de "pixels-on" varridos para cima quando o programa varreu a matriz-imagem a partir da coordenada *cdd* na função tarja-horiz-inf é maior ou igual a 10))), o programa armazena a coordenada do último "pixel-on" varrido ao delimitar a tarja vertical direita como sendo a coordenada de um ponto de cruzamento e executa os procedimentos que constituem o PROCESSO 1, considerando que as análises que foram executadas para a tarja vertical esquerda, serão agora executadas para a tarja vertical direita. Caso contrário, o programa considera que os "pixels-on" que foram varridos para cima nos procedimentos anteriores ao PROCESSO 1 não constituem a tarja vertical direita.

Se o programa considerou que os "pixels-on" varridos para cima a partir da coordenada *cee* e marcados com o valor 5 não constituem a tarja vertical esquerda ou constituem o número 1 ou 4, esses "pixels-on" devem continuar na matriz-imagem. Devido a isto, o programa varre a matriz-imagem em toda a sua extensão desmarcando esses "pixels-on". Caso contrário, os "pixels-on" que foram varridos para cima e marcados com o valor 5 constituem a tarja vertical esquerda e por isto esses "pixels-on" devem ser apagados da matriz-imagem. Com o objetivo de apagar a tarja vertical esquerda, o programa varre a matriz-imagem em toda a sua extensão, apagando os "pixels-on" marcados com o valor 5 que não possuem "pixel-on" vizinho que se encontra a sua direita (direção 0) e/ou a sua esquerda (direção 4).

Se o programa considerou que os "pixels-on" varridos para cima a partir da coordenada *cdd* e marcados com o valor 5 não constituem a tarja vertical direita

ou constituem o número 1 ou 4, esses "pixels-on" que foram varridos para cima e marcados com o valor 5 devem continuar na matriz-imagem. Devido a isto, o programa varre a matriz-imagem em toda a sua extensão desmarcando esses "pixels-on". Caso contrário, os "pixels-on" que foram varridos para cima e marcados com o valor 5 constituem a tarja vertical direita e por isto esses "pixels-on" devem ser apagados da matriz-imagem. Com o objetivo de apagar a tarja vertical direita, o programa varre a matriz-imagem em toda a sua extensão, apagando os "pixels-on" marcados com o valor 5 que não possuem "pixel-on" vizinho que se encontra a sua direita (direção 0) e/ou a sua esquerda (direção 4).

5.6 - ELIMINAÇÃO DA CERQUILHA

Várias pessoas ao preencherem a quantia numérica de cheques bancários, podem delimitar esta quantia, do lado esquerdo e/ou do lado direito por um caractere, denominado neste trabalho de cerquilha (#). Para que este caractere não interfira no processo de reconhecimento dos dígitos manuscritos de cheques bancários, optou-se por eliminá-lo.

A cerquilha é composta de 2 linhas verticais e 2 linhas horizontais. A figura 5.20 ilustra a área reservada à escrita da quantia numérica do cheque bancário em que esta quantia é delimitada por 2 cerquilhas.

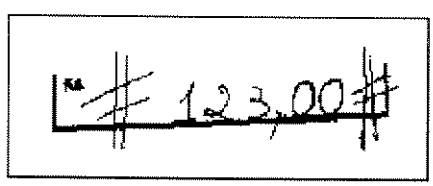


Figura 5.20 - Área reservada à escrita da quantia numérica do cheque bancário em que esta quantia é delimitada por 2 cerquilhas.

No programa a função que verifica a existência de cerquilhas na área reservada à escrita da quantia numérica do cheque bancário e realiza a eliminação

dessas cerquilhas, é denominada de verifica-cerquilha. A função verifica-cerquilha é descrita a seguir:

Verifica-cerquilha

A função verifica-cerquilha tem como objetivo verificar a existência de cerquilhas na área reservada à escrita da quantia numérica do cheque bancário e realizar a eliminação dessas cerquilhas.

Na função verifica-cerquilha foram utilizadas as variáveis **inie**, **inid**, **dir**, **last-dir**, **cruz, ccruz, lcruz** e **vert** (definida a seguir). As variáveis **inie** e **inid** armazenam, respectivamente, a coordenada inicial da linha vertical esquerda e a coordenada inicial da linha vertical direita que constituem a cerquilha, a variável **last-dir** armazena a direção do "pixel-on" que acabou de ser varrido, enquanto a variável **dir** armazena a direção do "pixel-on" que está sendo varrido, as variáveis **cruz** e **ccruz** indicam a existência de ponto de cruzamento, a variável **lcruz** armazena a ordenada do último ponto de cruzamento encontrado ao varrer a matriz-imagem para baixo. A matriz-imagem armazena os "pixels-on" e os "pixels-off" que constituem o esqueleto da imagem da área reservada à escrita da quantia numérica de um cheque bancário.

Para facilitar o entendimento da função verifica-cerquilha, optou-se por explicá-la no texto e esboçar o seu fluxograma. Aconselha-se ao leitor que para sua melhor clareza, leia o texto que descreve a função verifica-cerquilha e acompanhe ao mesmo tempo o seu fluxograma. A figura 5.21 que se encontra nas páginas 734 a 759 do 2º volume deste trabalho, ilustra o fluxograma da função verifica-cerquilha. Nesta figura, alguns comentários que se encontram no texto não constam do fluxograma. Devido à extensão do fluxograma ele foi dividido em 25 partes.

Antes de iniciar o processo de varrer a matriz-imagem, o programa iguala

as variáveis **dir**, **last-dir**, **cruz**, **ccruz**, **lcruz** e **vert** a 0.

Para verificar se o caractere é uma cerquilha, o programa realiza os seguintes procedimentos:

- Se o valor da variável **dir** é diferente de -1, a matriz-imagem é varrida da esquerda para a direita e de cima para baixo, até que um "pixel-on" seja encontrado (este passo será referenciado adiante como PASSO 1).

Se a variável **inie** não armazena a coordenada de algum "pixel-on" varrido, a coordenada do "pixel-on" encontrado no PASSO 1 é armazenada nessa variável. Caso contrário, se ((a variável **inid** não armazena a coordenada de algum "pixel-on" varrido) ou ((o valor da variável **last-dir** é igual a -1) e (a variável **inid** armazena uma coordenada diferente da coordenada do "pixel-on" encontrado no PASSO 1))), o programa armazena na variável **inid** a coordenada do "pixel-on" encontrado no PASSO 1. Se isto não ocorrer, a variável **vert** é igualada a "right". Se o valor da variável **vert** é igual a "right", o programa está varrendo a linha vertical direita da cerquilha, se o valor da variável **vert** é igual a "left", o programa está varrendo a linha vertical esquerda da cerquilha ou se o programa não definiu qual a linha vertical da cerquilha está sendo varrida, o valor da variável **vert** é igual a "?". A variável **vert** indica a linha vertical da cerquilha que está sendo varrida.

- Ao analisar o processo utilizado para varrer a matriz-imagem no PASSO 1, é observado que o primeiro "pixel-on" encontrado encontra-se em uma das linhas verticais da cerquilha. Devido a isto, a variável **dir** é igualada a 6.

- A partir do primeiro "pixel-on" encontrado no PASSO 1, a matriz-imagem é varrida para baixo, até que não existam mais "pixels-on" que possam ser varridos nessa direção, não existam mais coordenadas de pontos de

cruzamento armazenadas ou o valor da variável **cruz** seja diferente de 0. Para cada "pixel-on" que está sendo varrido para baixo (este passo será referenciado adiante como PASSO 2), os seguintes procedimentos são realizados:

- Se ((o "pixel-on" que está sendo varrido encontra-se na direção 5 ou 7 (valor da variável **dir** é igual a 5 ou 7)) e (o "pixel-on" varrido antes do "pixel-on" que está sendo varrido encontra-se na direção 0, 4 ou 6 (valor da variável **last-dir** é igual a 0, 4 ou 6))), a variável **dir** é igualada ao valor contido na variável **last-dir**. Se o valor da variável **last-dir** é igual a 0 ou 4, o valor da variável **"condição-horizontal"** é incrementado de uma unidade.

- Se os valores das variáveis **"condição-de-folga"** e **condição-horizontal"** são iguais a 0, a variável **pix** é incrementada de uma unidade e o "pixel-on" que está sendo varrido é marcado com o valor 5. A variável **pix** foi definida no capítulo 4 deste trabalho e o valor 5 é utilizado para marcar os "pixels-on" varridos, que provavelmente pertencem à cerquilha.

- Se (((o valor da variável **dir** é igual a 0 ou 4) ou (o valor da variável **pix** é igual a 1) ou (o valor absoluto da diferença entre a ordenada do "pixel-on" que está sendo varrido e o valor da ordenada armazenada na variável **lcruz** é menor ou igual a 2)) e (o valor da variável **"condição-de-folga"** é igual a 0)), o "pixel-on" que está sendo varrido é marcado com o valor 9. O valor 9 é utilizado para marcar os "pixels-on" que se encontram próximos (diferença de 1 "pixel-on") de um ponto de cruzamento.

- Ao encontrar um ponto de cruzamento, o programa executa os 6 procedimentos descritos a seguir:

1 - O "pixel-on" que está sendo varrido ("pixel-on" do ponto de cruzamento) é marcado com o valor 9 e a sua ordenada é armazenada na variável **lcruz**.

2 - Se ((o valor da variável **ccruz** é diferente de 0) ou (o valor desta variável é igual a 0 e o valor da variável **vert** é igual a "?")), as variáveis **cruz** e **ccruz** são igualadas a 0. Se a abscissa armazenada na variável **inie** é menor do que a abscissa armazenada na variável **inid**, a variável **vert** é igualada a "right", indicando que a linha vertical direita da cerquilha está sendo varrida. Se a abscissa armazenada na variável **inie** é maior ou igual a abscissa armazenada na variável **inid**, a variável **vert** é igualada a "left", indicando que a linha vertical esquerda da cerquilha está sendo varrida. Se (((o valor que a variável **vert** possuía era igual a "?") ou (o valor da variável **ccruz** é diferente de 0)) e (o valor que a variável **vert** possui é igual a "right")), a coordenada do ponto de cruzamento encontrado é armazenada na variável **pc-dir**. A variável **pc-dir** armazena o primeiro ponto de cruzamento encontrado quando a linha vertical direita da cerquilha é varrida de cima para baixo.

3 - Se a variável **pc-dir** não armazena a coordenada de nenhum ponto de cruzamento, a coordenada do ponto de

cruzamento encontrado é armazenada nessa variável.

4 - Se ((não foi definida qual das linhas verticais da cerquilha está sendo varrida (variável **vert** com valor igual a 0)) e (o ponto de cruzamento encontrado possui "pixels-on" vizinhos que se encontram nas direções 5 e 7)), as variáveis **ccruz** e **cruz** são igualadas a 1, a coordenada do ponto de cruzamento encontrado é armazenada na variável **pc-vert**, o número de "pixels-on" varridos que estão armazenados na variável **pix** é armazenado na variável **pix-vert** e a variável **pix** é igualada a 0. As variáveis **pc-vert** e **pix-vert** armazenam, respectivamente, a coordenada do ponto de cruzamento encontrado e o número de "pixels-on" varridos até este ponto de cruzamento ser encontrado.

5 - Se ((o valor da variável **vert** é diferente de 0) e (o ponto de cruzamento encontrado possui "pixels-on" vizinhos que se encontram nas direções 5 e 7)), a coordenada do ponto de cruzamento encontrado é armazenada e se o valor da variável **vert** é igual a "left", o programa continua a varrer a matriz-imagem para baixo a partir do "pixel-on" vizinho que se encontra na direção 5. Caso contrário, o programa continua a varrer a matriz-imagem para baixo a partir do "pixel-on" vizinho que se encontra na direção 7.

6 - Se ((o valor da variável **vert** é diferente de 0) e (o ponto

de cruzamento encontrado não possui "pixels-on" vizinhos que se encontram nas direções 5 e 7)), o programa verifica se o ponto de cruzamento encontrado possui "pixel-on" vizinho que se encontra na direção 6. Em caso afirmativo, se o valor da variável **vert** é igual a "left" e o ponto de cruzamento encontrado possui "pixel-on" vizinho que se encontra na direção 4, o programa continua a varrer a matriz-imagem para baixo a partir do "pixel-on" vizinho que se encontra na direção 4. Caso contrário, se o valor da variável **vert** é igual a "right" e o ponto de cruzamento encontrado possui "pixel-on" vizinho que se encontra na direção 0, o programa continua a varrer a matriz-imagem para baixo a partir do "pixel-on" vizinho que se encontra na direção 0. Se ((o ponto de cruzamento encontrado não possui "pixel-on" vizinho que se encontra na direção 6) e (o valor da variável **vert** é diferente de "left" ou o ponto de cruzamento encontrado não possui "pixel-on" vizinho que se encontra na direção 4) e (o valor da variável **vert** é diferente de "right" ou o ponto de cruzamento encontrado não possui "pixel-on" vizinho que se encontra na direção 0), o programa continua a varrer a matriz-imagem para baixo a partir do ponto de cruzamento encontrado.

- Ao terminar de varrer a matriz-imagem para baixo, o programa verifica se ((o valor absoluto da diferença entre a ordenada do último "pixel-on" varrido para baixo e a ordenada do extremo inferior do

caractere é maior ou igual a 9) e (enquanto a matriz-imagem estava sendo varrida para baixo, foi armazenada a coordenada de algum ponto de cruzamento)), a variável **dir** é igualada a 5, o programa retorna ao ponto de cruzamento cuja coordenada foi armazenada e continua a varrer a matriz-imagem para baixo a partir desse ponto de cruzamento. O programa considera como extremo inferior do caractere o "pixel-on" cuja coordenada encontra-se na parte inferior da matriz-imagem e não possui mais "pixels-on" vizinhos que possam ser varridos para baixo. A variável **dir** foi igualada a 5, com o objetivo de que quando o ponto de cruzamento cuja coordenada foi armazenada possuir "pixels-on" vizinhos que se encontram nas direções 5 e 7, a direção 5 tenha preferência sobre a direção 7.

- Se ((não existem mais "pixels-on" que possam ser varridos para baixo) ou (não existem pontos de cruzamento cujas as coordenadas foram armazenadas) e (o valor da variável **cruz** é diferente de 0)), o programa finaliza o processo de varrer a matriz-imagem para baixo. Caso contrário, o programa continua a varrer a matriz-imagem para baixo a partir do ponto de cruzamento cuja coordenada foi armazenada.

- Se não existem mais "pixels-on" que possam ser varridos para baixo e pontos de cruzamento cujas as coordenadas foram armazenadas, o programa verifica se o valor da variável **ccruz** é diferente de 0. Em caso afirmativo, a variável **last-dir** é igualada a -1, indicando que foi encontrado um ponto de cruzamento e que quando o programa voltou a varrer a matriz-imagem para baixo não foi encontrada uma das linhas verticais da cerquilha. Caso contrário, o programa verifica se o valor da variável **vert** é igual a 0. Se isto ocorrer, a variável **vert** é igualada a "?", a variável **pix-vert** é igualada ao valor contido na variável **pix** e

esta última variável é igualada a 0.

Se o valor da variável **vert** é diferente de 0, o programa verifica se ((foi armazenado na variável **pc-vert** a coordenada de algum ponto de cruzamento) e (o valor da variável **pix** é maior do que 12)). Em caso afirmativo, o programa substitui os valores contidos nas variáveis **pix** e **pix-vert** entre si, a variável **dir** é igualada a -1, a ordenada do ponto de cruzamento armazenada na variável **pc-vert** é armazenada na variável **lcruz** e o valor armazenado na variável **pc-vert** é desconsiderado. Após a substituição dos valores contidos nas variáveis **pix** e **pix-vert** entre si, se o valor da variável **pix** é menor do que 12, o programa armazena junto às coordenadas dos pontos de cruzamento armazenadas anteriormente a coordenada do ponto de cruzamento que estava armazenada na variável **pc-vert**. Independente do valor da variável **pix** ser maior ou menor do que 12, o programa varre a matriz-imagem do extremo inicial do caractere ao extremo final, desmarcando todos os "pixels-on" que foram marcados com o valor 9 e após isso, retorna ao "pixel-on" cuja coordenada estava armazenada na variável **pc-vert**, com o objetivo de continuar a varrer a matriz-imagem para baixo.

Se ((o valor da variável **vert** é diferente de 0) e ((a variável **pc-vert** não armazena a coordenada de nenhum ponto de cruzamento) ou (o valor da variável **pix** é menor ou igual a 12))), o programa verifica se o valor da variável **pix** é maior ou igual a 4 e menor ou igual a 12. Em caso afirmativo, se o valor da variável **pix-vert** é diferente de 0, porém é menor ou igual a 12, a variável **vert** é igualada a "V", indicando que provavelmente o caractere não é uma cerquilha. Caso contrário, se o valor da variável **pix-vert** é maior do que 12, a variável **vert** é igualada a "F", indicando que provavelmente o caractere é uma cerquilha.

Se o valor da variável **pix** não é maior ou igual a 4 e menor ou igual a 12, o programa verifica se o valor da variável **vert** é igual a "?". Em caso afirmativo, a variável **vert** é igualada a "D", indicando que o caractere não é uma cerquilha.

Caso contrário, se ((o valor da variável **vert** é igual a "right" (direita) ou "left" (esquerda)) e (os valores das variáveis **pix-vert** e **pix** são maiores do que 12)), a variável **vert** é igualada a "F".

- Se o valor da variável **vert** é diferente de "V", "D" ou "F", o programa continua o processo de verificação se o caractere é uma cerquilha ou não, realizando novamente todos os processos descritos acima, a partir do PASSO 1.

Ao terminar de varrer na matriz-imagem as linhas verticais, o programa continua o processo de verificação da existência da cerquilha varrendo as linhas horizontais da matriz-imagem, ou seja, as linhas horizontais dos caracteres contidos nessa matriz. O processo de varrer as linhas horizontais dos caracteres é realizado apenas para os caracteres que se encontram próximo ao início ou ao final da matriz-imagem, como descrito no item 5.3 deste capítulo. O processo de varrer as linhas horizontais dos caracteres é descrito a seguir:

Se ((o valor da variável **vert** é igual a "F") ou ((o valor da variável **vert** é igual a "V") e (o valor da variável **pix-vert** é maior do que 12)) e (a variável **pc-dir** armazena a coordenada de um ponto de cruzamento)) (este passo será referenciado adiante como PASSO 3), o programa varre a matriz-imagem do extremo inicial do caractere ao extremo final, desmarcando os "pixels-on" que foram marcados com o valor 9 ou 2 ao varrer as linhas verticais da cerquilha. Os "pixels-on" marcados com o valor 9 ou 2 são desmarcados, com o objetivo de que seja possível varrer esses "pixels-on" novamente. Em seguida, as variáveis **pix-horiz** e **horiz1** são igualadas a 0, a variável **horiz** é igualada a "right" (direita) e a partir do ponto de cruzamento cuja coordenada foi armazenada na variável **pc-dir** o programa varre a linha horizontal do caractere (este passo será referenciado adiante como PASSO 4). A coordenada do ponto de cruzamento

armazenada na variável **pc-dir** é também armazenada na variável **pc-horiz**. A variável **pix-horiz** armazena a quantidade de "pixels-on" varridos na linha horizontal do caractere, a variável **horiz1** indica se o caractere foi ou não reconhecido pelo programa como uma cerquilha e a variável **pc-horiz** armazena a coordenada do último ponto de cruzamento encontrado.

A linha horizontal do caractere é varrida utilizando-se a função horizontal, explicada no capítulo 4 deste trabalho. A direção que esse caractere deve ser varrido é determinada pela variável **horiz** e a variável **ccruz** é igualada a "right", indicando que o programa verificará se é possível varrer o caractere inicialmente para a direita. Se o valor da variável **horiz** é "right", o caractere é varrido para a direita (direção 0, 1 ou 7). Se o valor da variável **horiz** é "left", o caractere é varrido para a esquerda (direção 4, 3 ou 5). O processo de varrer a linha horizontal do caractere será referenciado adiante como PROCESSO 1. Para cada "pixel-on" varrido pela função horizontal, os seguintes procedimentos são realizados:

- Se ((o valor da variável **last-dir** é igual a 0, 2, 4 ou 6) e (o valor da variável **dir** é igual a 1, 3, 5 ou 7)), o valor **dir** é substituído pelo valor armazenado na variável **last-dir** e se o valor da variável **last-dir** é igual a 2 ou 6, o valor da variável **"condição-horizontal"** é incrementado de uma unidade.

- Se os valores das variáveis **"condição-horizontal"** e **"condição-de-folga"** são iguais a 0, a variável **pix** é incrementada de uma unidade e o "pixel-on" que está sendo varrido é marcado com o valor 5, com o objetivo de que o programa determine as linhas horizontais que foram varridas e que provavelmente constituem a cerquilha.

- Enquanto a linha horizontal do caractere é varrida para a direita ou para a

esquerda conforme os "pixels-on" estejam dispostos, o programa verifica se é encontrado ponto de cruzamento. Em caso afirmativo, com o objetivo de poder varrer novamente o "pixel-on" que constitui o ponto de cruzamento encontrado, o programa repõe este "pixel-on" na matriz-imagem e se o valor da variável **horiz** é igual a "left" (esquerda), o programa verifica se o ponto de cruzamento possui "pixel-on" vizinho que se encontra na direção 4, 5 ou 3, nesta ordem. Em caso afirmativo, o programa continua a varrer a linha horizontal do caractere a partir do "pixel-on" vizinho encontrado.

- O PROCESSO 1 é repetido, até que não existam mais "pixels-on" que possam ser varridos para a direita ou para a esquerda.

Se (((o valor da variável **horiz** é igual a "right" (direita)) ou (o valor da variável **ccruz** é diferente de 0)) e (o valor da variável **pix** é menor ou igual a 4)) (este passo será referenciado adiante como PASSO 5), o programa verifica se o valor da variável **horiz** é diferente de "right". Em caso afirmativo, a variável **pix** é igualada a 0 e o programa verifica se o ponto de cruzamento cuja coordenada foi armazenada na variável **pc-horiz** possui "pixel-on" vizinho que se encontra na direção 3, 4, 5 ou 6. Se isto ocorrer, a variável **ccruz** é igualada a "V", indicando que após varrer o caractere para a direita ou para a esquerda novamente, o PASSO 5 deve ser repetido. Caso contrário, a variável **ccruz** é igualada a 0, indicando que após varrer o caractere para a direita ou para a esquerda novamente, o PASSO 5 não deve ser repetido. Se o valor da variável **horiz** é igual a "right", o programa varre a área compreendida entre a coordenada do ponto de cruzamento armazenada na variável **pc-horiz** decrementada de 3 unidades e a coordenada desse ponto de cruzamento incrementada de 3 unidades, desmarcando os "pixels-on" marcados com valor 2 ao varrer as linhas verticais da cerquilha. Independente do valor da variável **horiz** ser diferente ou igual a

"right", esta variável é igualada a "left" e o programa retorna ao ponto de cruzamento cuja coordenada foi armazenada na variável **pc-horiz**.

Se (((o valor da variável **horiz** é diferente de "right") e (o valor da variável **ccruz** é igual a 0)) ou (o valor da variável **pix** é maior do que 4)), o programa verifica se esta condição foi satisfeita, mas o valor da variável **pix** é menor ou igual a 4. Em caso afirmativo, a variável **horiz1** é igualada a "H", indicando que o caractere não é uma cerquilha. Caso contrário, se o valor da variável **pix** é maior do que 4 e o valor da variável **pix-horiz** é igual a 0, o valor contido na variável **pix** é armazenado na variável **pix-horiz**, a variável **pix** é igualada a 0, a variável **horiz** é igualada a "right" e o programa realiza os procedimentos descritos a seguir:

- O programa varre a área compreendida entre a coordenada formada pela ordenada armazenada na variável **pc-dir** e a abscissa armazenada nessa variável decrementada de 4 unidades e a coordenada formada pela linha final da matriz-imagem e a abscissa armazenada na variável **pc-dir** incrementada de 4 unidades, desmarcando os "pixels-on" marcados com o valor 5 ao varrer a matriz-imagem para a direita ou para a esquerda. O programa armazena a coordenada armazenada na variável **pc-dir** junto aos pontos de cruzamento cujas coordenadas foram armazenadas ao varrer a matriz-imagem para a direita ou para a esquerda.

- O programa varre o caractere para baixo a partir da coordenada armazenada na variável **pc-dir**, até que não existam mais "pixels-on" que possam ser varridos nessa direção ou seja encontrado um ponto de cruzamento. Para cada "pixel-on" que está sendo varrido para baixo, o programa executa os procedimentos descritos a seguir:

- Se o "pixel-on" que está sendo varrido é um ponto de cruzamento, o programa verifica se a ordenada deste ponto de cruzamento encontra-se a uma distância menor ou igual 4 "pixels-on" da ordenada armazenada na variável **pc-dir**. Em caso afirmativo, o ponto de cruzamento é desconsiderado e o programa continua a varrer o caractere para baixo, a partir do "pixel-on" que constitui o ponto de cruzamento desconsiderado. Se a ordenada do ponto de cruzamento encontra-se a uma distância maior do que 4 "pixels-on" da ordenada armazenada na variável **pc-dir**, a coordenada do ponto de cruzamento é armazenada na variável **pc-horiz**.

- Quando não existem mais "pixels-on" que possam ser varridos para baixo, o programa verifica se ao varrer o caractere nessa direção foi encontrado algum ponto de cruzamento. Em caso afirmativo, o programa verifica se o ponto de cruzamento possui "pixel-on" vizinho. Se isto ocorrer, o programa continua a varrer o caractere para baixo, a partir do "pixel-on" que constitui o ponto de cruzamento. Caso contrário, o programa finaliza o processo de varrer o caractere para baixo. Se não foi encontrado nenhum ponto de cruzamento ao varrer o caractere para baixo, a variável **horiz1** é igualada a "H", indicando que o caractere que está sendo varrido não é uma cerquilha e o processo de varrer o caractere para baixo é finalizado.

Se o valor da variável **pix** é maior do que 4 e o valor da variável **pix-horiz** é diferente de 0, a variável **horiz1** é igualada a "S", indicando que o caractere que está sendo varrido é uma cerquilha.

Se o valor da variável **horiz1** é diferente de "H" ou "S", o programa volta a varrer a matriz-imagem executando os procedimentos subseqüentes ao

PROCESSO 1 e inclusive este processo. Caso contrário, se o valor da variável **horiz1** é igual a "S", o caractere varrido é uma cerquilha e portanto, o seu esqueleto é apagado da matriz-imagem. Se o valor da variável **horiz1** é diferente de "S", o caractere varrido não representa uma cerquilha e portanto, deve ser mantido na matriz-imagem.

5.7 - ELIMINAÇÃO DO 'R' E 'CIFRÃO' MANUSCRITOS E DO 'R' E 'CIFRÃO' IMPRESSOS NA ÁREA RESERVADA À ESCRITA DA QUANTIA NUMÉRICA DE CHEQUES BANCÁRIOS

5.7.1 - Eliminação do 'R' e 'Cifrão' Manuscritos

Algumas pessoas ao preencherem a quantia numérica de um cheque, podem delimitar esta quantia do lado esquerdo pelos caracteres 'R' e 'cifrão' (R$). Para que estes caracteres não interfiram no processo de reconhecimento dos dígitos, optou-se por eliminá-los. A figura 5.22 ilustra a área reservada à escrita da quantia numérica de um cheque bancário, preenchida e delimitada pelo 'R$' manuscrito.

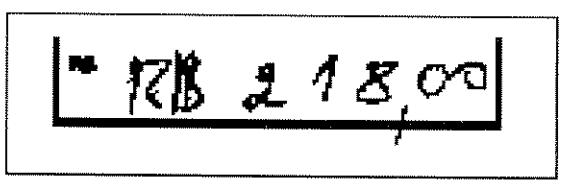


Figura 5.22 - Área reservada à escrita da quantia numérica de um cheque bancário, preenchida e delimitada pelo 'R$' manuscrito.

O programa realiza a eliminação do esqueleto do 'R$' manuscrito da matriz-imagem, executando a função verifica-real descrita a seguir. Esta função realiza apenas a verificação e a eliminação do caractere 'R' presente no campo da quantia numérica do cheque bancário. O programa considera que o 'R' sempre

é acompanhado do caractere '$'. Devido a isto, ao encontrar o caractere 'R', o programa considera que o próximo caractere a ser encontrado é o '$' e portanto, este caractere é apagado da matriz-imagem. A matriz-imagem armazena os "pixels-on" e os "pixels-off" do esqueleto da imagem da área reservada à escrita da quantia numérica de um cheque bancário.

Para facilitar o entendimento da função verifica-real, optou-se por explicá-la no texto e esboçar o seu fluxograma. Aconselha-se ao leitor que para sua melhor clareza, leia o texto que descreve a função verifica-real e acompanhe ao mesmo tempo o seu fluxograma. A figura 5.23 que se encontra nas páginas 760 771 do 2º volume deste trabalho, ilustra o fluxograma da função verifica-real. Nesta figura, alguns comentários que se encontram no texto não constam do fluxograma. Devido à extensão do fluxograma ele foi dividido em 11 partes.

Verifica-Real

Para verificar se o 'R' e o '$' foram escritos no campo da quantia numérica de um cheque bancário, o programa varre o lado direito e o lado esquerdo do caractere cuja coluna final está localizada a esquerda da 50ª coluna da matriz-imagem. O programa considera que o 'R' do 'R$' manuscrito é constituído de duas partes. A parte que se encontra do lado direito do caractere é denominada de linha vertical direita e é constituída de "pixels-on" que se encontram na direção vertical ou na direção diagonal. A parte que se encontra do lado esquerdo do caractere é denominada de linha vertical esquerda e é constituída em sua maior parte de "pixels-on" que se encontram na direção vertical.

Com o objetivo de varrer a linha vertical direita do caractere, o programa inicializa a variável **lr** com o valor "right" (direita) e para varrer a linha vertical esquerda do caractere, o programa inicializa a variável **lr** com o valor "left" (esquerda). Ao inicializar a função verifica-real, o programa iguala a variável **lr** a

"right". A variável **lr** determina qual a linha vertical do 'R' do 'R$' manuscrito será varrida pelo programa.

Se o valor da variável **lr** é "right" (esta condição será referenciada adiante como CONDIÇÃO 1), o programa varre a matriz-imagem, isto é, varre a região da matriz-imagem em que se encontra o caractere, da esquerda para a direita no intervalo compreendido entre a ordenada do "pixel-on" que se encontra no extremo superior do caractere e a ordenada deste caractere, que corresponde a 2/3 de sua altura, procurando entre os "pixels-on" desse intervalo aquele que se encontra mais a direita. Ao encontrar nesse intervalo o "pixel-on" que se encontra mais a direita, o programa o denomina de *"pixel-inicial"*.

A partir do *"pixel-inicial"*, o programa varre o caractere para cima, marcando os "pixels-on" varridos com o valor 5. O caractere é varrido para cima, até que não existam mais "pixels-on" que possam ser varridos nessa direção. O último "pixel-on" varrido para cima é denominado no programa de *"pixel-final"*.

Se o valor da variável **lr** é diferente de "right", o programa retorna ao *"pixel-final"* e varre o caractere para a esquerda. O caractere é varrido para a esquerda, até que não existam mais "pixels-on" que possam ser varridos nessa direção ou tenham sido varridos mais do que 5 "pixels-on" para a esquerda e seja encontrado um "pixel-on" que se encontra na direção 5 ou 6. Enquanto o caractere é varrido para a esquerda, o programa armazena a quantidade de "pixels-on" varridos nesse processo e marca estes "pixels-on" com o valor 5. O último "pixel-on" varrido para a esquerda é considerado como sendo o *"pixel-final"*.

O programa varre a região compreendida entre a coordenada formada pela coluna inicial do caractere e 2/3 de sua altura e a coordenada formada pela coluna final do caractere e seu extremo superior, desmarcando os "pixels-on" marcados com o valor 5 ao varrer o caractere para cima e quando o caractere é varrido para a esquerda.

A partir do *"pixel-final"*, o programa varre o caractere para baixo, até que não existam mais "pixels-on" que possam ser varridos nessa direção. Para cada "pixel-on" que está sendo varrido para baixo, o programa realiza os procedimentos descritos a seguir:

- Se o valor da variável **"condição-de-folga"** é igual a 0, a variável **pix0** é incrementada de uma unidade e se ((o valor da variável **lr** é igual a "right" e o "pixel-on" que acabou de ser varrido para baixo encontra-se na direção 5 ou 6) ou (o valor da variável **lr** é igual a "left" e o "pixel-on" que acabou de ser varrido para baixo encontra-se na direção 6)), a variável **pix1** é incrementada de uma unidade. A variável **pix0** armazena a quantidade de "pixels-on" varridos para baixo e a variável **pix1** armazena a quantidade de "pixels-on" varridos que se encontram na direção 5 ou 6 ou apenas na direção 6.

- Enquanto o caractere é varrido para baixo, se ((é encontrado ponto de cruzamento) ou (o valor da variável **pix0** é igual a 1, indicando que o "pixel-on" que está sendo varrido é o primeiro "pixel-on" a ser varrido para baixo)), o programa verifica se (((o "pixel-on" que está sendo varrido possui "pixel-on" vizinho que se encontra na direção 0, 7 ou 6, nesta ordem) e (o valor da variável **lr** é igual a "right")) ou ((o "pixel-on" que está sendo varrido possui "pixel-on" vizinho que se encontra na direção 4, 5 ou 6, nesta ordem) e (o valor da variável **lr** é igual a "left"))). Em caso afirmativo, o programa continua a varrer o caractere para baixo, a partir do "pixel-on" vizinho que se encontra em uma dessas direções. Antes do programa continuar a varrer o caractere para baixo, se ((o "pixel-on" que está sendo varrido é um ponto de cruzamento) e (a variável **pc-med** não armazena a coordenada de nenhum ponto de cruzamento)), a coordenada do ponto de cruzamento encontrado é armazenada na variável **pc-med**, os valores contidos nas variáveis **pix0** e **pix1**

são armazenados nas variáveis **pix2** e **pix3**, respectivamente, e as variáveis **pix0** e **pix1** são igualadas a 0. A variável **pix2** armazena a quantidade de "pixels-on" varridos para baixo antes de ser encontrado um ponto de cruzamento, a variável **pix3** armazena a quantidade de "pixels-on" varridos que se encontram na direção 5 ou 6, antes de ser encontrado um ponto de cruzamento e a variável **pc-med** armazena a coordenada do ponto de cruzamento encontrado.

- Quando não existem mais "pixels-on" que possam ser varridos para baixo, o programa verifica se a variável **pc-med** armazena a coordenada de algum ponto de cruzamento. Em caso afirmativo, a coordenada do último "pixel-on" varrido para baixo é armazenada na variável **end-vert**, os valores contidos nas variáveis **pix0** e **pix1** são armazenados nas variáveis **pix4** e **pix5**, respectivamente, o programa continua a varrer o caractere para baixo a partir da coordenada armazenada na variável **pc-med**, a coordenada armazenada nesta variável é desconsiderada e as variáveis **pix0** e **pix1** são igualadas a 0. Caso contrário, se a variável **pc-med** não armazena a coordenada de nenhum ponto de cruzamento, o programa finaliza o processo de varrer o caractere para baixo. Se existem "pixels-on" que possam ser varridos para baixo, o programa continua a varrer o caractere nessa direção. A variável **end-vert** armazena a coordenada final de uma das linhas verticais do caractere, a variável **pix4** armazena a quantidade de "pixels-on" varridos para baixo, após ter sido encontrado o ponto de cruzamento cuja coordenada foi armazenada na variável **pc-med** e a variável **pix5** armazena a quantidade de "pixels-on" varridos que se encontram na direção 5 ou 6, ou apenas na direção 6, após ter sido encontrado o ponto de cruzamento cuja coordenada foi armazenada na variável **pc-med**.

  Se o valor da variável **pix0** é maior do que o valor da variável **pix4**, são

adicionados aos valores contidos nas variáveis **pix2** e **pix1**, os valores das variáveis **pix4** e **pix3**, respectivamente.

Se o valor da variável **pix0** é menor ou igual ao valor da variável **pix4**, o programa adiciona o valor contido na variável **pix2** ao valor contido na variável **pix4** e armazena o resultado dessa soma na variável **pix0**, adiciona o valor contido na variável **pix3** ao valor contido na variável **pix5** e armazena o resultado dessa soma na variável **pix1**. O "pixel-on" cuja coordenada foi armazenada na variável **end-vert** é considerado como sendo o último "pixel-on" varrido para baixo.

Se o valor da variável **lr** é igual a "right", a coordenada do último "pixel-on" varrido para baixo é armazenada na variável **hi-dir** e se ((o valor da variável **pix0** é maior ou igual a 10) e (o valor da variável **pix1** é maior ou igual a 60% do valor contido na variável **pix0**)), a variável **lr** é igualada a "left". Caso contrário, a variável **lr** é igualada a "N", indicando que o caractere que foi varrido para baixo não é o 'R' do 'R$' manuscrito. A variável **hi-dir** armazena a coordenada inferior da linha vertical direita do esqueleto da imagem do 'R' do 'R$' manuscrito.

Se o valor da variável **lr** é diferente de "right", a coordenada do último "pixel-on" varrido para baixo é armazenada na variável **hi-esq** e se ((o valor da variável **pix0** é maior ou igual a 10) e (o valor da variável **pix1** é maior ou igual a 60% do valor contido na variável **pix0**) e (o valor absoluto da diferença entre a abscissa da coordenada armazenada na variável **hi-dir** e a abscissa da coordenada armazenada na variável **hi-esq** é maior ou igual a 2)), a variável **lr** é igualada a "R", indicando que o caractere que foi varrido para baixo é o 'R' do 'R$' manuscrito. Caso contrário, a variável **lr** é igualada a "N". A variável **hi-esq** armazena a coordenada inferior da linha vertical esquerda do esqueleto da imagem do 'R' do 'R$' manuscrito.

Se o valor da variável **lr** é diferente de "N" ou "R", o programa retorna à

CONDIÇÃO 1 e executa os processos subseqüentes a esta condição. Caso contrário, o programa finaliza o processo de varrer o caractere para baixo.

Se o valor da variável **Ir** é igual a "R", o caractere varrido é o 'R' do 'R$' manuscrito, portanto, este caractere é apagado da matriz-imagem e considerando que o 'R' é sempre acompanhado do '$', o programa apaga da matriz-imagem o caractere subseqüente ao 'R' apagado. Se o valor da variável **Ir** é diferente de "R", o caractere varrido não representa o 'R' do 'R$' manuscrito e portanto, deve ser mantido na matriz-imagem.

5.7.2 - Eliminação do 'R$' Impresso na Área Reservada à Escrita da Quantia Numérica de Cheques Bancários

Na área reservada à escrita da quantia numérica do cheque é impresso o 'R' e '$'. Se o campo que envolve a quantia numérica é composto de 2 tarjas verticais e uma tarja horizontal ou apenas de 2 tarjas verticais, o 'R$' impresso localiza-se no extremo superior esquerdo desse campo, como está ilustrado nas figuras 5.15 (b) e 5.15 (c). Para que o esqueleto da imagem do 'R$' impresso não interfira no processo de reconhecimento dos caracteres numéricos, optou-se por eliminá-lo da matriz-imagem. A matriz-imagem armazena os "pixels-on" e os "pixels-off" do esqueleto da imagem da área reservada à escrita da quantia numérica de um cheque bancário.

Se o campo que envolve a quantia numérica é constituído de 2 tarjas horizontais e 2 tarjas verticais, o 'R$' é impresso na parte central da tarja horizontal superior, como está ilustrado na figura 5.15 (a). Neste caso a imagem do esqueleto do 'R$' impresso é eliminada da matriz-imagem, quando a tarja horizontal superior for eliminada desta matriz.

Devido a resolução utilizada no "scanner", o 'R$' impresso apresenta pouca definição, aparecendo na matriz-imagem como 2 caracteres conectados.

Quando o programa realiza o afilamento dos caracteres da matriz-imagem, o esqueleto do 'R$' impresso possui número de "pixels-on" maior ou igual a 5 e menor ou igual a 20. Por este motivo, quando o programa varre o esqueleto da imagem de um caractere cujo número total de "pixels-on" da matriz-final é maior ou igual a 5 e menor ou igual a 20 e que está localizado no início da matriz-imagem, ou seja, o valor da abscissa do extremo direito do caractere é menor ou igual a 15% do número total de colunas da matriz-imagem e o valor da ordenada do extremo superior é menor ou igual a 40% do número total de linhas da matriz-imagem, o programa considera que esse caractere é o 'R$' impresso e portanto, o apaga da matriz-imagem. A matriz-final armazena os "pixels-on" e os "pixels-off" que constituem o esqueleto da imagem de um caractere isolado.

5.8 - JANELAS DE ENTRADA E SAÍDA DO PROGRAMA

As janelas de entrada e de saída são utilizadas pelo programa para solicitar ao usuário as informações necessárias para que o programa possa iniciar o processo de reconhecimento e informar os resultados obtidos.

Na janela de entrada, o programa solicita ao usuário o nome do arquivo em que foi gravado a área reservada à escrita da quantia numérica que deve ser reconhecida. Na janela de saída, o programa mostra ao usuário a imagem da área reservada à escrita da quantia numérica, a quantia numérica reconhecida pelo programa, o extenso dessa quantia e informa ao usuário qualquer erro que possa ter ocorrido durante a fase de reconhecimento.

O algoritmo de reconhecimento foi desenvolvido e testado em um microcomputador Pentium - 100 MHz com 16 Mbytes de memória RAM.

A utilização do programa de reconhecimento é muito simples, sendo que este programa pode ser executado em qualquer microcomputador que possua o sistema operacional DOS.

Para que o usuário execute o programa de reconhecimento, é necessário apenas que a partir do diretório que contém este programa, o usuário tecle a palavra **"cheque"**.

A explicação à respeito das janelas de entrada e saída do programa é descrita adiante e na ordem em que essas janelas são apresentadas pelo programa ao usuário.

5.8.1 - Janela de Apresentação do Programa

Inicialmente o programa apresenta ao usuário a janela ilustrada na figura 5.24. Esta janela não é considerada como uma janela de entrada ou uma janela de saída, pois o seu objetivo é de apenas informar ao usuário a que se destina o software.

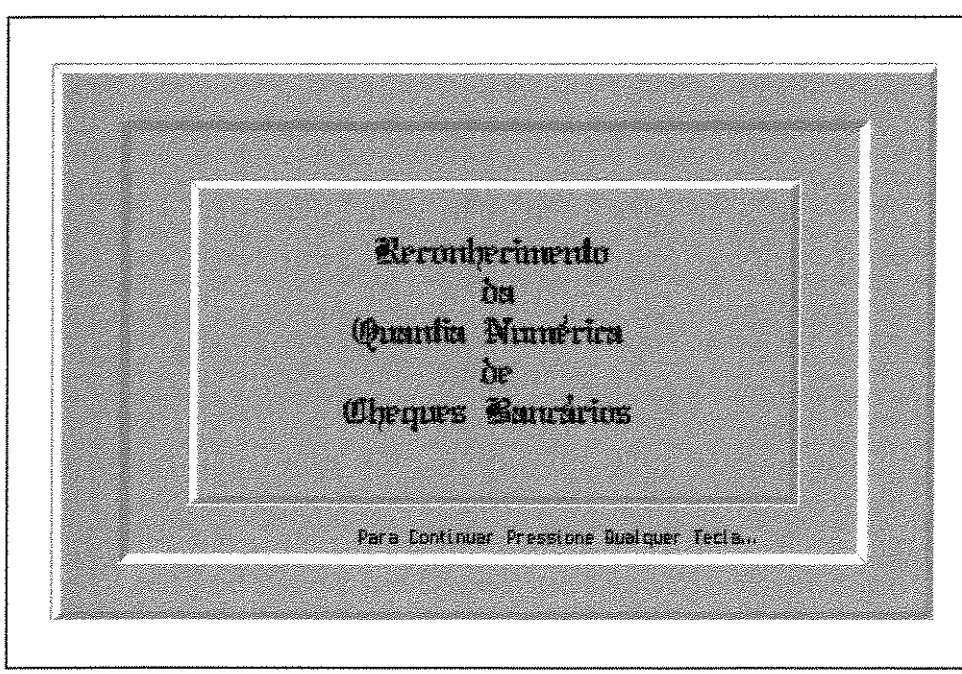


Figura 5.24 - Janela de apresentação.

5.8.2 - Janela de Entrada do Programa

A figura 5.25 ilustra a janela que solicita ao usuário o nome do arquivo onde está gravada a imagem da área reservada à escrita da quantia numérica a ser reconhecida. Nessa figura também é mostrada a resposta fornecida pelo usuário.

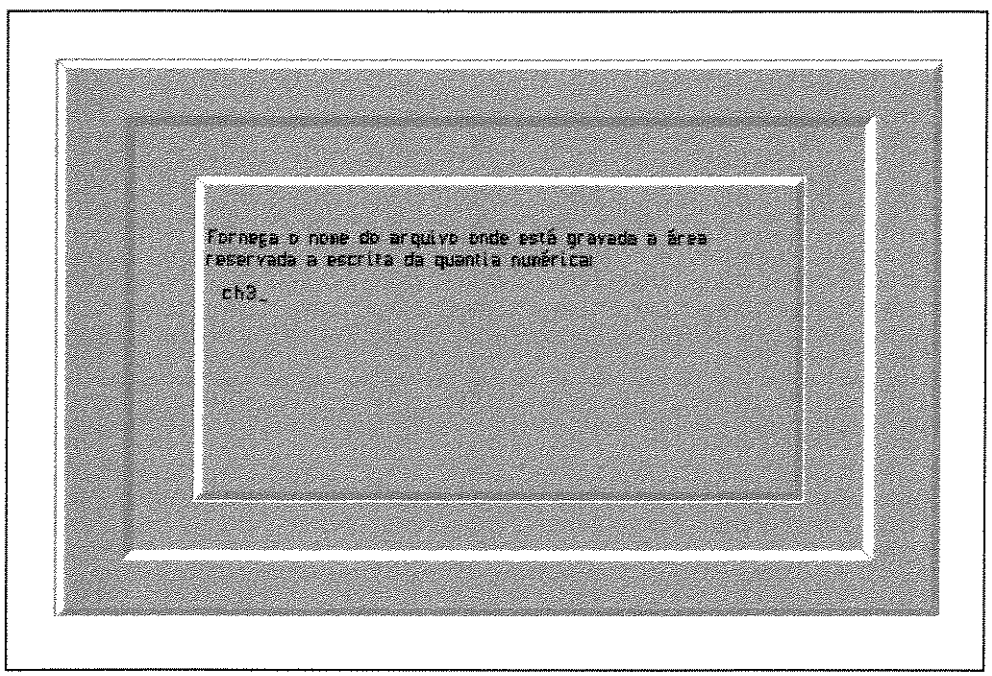


Figura 5.25 - Janela que solicita ao usuário o nome do arquivo onde está gravada a imagem da área reservada à escrita da quantia numérica a ser reconhecida.

5.8.3 - Janela de Saída do Programa

Se na janela ilustrada na figura 5.25, o usuário digitar o nome de um arquivo que não é encontrado pelo programa, é apresentada ao usuário a janela de saída mostrada na figura 5.26.

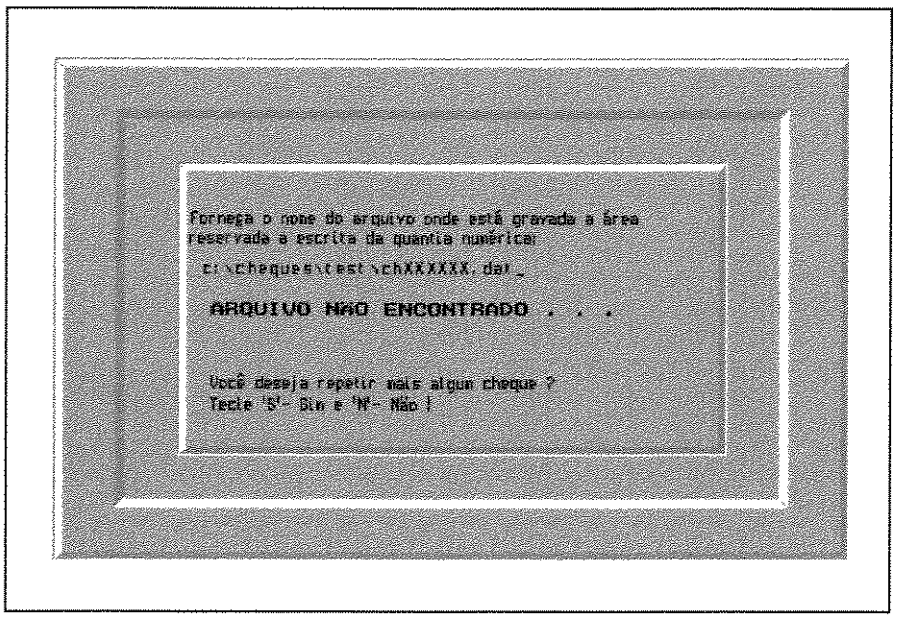


Figura 5.26 - Janela que informa ao usuário que o arquivo onde está gravada a imagem da área reservada à escrita da quantia numérica a ser reconhecida não foi encontrado.

Após o usuário responder o nome do arquivo onde está gravada a imagem da área reservada à escrita da quantia numérica a ser reconhecida e teclar "enter", se este arquivo for encontrado, o programa realiza o reconhecimento da quantia numérica gravada nesse arquivo.

Para reconhecer a quantia numérica, o programa realiza a leitura do arquivo onde está gravada a área reservada à escrita dessa quantia, a obtenção do esqueleto dessa área, a segmentação e reconhecimento de cada caractere que está contido nessa área. Estes procedimentos foram explicados nos capítulos 3, 4 e 5 deste trabalho.

Enquanto o programa está executando o reconhecimento da quantia numérica, é mostrada ao usuário a janela ilustrada na figura 5.27.

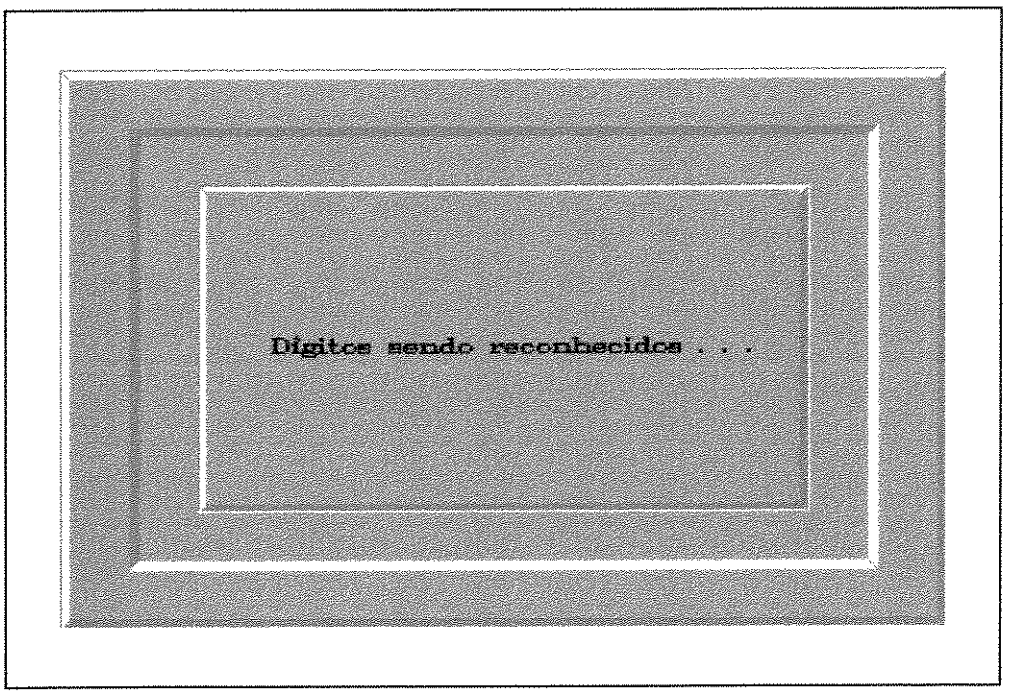


Figura 5.27 - Janela apresentada ao usuário informando que os caracteres estão sendo reconhecidos pelo programa.

Enquanto o programa realiza o processo de reconhecimento, se a quantidade de memória disponível no sistema for insuficiente, é apresentada a janela mostrada na figura 5.28 e após o usuário teclar "enter" o programa finaliza o processo de reconhecimento.

Para que uma nova quantia numérica seja reconhecida, o usuário deve executar novamente o programa.

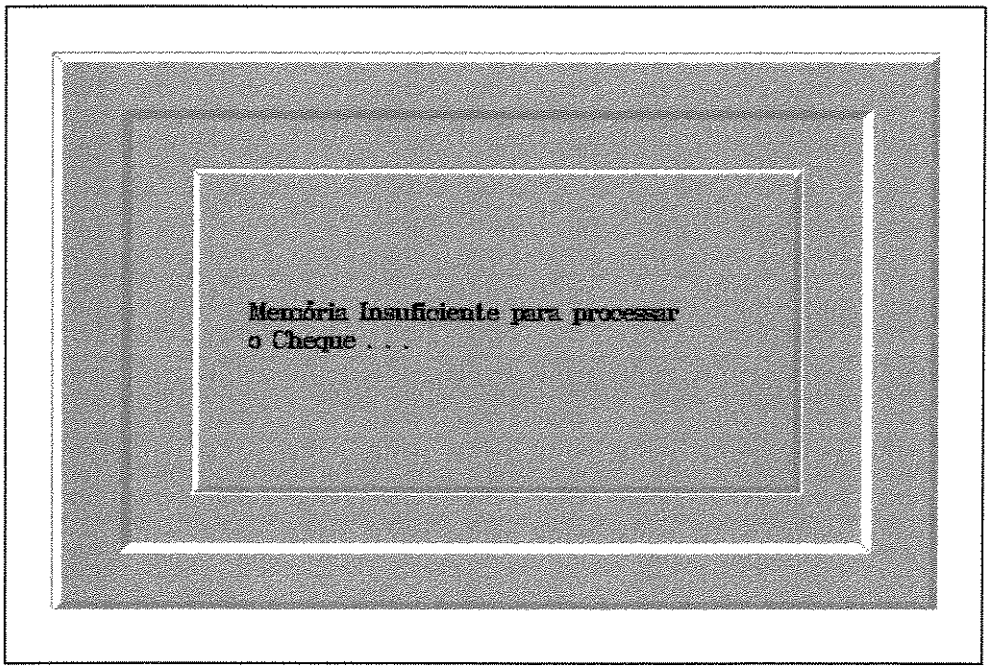


Figura 5.28 - Janela apresentada ao usuário informando que o sistema não possui memória suficiente.

No processo de reconhecimento o programa armazena as coordenadas dos pontos de cruzamento do caractere que está sendo reconhecido. A coordenada desses pontos de cruzamento é armazenada em uma pilha. Para cada caractere que está sendo reconhecido, o programa delimita uma pilha com capacidade de armazenar a coordenada de até 20 pontos de cruzamento, sendo que se o caractere que está sendo reconhecido possuir mais de 20 pontos de cruzamento, é apresentada a janela mostrada na figura 5.29 e após o usuário teclar "enter" o programa finaliza o processo de reconhecimento.

Para que uma nova quantia numérica seja reconhecida, o usuário deve executar novamente o programa.

O termo pilha é utilizado para designar uma estrutura de armazenamento e recuperação de dados. FIFO, LIFO e FILO são algumas das formas existentes para o armazenamento e recuperação de dados. Na FIFO o primeiro dado a ser armazenado é o primeiro dado a ser recuperado, na LIFO o último dado a ser armazenado é o primeiro a ser recuperado e na FILO o primeiro dado a ser

armazenado é o último a ser recuperado.

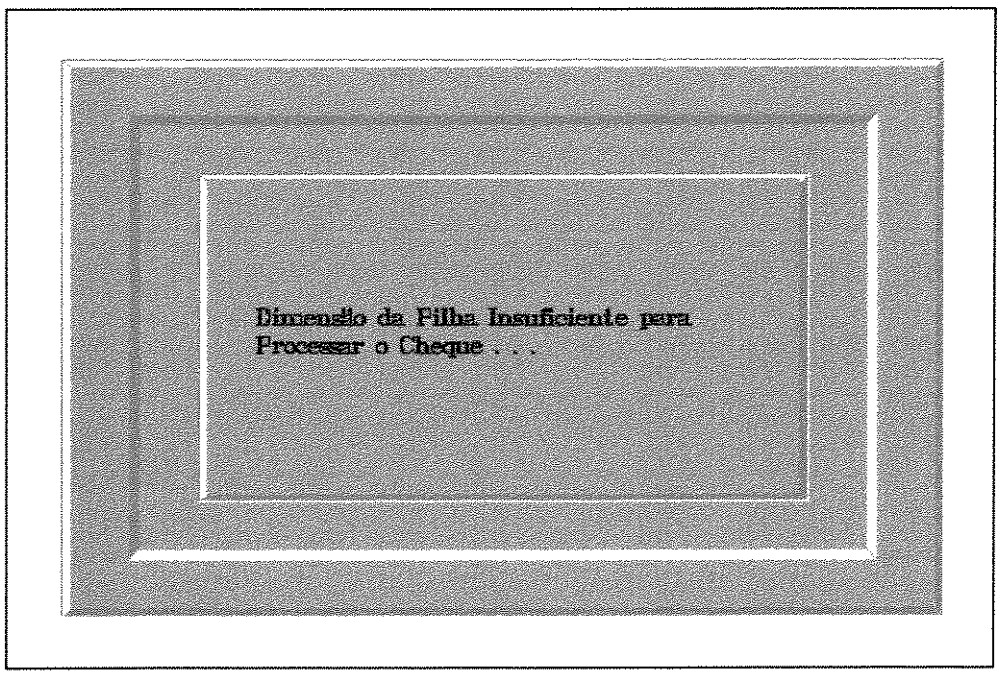


Figura 5.29 - Janela apresentada ao usuário informando que a dimensão da pilha é insuficiente.

Após terminar de reconhecer os caracteres contidos na área reservada à escrita da quantia numérica, o programa apresenta ao usuário a janela ilustrada na figura 5.30. Nesta janela o programa apresenta o nome do arquivo cuja quantia numérica foi reconhecida, a imagem contida nesse arquivo, a quantia numérica reconhecida pelo programa, o extenso dessa quantia e o programa solicita ao usuário que o informe se ele deseja ou não reconhecer a quantia numérica de mais algum cheque. Se o usuário deseja reconhecer a quantia numérica de mais algum cheque deve responder "S" e se ele responder "N" o programa é finalizado, pois o usuário não deseja que seja reconhecido a quantia numérica de mais nenhum cheque.

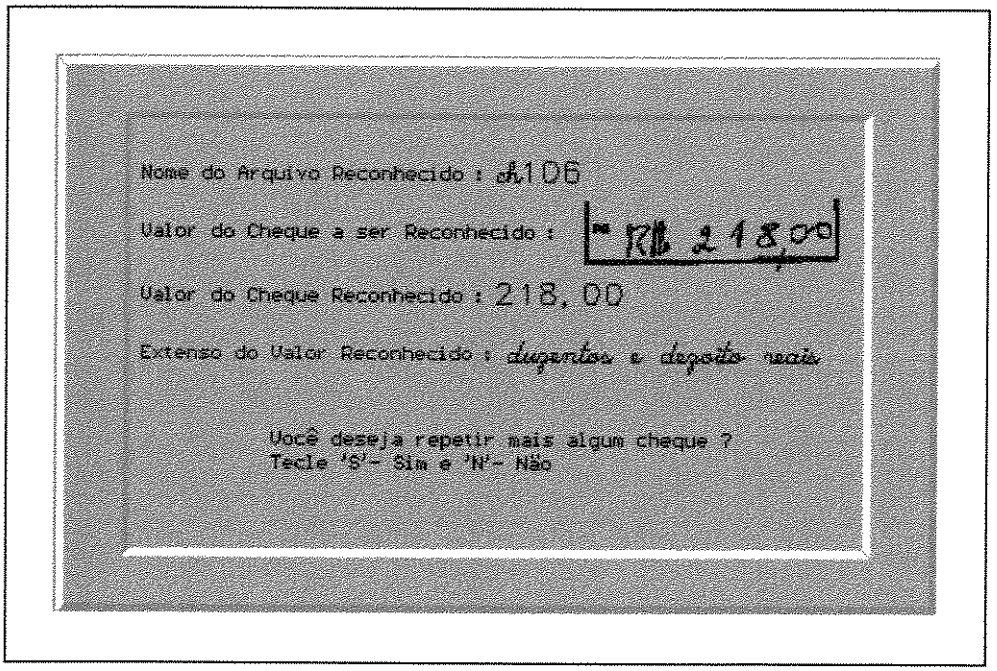


Figura 5.30 - Janela que apresenta os resultados obtidos no processo de reconhecimento.

Se durante o reconhecimento dos caracteres, o programa não conseguir reconhecer algum dos caracteres que estão presentes na área reservada à escrita da quantia numérica. O programa coloca na janela ilustrada na figura 5.30 e na posição do caractere que não foi reconhecido o sinal "?". O significado desse sinal é apresentado na parte inferior dessa janela.

5.9 - RESULTADOS OBTIDOS

O algoritmo de reconhecimento de caracteres manuscritos desenvolvido neste trabalho foi testado com 1212 números. Estes 1212 números foram obtidos de 300 cheques escritos por 75 pessoas. É importante ressaltar que nos 1212 dígitos citados não estão incluídos os pontos, as vírgulas, as cerquilhas, os 'R$' impressos e 'R$' manuscritos que podem aparecer nos cheques bancários. Foi solicitado a cada uma das pessoas que escrevessem em 4 áreas reservadas à escrita da quantia numérica valores numéricos aleatórios. A área reservada à escrita da quantia numérica foi simulada em papel sulfite branco.

Cada dígito escrito na área reservada à escrita da quantia numérica é afilado e segmentado pelos algoritmos explicados no capítulo 3 e 5 e o seu esqueleto é reconhecido pelo algoritmo explicado no capítulo 4. Várias pessoas ao escreverem os valores numéricos na área reservada utilizaram caneta azul, outras utilizaram caneta preta, ou mesmo caneta vermelha, etc. Independente da cor da caneta utilizada, todos as áreas reservadas à escrita da quantia numérica foram varridas por um "scanner" monocromático.

Dos 1212 números escritos, 1211 destes números foram reconhecidos de forma correta e apenas um número isolado foi reconhecido de maneira incorreta. O programa é muito eficiente para reconhecer caracteres e reconheceu de maneira correta todos os caracteres numéricos sobrepostos e conectados, todos os pontos, as vírgulas, as cerquilhas, os 'R$' impressos, os 'R$' manuscritos e o campo que envolve a quantia numérica de um cheque bancário. Alguns dos 1212 números testados são mostrados nas figuras 5.5, 5.14 e 5.20.

Os resultados mostraram uma taxa de acerto de 99,92% e uma taxa de erro de 0,08%.

As tabelas 5.1, 5.2, 5.3 e 5.4 mostram o tempo de execução do programa desenvolvido neste trabalho para o reconhecimento de alguns dos valores numéricos escritos na área reservada, quando é utilizado um microcomputador Pentium - 100 MHz com 16 Mbytes de memória RAM.

Tabela 5.1 - Tempo de execução do programa para o reconhecimento da quantia numérica de cheques bancários, quando são encontrados apenas caracteres numéricos isolados.

QUANTIDADE DE DÍGITOS PRESENTES NA QUANTIA NUMÉRICA	TEMPO DE EXECUÇÃO (seg)
3	0.120
4	0.234
5	0.171
6	0.229
7	0.356

Tabela 5.2 - Tempo de execução do programa para o reconhecimento da quantia numérica de cheques bancários, quando são encontrados caracteres numéricos isolados, sobrepostos e conectados.

QUANTIDADE DE DÍGITOS PRESENTES NA QUANTIA NUMÉRICA	DÍGITOS PRESENTES NA QUANTIA NUMÉRICA	TEMPO DE EXECUÇÃO (seg)
3	isolado e conectados	0.100
4	isolados e sobrepostos	0.234
5	isolados e conectados	0.177
4	isolados e conectados	0.228
5	sobrepostos e conectados	0.310
6	isolados e conectados	0.332
6	isolados, sobrepostos e conectados	0.411
8	isolados, sobrepostos e conectados	0.470

Tabela 5.3 - Tempo de execução do programa para o reconhecimento de caracteres numéricos escritos na área reservada à escrita da quantia numérica de cheques bancários, quando nesta área existem tarjas verticais e/ou tarjas horizontais.

TIPOS DE TARJAS PRESENTES NA ÁREA RESERVADA À ESCRITA DA QUANTIA NUMÉRICA	TIPO DE CARACTERE DELIMITADOR DA QUANTIA NUMÉRICA	QUANTIDADE DE CARACTERES PRESENTES NA QUANTIA NUMÉRICA	TEMPO DE EXECUÇÃO (seg)
TVE e THI	'R$' impresso	5	0.583
TVD e THI	'R$' impresso	5	0.526
THI	'R$' impresso	5	0.583
THS, THI, TVE e TVD	'R$' impresso	5	0.625
THS, THI e TVE	'R$' impresso	5	0.643
THS, THI, TVD	'R$' impresso	5	0.588
THS e THI	'R$' impresso	5	0.567
TVE e TVD	'R$' impresso	5	0.454
TVE	'R$' impresso	5	0.438
TVD	'R$' impresso	5	0.417
THS, TVE e TVD	'R$' impresso	5	0.491
THS e TVE	'R$' impresso	5	0.415
THS e TVD	'R$' impresso	5	0.352
THS	'R$' impresso	5	0.612
THI, TVE e TVD	'R$' impresso e uma cerquilha do lado esquerdo da quantia numérica	5	0.676
THI, TVE e TVD	'R$' impresso e uma cerquilha do lado direito da quantia numérica	7	0.735

Legenda da tabela 5.3

THS - tarja horizontal superior THI - tarja horizontal inferior

TVE - tarja vertical esquerda TVD - tarja vertical direita.

Tabela 5.4 - Tempo de execução do programa para o reconhecimento de 5 caracteres numéricos isolados escritos na área reservada à escrita da quantia numérica de cheques bancários, quando nesta área existem as tarjas verticais, a tarja horizontal inferior, cerquilhas, 'R$' impresso e 'R$' manuscrito.

CARACTERES DELIMITADORES DA QUANTIA NUMÉRICA	DÍGITOS PRESENTES NA QUANTIA NUMÉRICA	TEMPO DE EXECUÇÃO (seg)
2 cerquilhas	isolados	1.103
1 'R$' manuscrito e 1 'R$' impresso	isolados	0.499
1 'R$' MANUSCRITO	isolados e conectados	0.337

5.10 - CONCLUSÕES

Neste capítulo o algoritmo de reconhecimento de caracteres numéricos manuscritos isolados descrito no capítulo 4 deste trabalho, é utilizado no reconhecimento da quantia numérica de cheques bancários. Para que isto fosse possível, foi necessário adicionar a esse algoritmo de reconhecimento, algoritmos que realizassem a segmentação de caracteres sobrepostos e caracteres conectados, eliminação de vírgulas, pontos, cerquilhas, 'R$' impressos, 'R$' manuscritos e do campo que envolve a quantia numérica.

O algoritmo de segmentação de caracteres sobrepostos realiza a segmentação destes caracteres varrendo o caractere em toda a sua extensão. Este processo é realizado varrendo a matriz-original a partir da coordenada CILHI e copiando os "pixels-on" e os "pixels-off" varridos para a nova-matriz. Ao final desse processo a nova-matriz contém um caractere isolado ou caracteres conectados.

O algoritmo de segmentação de caracteres conectados verifica se a nova-matriz contém um caractere isolado ou caracteres conectados, através da análise da quantidade de linhas horizontais superior e inferior que são encontradas na

nova-matriz e dos pontos de cruzamento encontrados no intervalo compreendido entre 80% e 140% da largura média dos caracteres contidos nessa matriz.

As principais características que o algoritmo de reconhecimento de caracteres manuscritos desenvolvido neste trabalho possui são:

a) o reconhecimento dos caracteres manuscritos independe da pessoa que os escreveu;

b) o reconhecimento dos caracteres independe do seu tamanho e da posição em que o caractere foi escrito;

c) desempenho satisfatório, mesmo na presença de ruído;

d) taxa de acerto elevada.

Ao analisar os resultados obtidos no item 5.9 pode-se verificar que o algoritmo desenvolvido neste trabalho para o reconhecimento de caracteres manuscritos apresenta uma excelente porcentagem de reconhecimento.

CAPÍTULO 6

CONCLUSÕES, CONTRIBUIÇÕES DO TRABALHO E SUGESTÕES PARA FUTUROS TRABALHOS

6.1 - CONCLUSÕES

Este trabalho descreveu o desenvolvimento de um software em linguagem C para o reconhecimento de caracteres numéricos manuscritos. Estes caracteres podem estar isolados, sobrepostos ou conectados e apresentar ao longo do seu contorno trechos contínuos ou fragmentados.

No desenvolvimento deste trabalho, o reconhecimento dos caracteres foi realizado inicialmente utilizando-se redes neuronais. Os resultados obtidos com esse processo não são satisfatórios quando ocorrem diferenças expressivas entre os padrões utilizados no treinamento e os padrões utilizados no reconhecimento. As diferenças que podem ocorrer entre os padrões utilizados são: mudança de escala, translação e rotação. A mudança de escala ocorre quando as dimensões do padrão utilizado no treinamento são diferentes das dimensões do padrão

utilizado no reconhecimento. A translação ocorre quando o padrão utilizado no treinamento encontra-se localizado em uma posição distinta à posição do padrão utilizado no reconhecimento. A rotação ocorre quando os padrões utilizados no treinamento e no reconhecimento apresentam inclinações diferentes. Com o objetivo de minimizar essas diferenças, os dígitos utilizados antes de serem treinados e reconhecidos pela rede neuronal são normalizados, tornando-se insensíveis a mudança de escala, translação e rotação.

Vários testes foram realizados utilizando-se padrões normalizados. Os resultados obtidos nesses testes foram satisfatórios quando na fase de treinamento os padrões utilizados foram escritos pelas mesmas pessoas que escreveram os padrões utilizados na fase de reconhecimento. Porém, para que 90% ou mais dos padrões testados fossem reconhecidos, foi necessário aumentar a quantidade de padrões utilizados no processo de treinamento da rede neuronal. Esse aumento resultou em uma elevação considerável no tempo de processamento da rede neuronal na fase de treinamento. Ao serem utilizados na fase de reconhecimento padrões escritos por pessoas diferentes das pessoas que escreveram os padrões utilizados na fase de treinamento, a taxa de reconhecimento apresentou expressiva redução; sendo que em alguns casos essa taxa chegou a ser igual a 57,75%. Devido a este baixo percentual de acerto e aos problemas encontrados na fase de reconhecimento quando ocorrem diferenças expressivas entre o padrão utilizado na fase de treinamento e o padrão utilizado na fase de reconhecimento, resolveu-se desenvolver um novo algoritmo de reconhecimento de caracteres manuscritos que é totalmente diferente do algoritmo de reconhecimento utilizado na rede neuronal.

O processo utilizado pelo novo algoritmo de reconhecimento de caracteres manuscritos que é a base deste trabalho tem como princípio a verificação das características intrínsecas de cada dígito e a verificação das características que os dígitos têm em comum. Ao analisar estas características, esse algoritmo procurou

aproximar-se do processo de reconhecimento utilizado pelo ser humano. Com o objetivo de facilitar a definição das características intrínsecas e comuns dos dígitos, o esqueleto de cada um destes dígitos é obtido utilizando-se o algoritmo rápido de afilamento (ARA) desenvolvido neste trabalho.

O algoritmo ARA possui dois passos principais, que são repetidos até que a imagem obtida aproxime-se do eixo médio da imagem original. No primeiro passo é marcado o contorno da imagem e no segundo passo o contorno marcado é analisado visando verificar quais os "pixels-on" pertencentes a este contorno devem ser apagados.

Com o objetivo de verificar a eficiência do algoritmo ARA foram realizados vários testes. Através da análise dos resultados obtidos nesses testes, observa-se que o algoritmo ARA mantém a conectividade e preserva os pontos finais da imagem original, é praticamente imune ao ruído e, de extrema importância, têm rápido tempo de execução. Os resultados obtidos pelo algoritmo ARA foram comparados com os resultados obtidos pelos algoritmos OPTA ("One-Pass Thinning Algorithm") e PSTA ("Paralled-Symmetric Thinning Algorithm"). Nestes testes foi observado que o algoritmo ARA é aproximadamente 3,0 vezes mais rápido do que o algoritmo PSTA, e 3,3 a 5,3 vezes mais rápido do que o algoritmo OPTA.

O algoritmo de reconhecimento de caracteres numéricos manuscritos desenvolvido neste trabalho, parte do topo da imagem de cada dígito e agrupa os "pixels-on" em linhas horizontais e linhas verticais. É a partir das coordenadas iniciais e finais dessas linhas que são definidas as características intrínsecas e comuns de cada um dos dígitos. O algoritmo considera que em cada dígito existem duas linhas horizontais, linha horizontal superior e linha horizontal inferior; duas linhas verticais, linha vertical esquerda e linha vertical direita e podem ainda existir um ou mais pontos de cruzamento.

Visando testar a eficiência do algoritmo de reconhecimento de caracteres

manuscritos, inicialmente foram realizados testes com os dígitos de 0 a 9, escritos por 30 pessoas distintas, totalizando 300 dígitos. Foi solicitado a cada uma dessas pessoas que escrevessem os dígitos isolados uns dos outros. Dos 300 dígitos que foram escritos, 299 foram reconhecidos corretamente e apenas 1 foi reconhecido de maneira incorreta pelo algoritmo. Com estes resultados o algoritmo de reconhecimento de caracteres numéricos manuscritos apresentou uma porcentagem de acerto de 99,67%.

Em nossa vida diária existem várias aplicações para o reconhecimento de caracteres numéricos manuscritos, uma dessas aplicações é o reconhecimento da quantia numérica de cheques bancários. Neste trabalho, com o objetivo de comprovar a eficiência do algoritmo de reconhecimento desenvolvido, além de serem realizados testes com dígitos isolados, foram realizados testes em que a quantia numérica de cheques bancários era reconhecida. Para que estes testes pudessem ser realizados, foi necessário adicionar ao algoritmo de reconhecimento, algoritmos que realizassem a segmentação de caracteres sobrepostos e caracteres conectados, eliminação de vírgulas, pontos, cerquilhas, 'R$' impressos, 'R$' manuscritos e campo que envolve a quantia numérica.

O algoritmo de segmentação de caracteres sobrepostos desenvolvido neste trabalho, realiza a segmentação destes caracteres varrendo o caractere em toda a sua extensão. Este processo é realizado varrendo a matriz-original a partir da coordenada CILHI e copiando os "pixels-on" e os "pixels-off" varridos para a nova-matriz. Ao final desse processo a nova-matriz contém um caractere isolado ou caracteres conectados.

O algoritmo de segmentação de caracteres conectados desenvolvido neste trabalho, verifica se a nova-matriz contém um caractere isolado ou caracteres conectados, através da análise da quantidade de linhas horizontais superior e inferior que são encontradas na nova-matriz e dos pontos de cruzamento encontrados no intervalo compreendido entre 80% e 140% da largura média dos

caracteres contidos nessa matriz.

Nos testes realizados com cheques bancários foram utilizados 300 cheques escritos por 75 pessoas distintas. O número total de dígitos obtidos desses 300 cheques foi igual a 1212. Dos 1212 dígitos utilizados, 1211 foram reconhecidos corretamente e apenas um dígito isolado foi reconhecido de maneira incorreta pelo algoritmo de reconhecimento de caracteres numéricos manuscritos. Foi observado que o processo utilizado pelo algoritmo de reconhecimento na segmentação de dígitos sobrepostos e conectados possibilitou que o algoritmo de reconhecimento apresentasse uma taxa de reconhecimento de 99,92%. Esta taxa é um pouco superior as taxas de reconhecimento de sistemas desenvolvidos atualmente, que estão normalmente entre 93,00% a 99,50%.

As principais características que o algoritmo de reconhecimento de caracteres manuscritos desenvolvido neste trabalho possui são:

a) o reconhecimento dos caracteres manuscritos independe da pessoa que os escreveu;

b) o reconhecimento dos caracteres independe do seu tamanho e da posição em que o caractere foi escrito;

c) desempenho satisfatório, mesmo na presença de ruído;

d) taxa de acerto elevada.

Ao analisar os resultados obtidos nos testes realizados com o algoritmo de reconhecimento de caracteres numéricos manuscritos, foi observado que a resolução utilizada e o realce conseguido ao ser utilizado o "scanner" no processo de digitalização da imagem influem de maneira decisiva no processo de

reconhecimento dos caracteres manuscritos, pois quanto melhor a qualidade da imagem obtida através do processo de digitalização, menor é a probalidade de ocorrerem ruídos ou deformações na imagem e, portanto, maior é a eficiência obtida no processo de reconhecimento.

6.2 - CONTRIBUIÇÕES DO TRABALHO

1 - Desenvolvimento de um algoritmo que torna os dígitos insensíveis a mudança de escala, translação e rotação e utilizando redes neuronais realiza o reconhecimento desses dígitos.

2 - Desenvolvimento de um algoritmo rápido de afilamento que mantém a conectividade e preserva os pontos finais da imagem original, é praticamente imune ao ruído e, de extrema importância tem rápido tempo de execução.

3 - Desenvolvimento de um algoritmo para o reconhecimento de caracteres numéricos manuscritos, que podem estar isolados, sobrepostos ou conectados e apresentar ao longo do seu contorno trechos contínuos ou fragmentados. O processo de reconhecimento do algoritmo desenvolvido tem como base a verificação das características intrínsecas de cada dígito e a verificação das características que os dígitos têm em comum. Ao verificar estas características, o algoritmo desenvolvido procurou aproximar-se do processo de reconhecimento utilizado pelo ser humano.

6.3 - SUGESTÕES PARA FUTUROS TRABALHOS

No Brasil, o número de trabalhos desenvolvidos na área de reconhecimento de caracteres numéricos manuscritos através da utilização de computadores,

ainda é reduzido. Com o objetivo de suprir essa deficiência foi idealizado e desenvolvido este trabalho.

Analisando-se as inúmeras possibilidades de desenvolvimento de programas na área de reconhecimento de caracteres e tomando-se como base o algoritmo de reconhecimento deste trabalho, pode-se propor alguns dos trabalhos que poderão ser desenvolvidos futuramente:

1 - Desenvolvimento de procedimentos que possibilitem que o algoritmo de reconhecimento de caracteres numéricos manuscritos reconheça caracteres numéricos impressos.

2 - Desenvolvimento de um algoritmo que localize os campos do cheque bancário. Onde os principais campos a serem localizados devem ser: o campo da quantia numérica, o campo do extenso da quantia numérica, o campo que contém a assinatura da pessoa que preencheu o cheque e o campo que identifique a localidade onde o cheque foi emitido.

3 - Desenvolvimento de um algoritmo que reconheça o extenso da quantia numérica escrita quando o cheque bancário é preenchido e compare esse extenso com o extenso fornecido pelo algoritmo de reconhecimento desenvolvido neste trabalho.

4 - Desenvolvimento de um algoritmo que reconhece a assinatura da pessoa que preencheu o cheque bancário.

5 - Desenvolvimento de um algoritmo que reconhece a localidade onde o cheque foi emitido.

6 - Desenvolvimento de um hardware que em conjunto com o algoritmo de

reconhecimento desenvolvido neste trabalho e os algoritmos propostos nos itens 1, 2, 3, 4 e 5 seja capaz de reconhecer a quantia numérica e o extenso desta quantia. Também seja capaz de comparar o extenso da quantia numérica com o extenso encontrado pelo algoritmo de reconhecimento desenvolvido neste trabalho, reconhecer a assinatura da pessoa que preencheu o cheque e classificar de acordo com a localidade onde o cheque foi emitido. Todo esse processo deve ser realizado sem a intervenção do homem, que apenas se encarregaria da colocação dos pacotes de cheques a serem reconhecidos.

7 - Desenvolvimento de um algoritmo que reconhece o código de endereçamento postal, onde os números que constituem esse código podem ser manuscritos ou impressos.

8 - Desenvolvimento de um algoritmo que localize o código de endereçamento postal em envelopes de carta.

9 - Desenvolvimento de um hardware que em conjunto com os algoritmos propostos nos itens 7 e 8 seja capaz de localizar e reconhecer o código de endereçamento postal de envelopes de carta.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] Govindan, V. K., Schivaprasad, A. P.; *Character Recognition - A Review*, Pattern Recognition, Great Britain, vol. 23, n. 7, pp. 671 - 683, July 1990.

[2] Kimura, F., Shridhar, M.; *Handwritten Numeral Recognition Based on Multiple Algorithms*, Pattern Recognition, Great Britain, vol. 24, n. 10, pp. 969 - 983, October 1991.

[3] Lam, L., Suen, C. Y.; *Structural Classification and Relaxation Matching of Totally Unconstrained Handwritten ZIP-Code Numbers*, Paterrn Recognition, Great Britain, vol. 21, n. 1, pp. 19 - 31, January 1988.

[4] Fu, K. S., Keidel, W. D., Wolter, H.; *Introduction to Syntatic Pattern Recognition In: Syntactic Pattern Recognition, Applications*, 1ª.ed, Springer-Verlag, New York, pp. 5-6, 1977.

[5] Westall, J. M., Narasimha, M. S.; *Vertex Directed Segmentation of Handwritten Numerals*, Pattern Recognition, Great Britain, vol. 26, n. 10, pp. 1473 - 1486, October 1993.

[6] Shridhar, M., Badreldin, A.; *Recognition of Isolated and Simply Connected*

Handwritten Numerals, Pattern Recognition, Great Britain, vol. 19, n. 1, pp. 1 - 12, January 1986.

[7] Shridhar, M., Badreldin, A.; *Context-directed Segmentation Algorithm for Handwritten Numeral Strings*, Image Vision Computer, Inglaterra, vol. 5, n. 1, pp. 3 - 9, January 1987.

[8] Fujisawa, H., Nakano, Y., Kurino, K.; *Segmentation Methods for Character Recognition: From Segmentation to Document Structure Analysis*, Proceedings of the IEEE, Piscataway, vol. 80, n. 7, pp. 1079 - 1092, July 1992.

[9] Shridhar, M., Badreldin, A.; *High Accuracy Character Recognition Algorithm Using Fourier and Topological Descriptors*, Pattern Recognition, Great Britain, vol. 17, n. 5, pp. 515 - 524, May 1984.

[10] Shridhar, M., Badreldin, A.; *A High Accuracy Syntatic Recognition Algorithm for Handwritten Numerals*, IEEE Transactions on Systems, Man. and Cybernetics, New York, vol. 15, n.1, pp.152 - 158, January 1985.

[11] Suen, C. Y., Nadal, C., Legault, R., Mai, T. A, Lam, L.; *Computer Recognition of Unconstrained Handwritten Numerals*, Proceedings of the IEEE, Piscataway, vol. 80, n. 7, pp. 1162 - 1180, July 1992.

[12] Fukumi, M., Omatu, S., Takeda, F., Kosaka, T.; *Rotation-Invariant Neural Pattern Recognition System with Application to Coin Recognition*, IEEE Transactions on Neural Network, New York, vol. 3, n. 2, pp. 272 - 279,

March 1992.

[13] Flôres, E. L., Rezende, E. N., Carrijo, G. A., Yabu-uti, J. B. T.; *Reconhecimento de Caracteres Manuscritos Utilizando Redes Neurais*, 1º CONGRESSO BRASILEIRO DE REDES NEURAIS, Itajubá, Brasil, pp. 139 - 144, 24 - 27 Outubro 1994.

[14] Flôres, E. L., Rezende, E. N., Carrijo, G. A., Yabu-uti, J. B. T; *Reconhecimento de dígitos manuscritos insensíveis a mudança de escala, translação, rotação usando redes neuronais*, Revista Ciência & Engenharia, Uberlândia, Brasil, n. 1, pp. 83 - 89, Março 1995.

[15] Yüceer, C., Oflazer, K.; *A Rotation, Scaling and Translation Invariant Pattern Classification System*, Pattern Recognition, Great Britain, vol. 26, n. 5, pp. 687 - 710, May 1993.

[16] Jain, A. K.; *Fundamentals of Digital Image Processing*, 1ª.ed, Prentice Hall International Editions, New Jersey, pp. 363, 1989.

[17] Burr, David J.; *Experiments on Neural Net Recognition of Spoken and Written Text*, IEEE Transactions on Acoustics, Speech, and Signal Processing, New York, vol. 36, n. 7, pp. 1162 - 1168, July 1988.

[18] Flôres, E. L., Rezende, E. N., Carrijo, G. A., Yabu-uti, J. B. T.; *A Fast Thinning Algorithm for Characters*, 1995 IEEE WORKSHOP ON NONLINEAR SIGNAL AND IMAGE PROCESSING, Neos Marmaras Halkidiki, Greece, vol. 1, pp. 166 - 169, 20 - 22 June 1995.

[19] Aldus Corporation and Microsoft Corporation; An Aldus/Microsoft Technical Memorandum, 70 p., August 1988.

[20] Yu, S., Tsai, W.; *A New Thinning Algorithm for Gray-Scale Images by the Relaxation Technique*, Pattern Recognition, Great Britain, vol. 23, n. 10, pp. 1067 - 1076, October 1990.

[21] Kwok, P. C. K.; *A Thinning Algorithm by Contour Generation*, Communications of the ACM, New York, vol. 31, n. 11, pp. 1314 - 1324, November 1988.

[22] Bourbakis, N. G.; *A Parallel-Symmetric Thinning Algorithm*, Pattern Recognition, Great Britain, vol. 22, n. 4, pp. 387 - 396, April 1989.

[23] Suzuki, S., Abe, K.; *Binary Picture Thinning by an Iterative Parallel Two-Subcycle Operation*, Pattern Recognition, Great Britain, vol. 20, n. 4, pp. 297 - 307, April 1987.

[24] Li, B., Suen, C. Y.; *A Knowledge-Based Thinning Algorithm*, Pattern Recognition, Great Britain, vol. 24, n. 12, pp. 1211 - 1221, December 1991.

[25] Flôres, E. L., Rezende, E. N., Carrijo, G. A., Yabu-uti, J. B. T.; *An Numerical Characters Recognition Algorithm*, 38TH MIDWEST SYMPOSIUM ON CIRCUITS AND SYSTEMS, Rio de Janeiro, Brazil, vol. 2, pp. 1074-1077 , 13 - 16 August 1995.

[26] Flôres, E. L., Rezende, E. N., Carrijo, G. A., Yabu-uti, J. B. T.; *An Algorithm to Recognize Numerical Handwritten Characters Isolated and*

Overlapped, INTERNATIONAL SYMPOSIUM ON SIGNALS, SYSTEMS AND ELECTRONICS, San Francisco, USA, pp. 505-508, 25 - 27 October, 1995.

[27] Flôres, E. L., Rezende, E. N., Carrijo, G. A., Yabu-uti, J. B. T.; *An Algorithm to Recognize Isolated, Overlapped and Connected Numerical Handwritten Characters*, THIRD AUSTRALIAN AND NEW ZELAND CONFERENCE ON INTELLIGENT INFORMATION SYSTEMS, Perth, Western Australia, pp. 53 - 58, 27 November - 1º December 1995.

[28] Flôres, E. L., Rezende, E. N., Carrijo, G. A., Yabu-uti, J. B. T.; *An Algorithm to Recognize Isolated, Overlapped and Connected Numerical Handwritten Characters*, INTERNATIONAL CONFERENCE ON NEURAL NETWORKS AND SIGNAL PROCESSING, Nanjing, China, vol. II, pp. 1439 - 1442, 10 - 13 December, 1995.

UNIVERSIDADE ESTADUAL DE CAMPINAS
FACULDADE DE ENGENHARIA ELÉTRICA E DE COMPUTAÇÃO
DEPARTAMENTO DE COMUNICAÇÕES

RECONHECIMENTO DE CARACTERES NUMÉRICOS MANUSCRITOS QUE CONSIDERA AS CARACTERÍSTICAS INTRÍNSECAS E COMUNS DE CADA DÍGITO

Este exemplar corresponde à [illegible] da tese
defendida por EDNA LUCIA FLORES
e aprovada pela Comissão
Julgador 04/02/97

Autora: Edna Lúcia Flôres
Orientador: Prof. Dr. João Baptista Tadanobu Yabu-uti

2º volume da tese apresentada à Faculdade de Engenharia Elétrica e de Computação da Universidade Estadual de Campinas, como parte dos requisitos exigidos para a obtenção do título de **Doutora em Engenharia Elétrica**

Fevereiro - 1997

UNIDADE BC
N.º CHAMADA:
TUNICAMP
F663r
V. 02 Ex.
TOMBO BC/ 30968
PROC. 281/97
C [ ] D [X]
PREÇO R$ 11,00
DATA 05/03/97
N.º CPD 97-0000 99252-4

FICHA CATALOGRÁFICA ELABORADA PELA
BIBLIOTECA DA ÁREA DE ENGENHARIA - BAE - UNICAMP

F663r

Flôres, Edna Lúcia
Reconhecimento de caracteres numéricos manuscritos que considera as características intrínsecas e comuns de cada dígito / Edna Lúcia Flôres.--Campinas, SP: [s.n.], 1997
2 v.

Orientador: João Baptista Tadanobu Yabu-uti.
Tese (doutorado) - Universidade Estadual de Campinas Faculdade de Engenharia Elétrica e de Computação.

1. Reconhecimento de padrões. 2.Processamento de imagens. I. Yabu-uti, João Baptista Tadanobu. II. Universidade Estadual de Campinas. Faculdade de Engenharia Elétrica e de Computação. III. Título.

RECONHECIMENTO DE CARACTERES NUMÉRICOS MANUSCRITOS QUE CONSIDERA AS CARACTERÍSTICAS INTRÍNSECAS E COMUNS DE CADA DÍGITO

Autora: Edna Lúcia Flôres

2º volume da Tese apresentada à Faculdade de Engenharia Elétrica e de Computação da Universidade Estadual de Campinas, como parte dos requisitos exigidos para a obtenção do título de **Doutora em Engenharia Elétrica**

Banca Examinadora:

Prof. Dr. João Baptista Tadanobu Yabu-uti

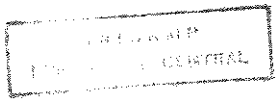
Prof. Dr. Gilberto Arantes Carrijo

Prof. Dr. Yuzo Iano

Prof. Dr. Edson Moschim

Prof. Dr. Lee Luan Ling

UNICAMP, 04 de Fevereiro de 1997



SUMÁRIO DO 2º VOLUME

FLUXOGRAMAS DO CAPÍTULO 3.....327

FLUXOGRAMAS DO CAPÍTULO 4.....352

FLUXOGRAMAS DO CAPÍTULO 5.....571

LISTA DE FIGURAS

Figura 3.9 - Fluxograma da função delimita-contorno.....328

Figura 3.14 - Fluxograma da função apaga-contorno.....332

Figura 3.15 - Fluxograma da função verifica-ruído.....339

Figura 3.17 - Fluxograma da função verifica-conectividade.....342

Figura 3.18 - Fluxograma da função verifica-desvio.....347

Figura 4.1 - Fluxograma que realiza a delimitação das coordenadas inicial e final da linha horizontal superior, quando a matriz-original é varrida da esquerda para a direita.....353

Figura 4.3 - Fluxograma que realiza a delimitação das coordenadas inicial e final da linha horizontal superior, quando a matriz-original é varrida da direita para a esquerda.....364

Figura 4.4 - Fluxograma que realiza a delimitação das coordenadas inicial e final da linha horizontal inferior, quando a matriz-original é varrida da esquerda para a direita.....377

Figura 4.6 - Fluxograma que realiza a delimitação das coordenadas inicial e final da linha horizontal inferior, quando a matriz-original é varrida da direita para a esquerda.....389

Figura 4.7 - Fluxograma que realiza a delimitação das coordenadas inicial e final da linha vertical esquerda.....405

Figura 4.12 - Fluxograma da função vertical.....412

Figura 4.14 - Fluxograma da função horizontal.....423

Figura 4.16 - Fluxograma de todo o processo utilizado no reconhecimento de caracteres numéricos manuscritos isolados.....434

Figura 4.17 - Fluxograma da função verif-1-3-5-6-0.....440

Figura 4.19 - Fluxograma da função 0-8-6-3-5-1-2-4.....462

Figura 4.21 - Fluxograma da função verif-2-3-5-7-9-8-0.....496

Figura 4.23 - Fluxograma da função verif-7-9-4-1-2-0-6-8-3.....538

Figura 4.25 - Fluxograma da função verif-1-4-5-6-7.....559

Figura 5.1 - Fluxograma da função virg-1.....572

Figura 5.2 - Fluxograma da função elim-virg.....575

Figura 5.3 - Fluxograma do processo de eliminação do ponto e/ou ruído.....587

Figura 5.4 - Fluxograma de todo o processo de reconhecimento de caracteres manuscritos na área reservada a escrita da quantia numérica de cheques bancários.....593

Figura 5.6 - Fluxograma da função verif-conec-isolado.....604

Figura 5.7 - Fluxograma da função horiz-conec.....617

Figura 5.8 - Fluxograma da função ver-conec-00.....632

Figura 5.9 - Fluxograma da função verif-HS-HI.....642

Figura 5.10 - Fluxograma da função verif-pc-HS-HI.....645

Figura 5.11 - Fluxograma da função isolad-conec.....653

Figura 5.12 - Fluxograma da função conectados.....668

Figura 5.13 - Fluxograma da função elim-conec.....676

Figura 5.16 - Fluxograma da função tarja-horiz-sup.....685

Figura 5.17 - Fluxograma da função tarja-horiz-inf.....690

Figura 5.18 - Fluxograma da função verifica-tarja.....705

Figura 5.19 - Fluxograma da função tarja-vertical.....721

Figura 5.21 - Fluxograma da função verifica-cerquilha.....734

Figura 5.23 - Fluxograma da função verifica-real.....760

FLUXOGRAMAS DO CAPÍTULO 3

Fluxograma da função delimita-contorno

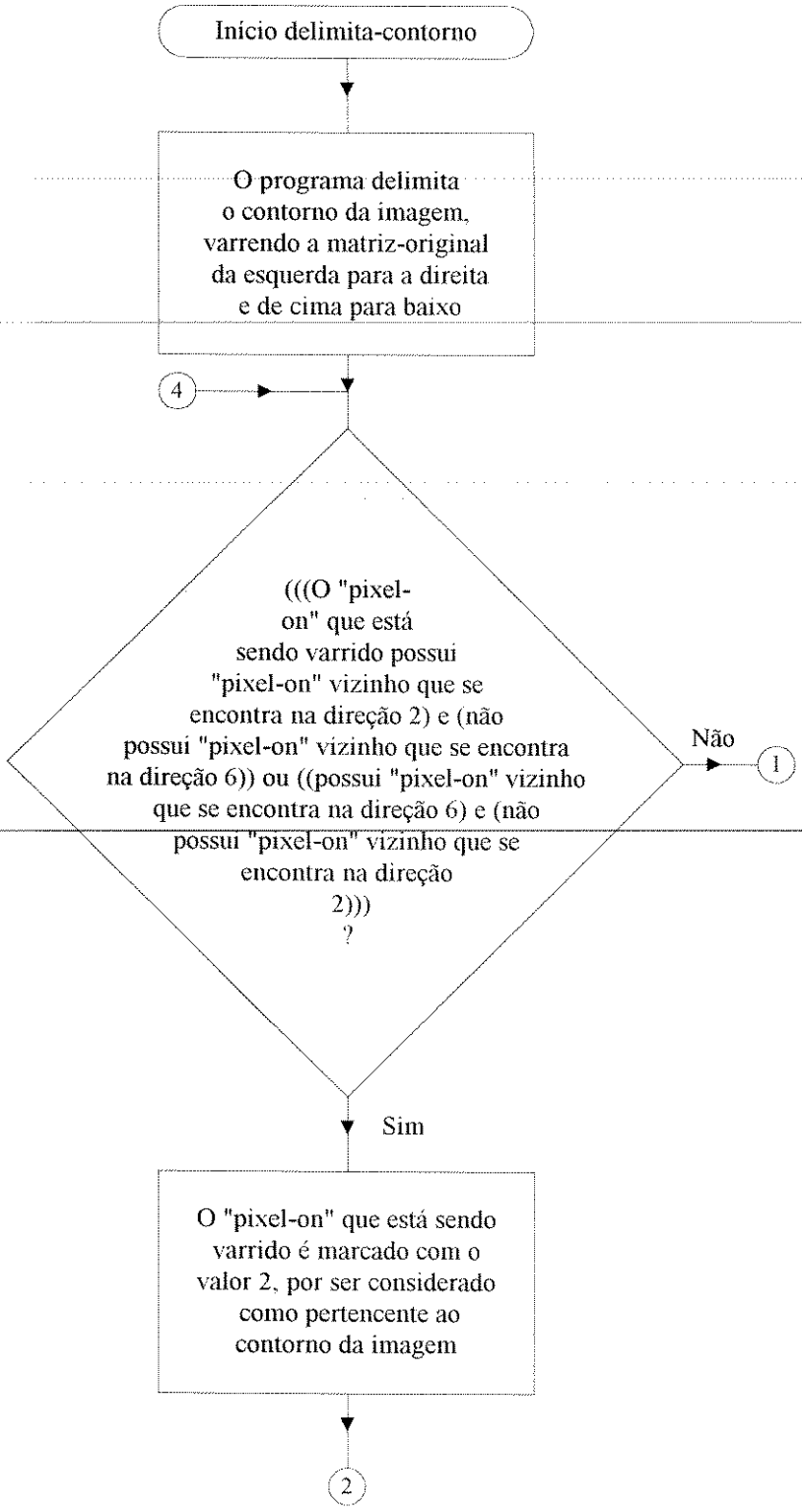


Figura 3.9(a) - Primeira parte do fluxograma da função delimita-contorno.

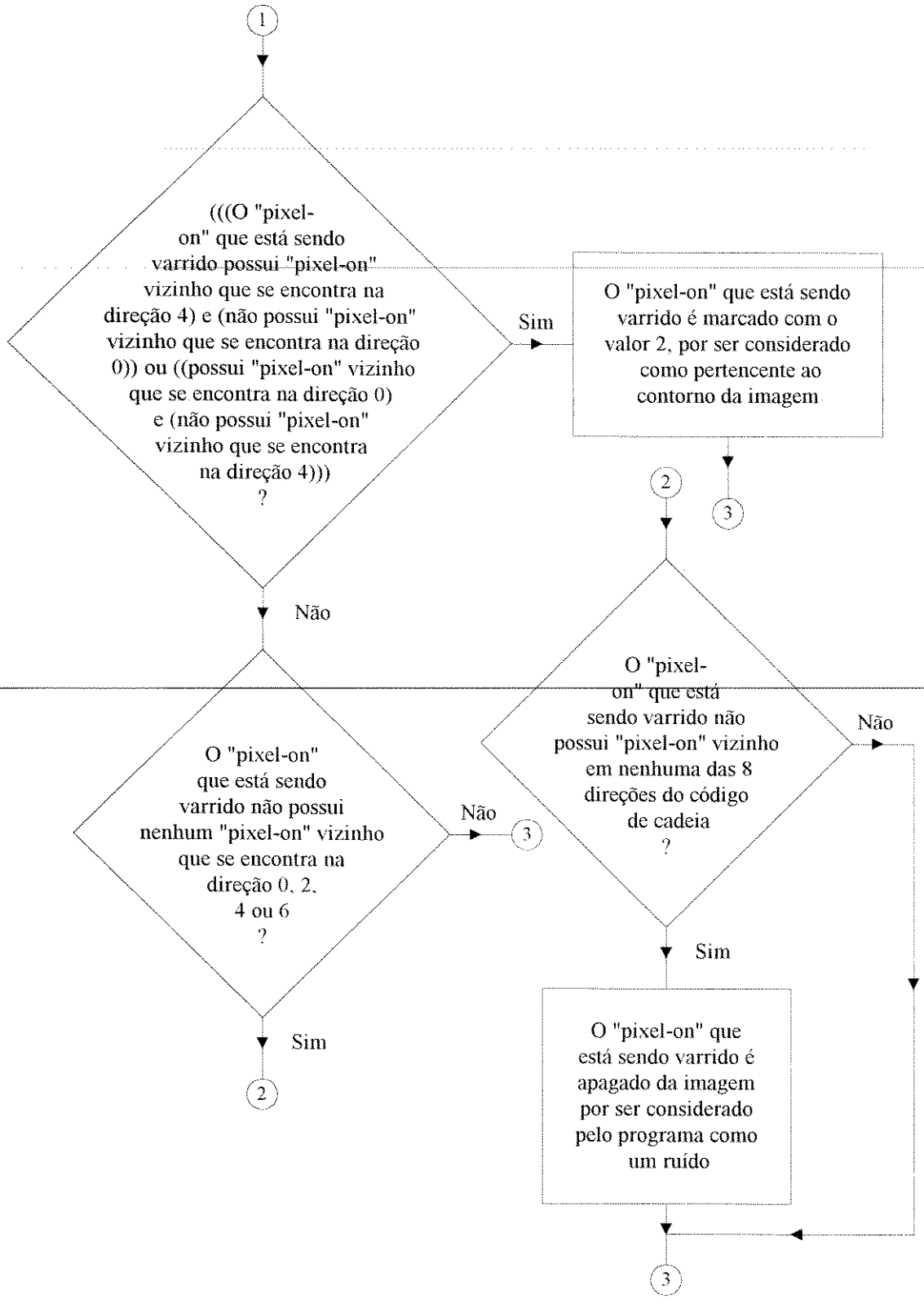


Figura 3.9(b) - Segunda parte do fluxograma da função delimita-contorno.

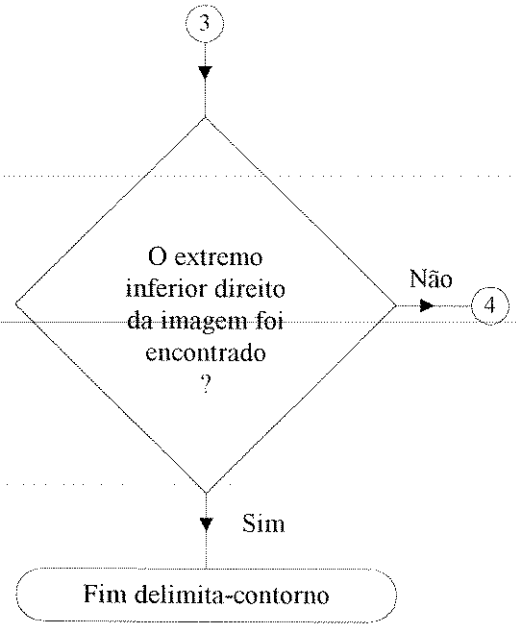


Figura 3.9(c) - Terceira parte do fluxograma da função delimita-contorno.

Fluxograma da função apaga-contorno

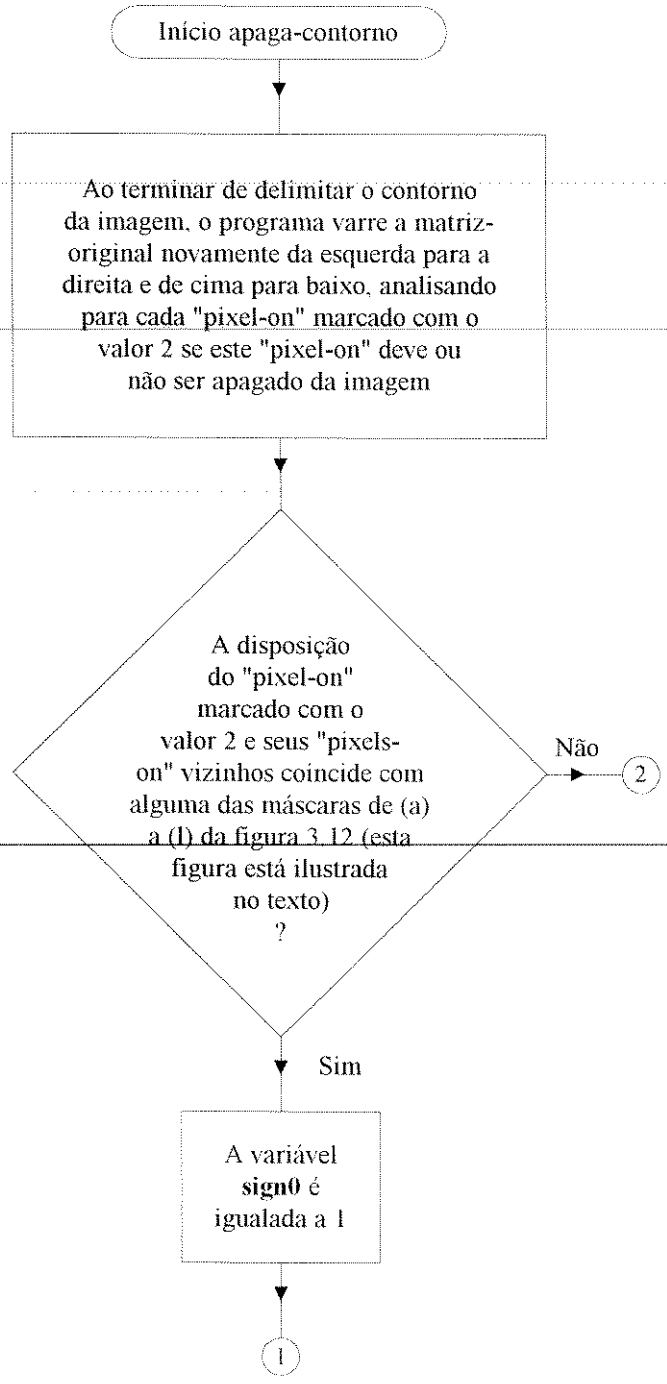


Figura 3.14(a) - Primeira parte do fluxograma da função apaga-contorno.

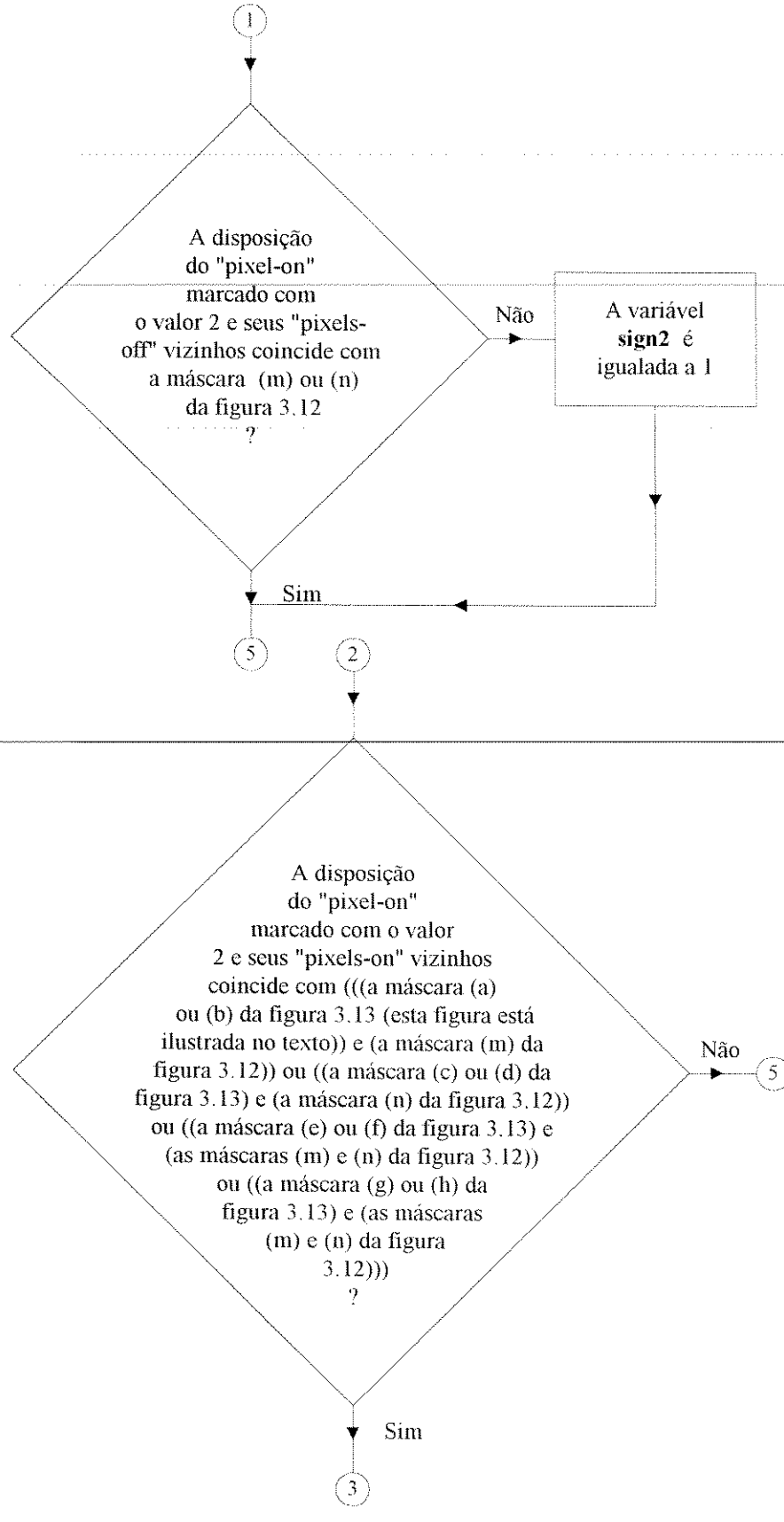


Figura 3.14(b) - Segunda parte do fluxograma da função apaga-contorno.

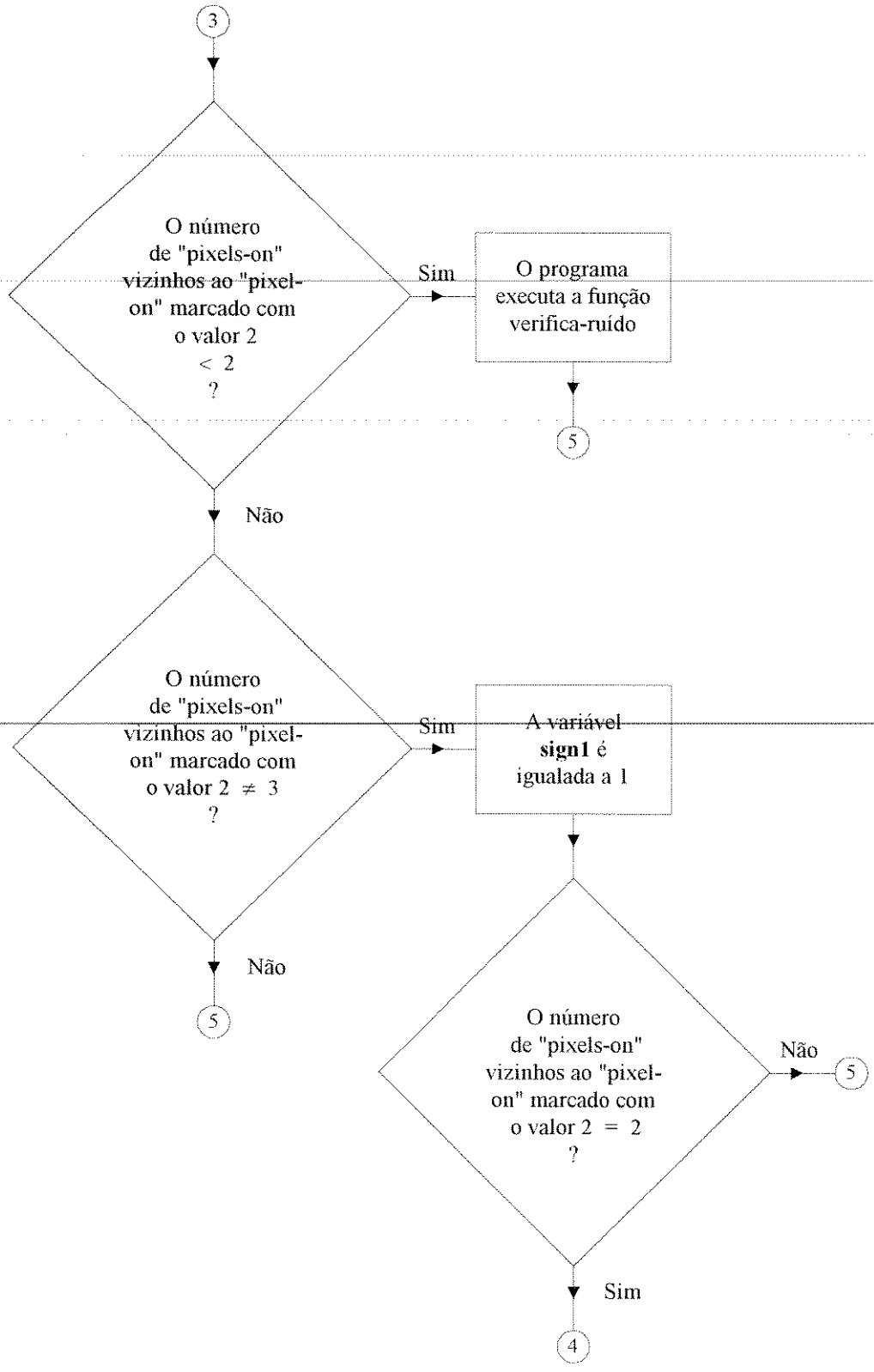


Figura 3.14(c) - Terceira parte do fluxograma da função apaga-contorno.

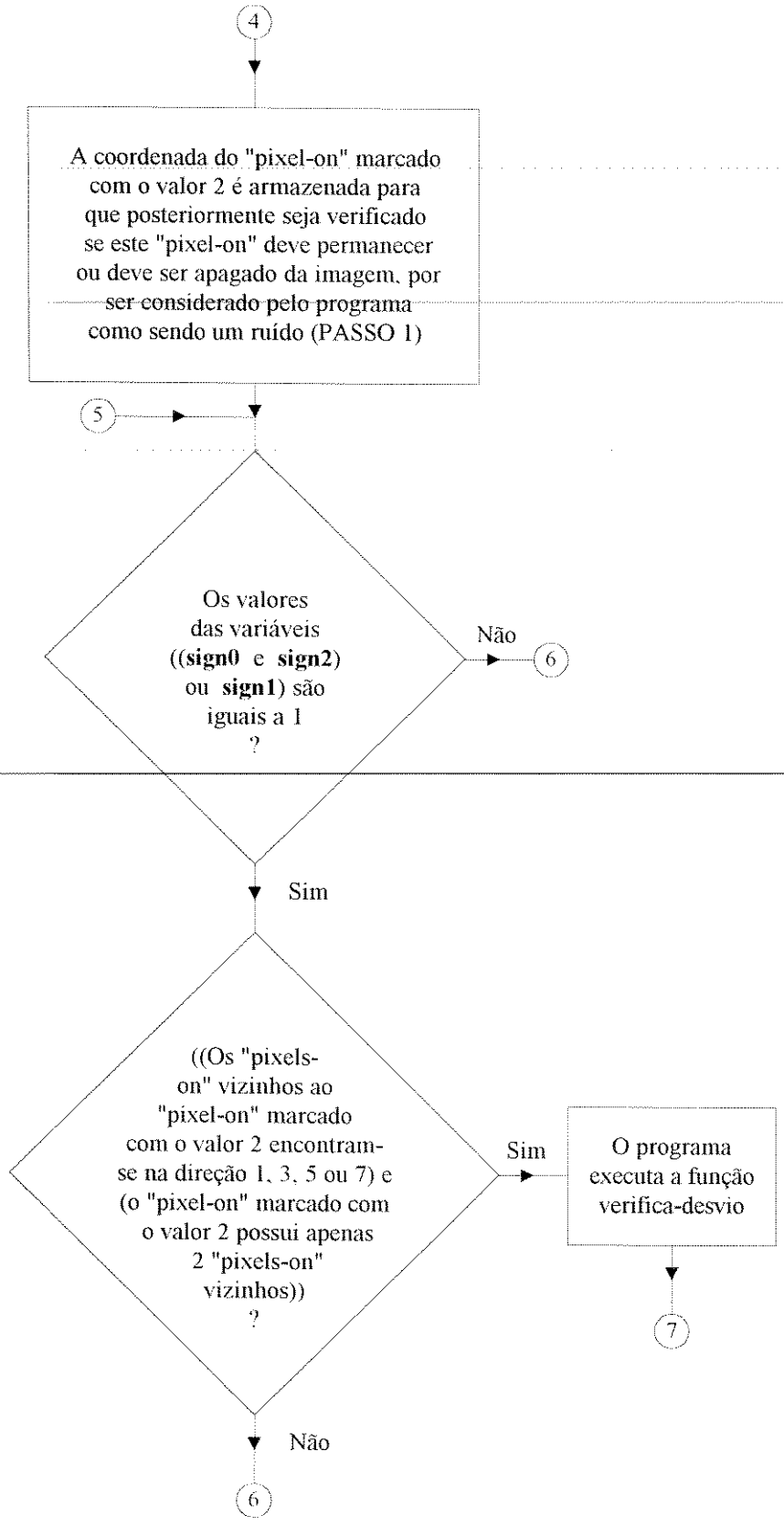


Figura 3.14(d) - Quarta parte do fluxograma da função apaga-contorno.

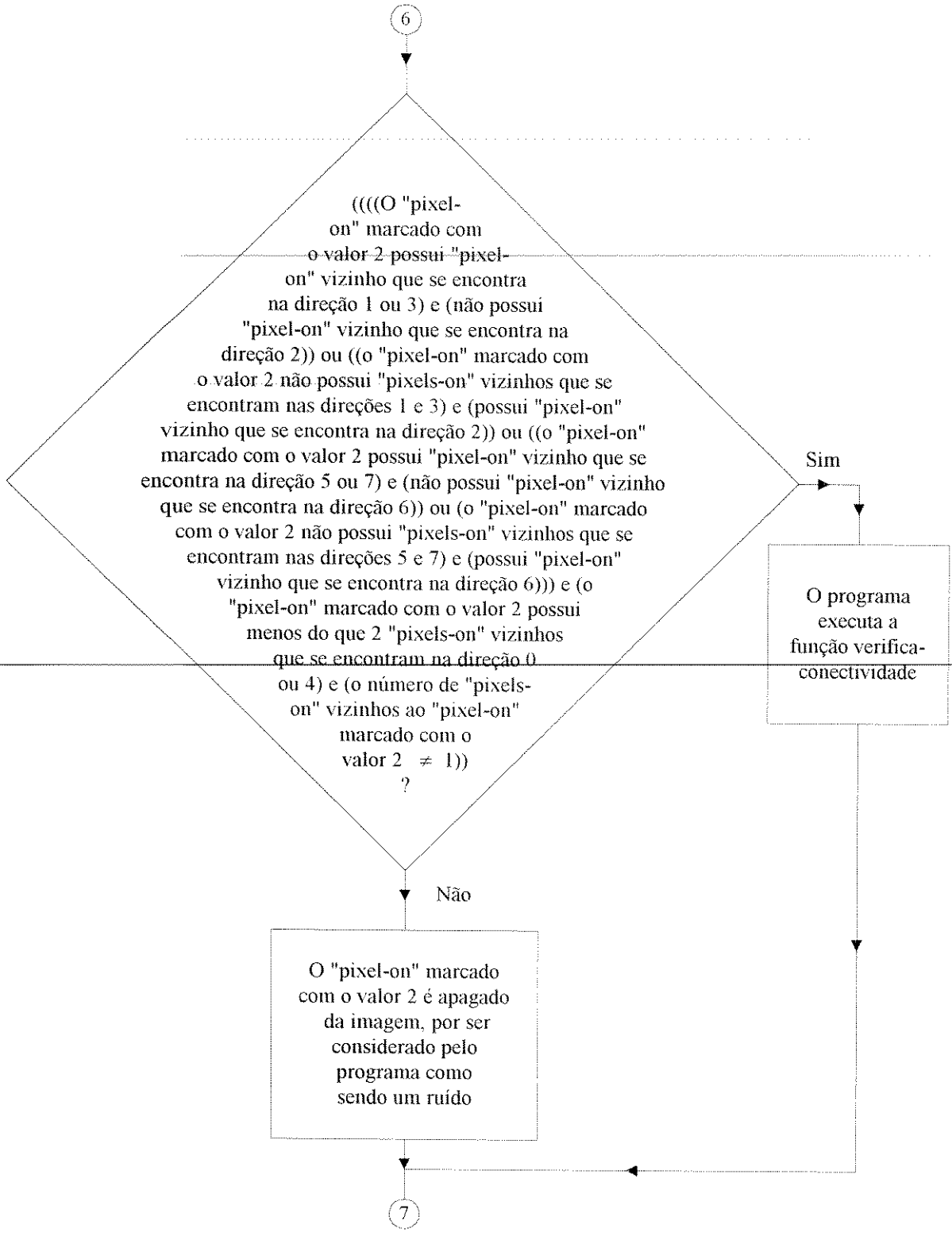


Figura 3.14(e) - Quinta parte do fluxograma da função apaga-contorno.

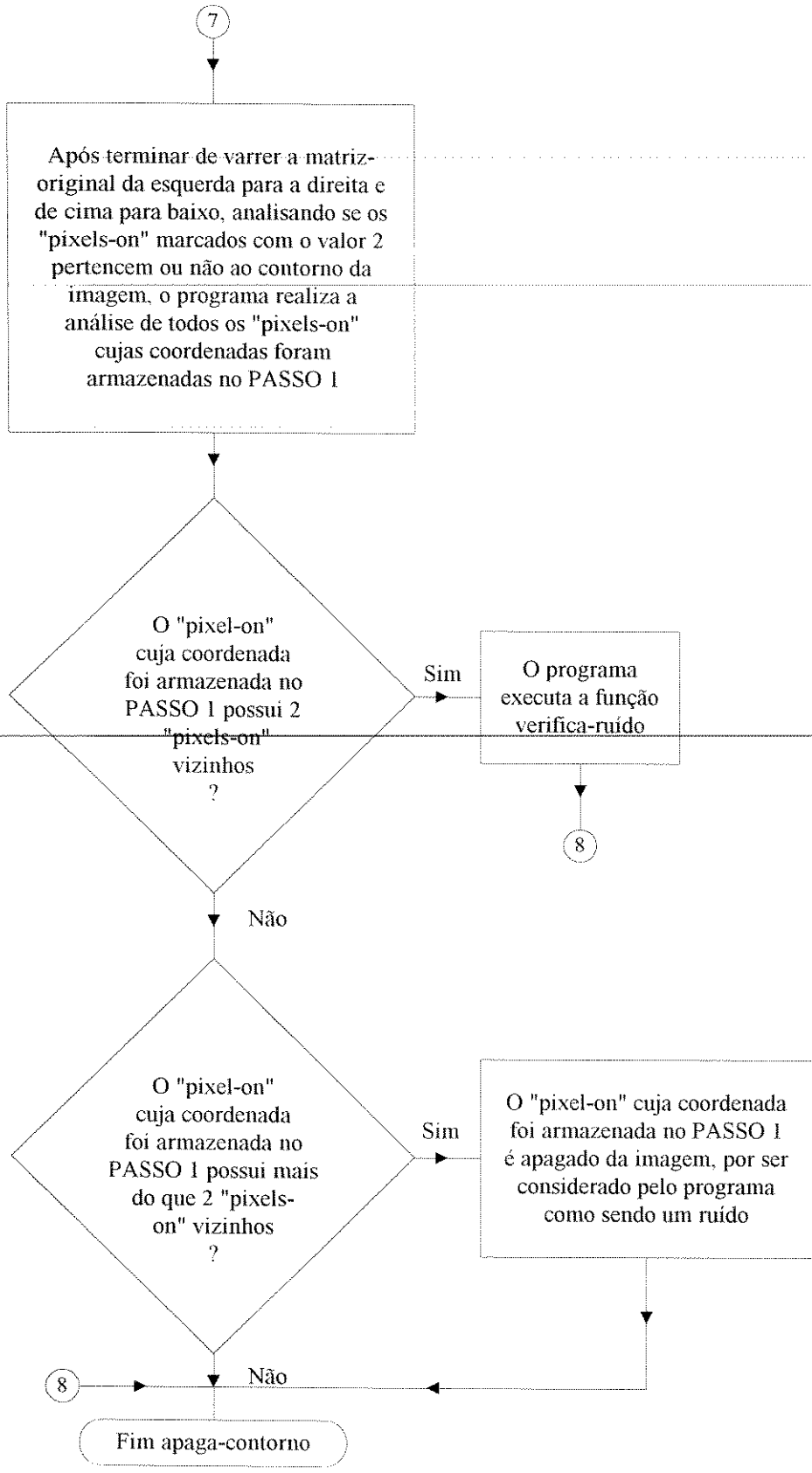


Figura 3.14(f) - Sexta parte do fluxograma da função apaga-contorno.

Fluxograma da função verifica-ruído

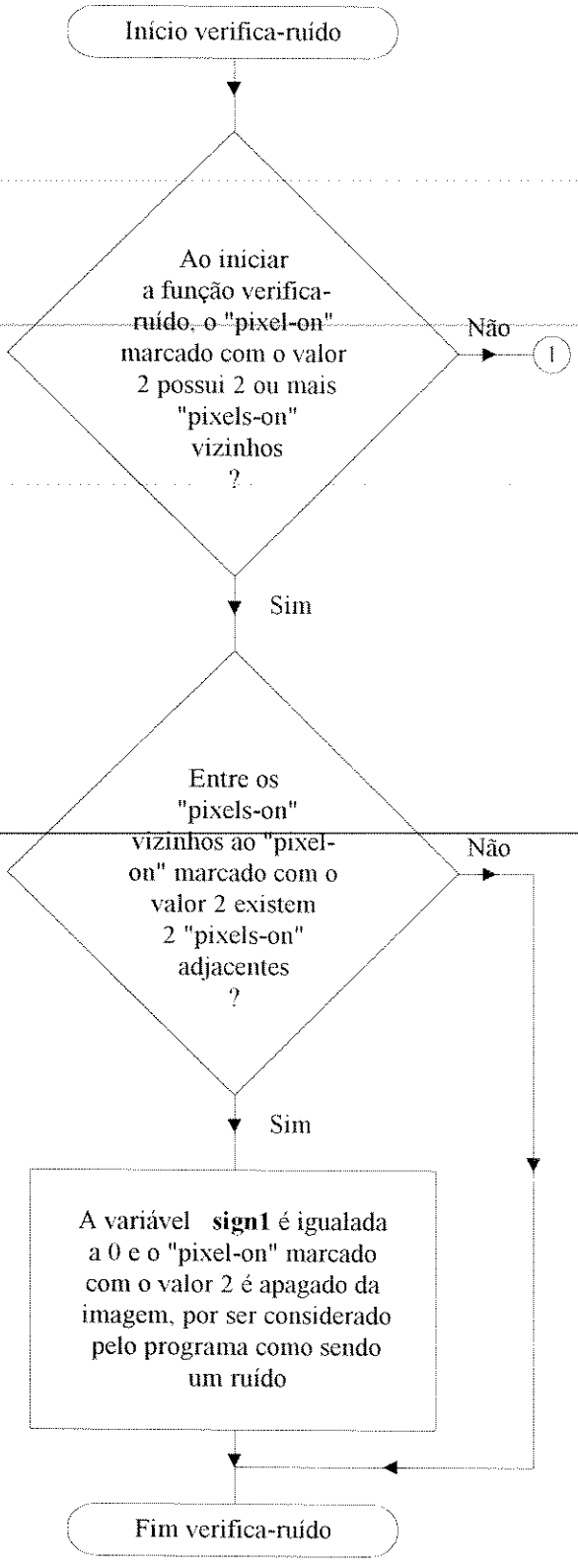


Figura 3.15(a) - Primeira parte do fluxograma da função verifica-ruído.

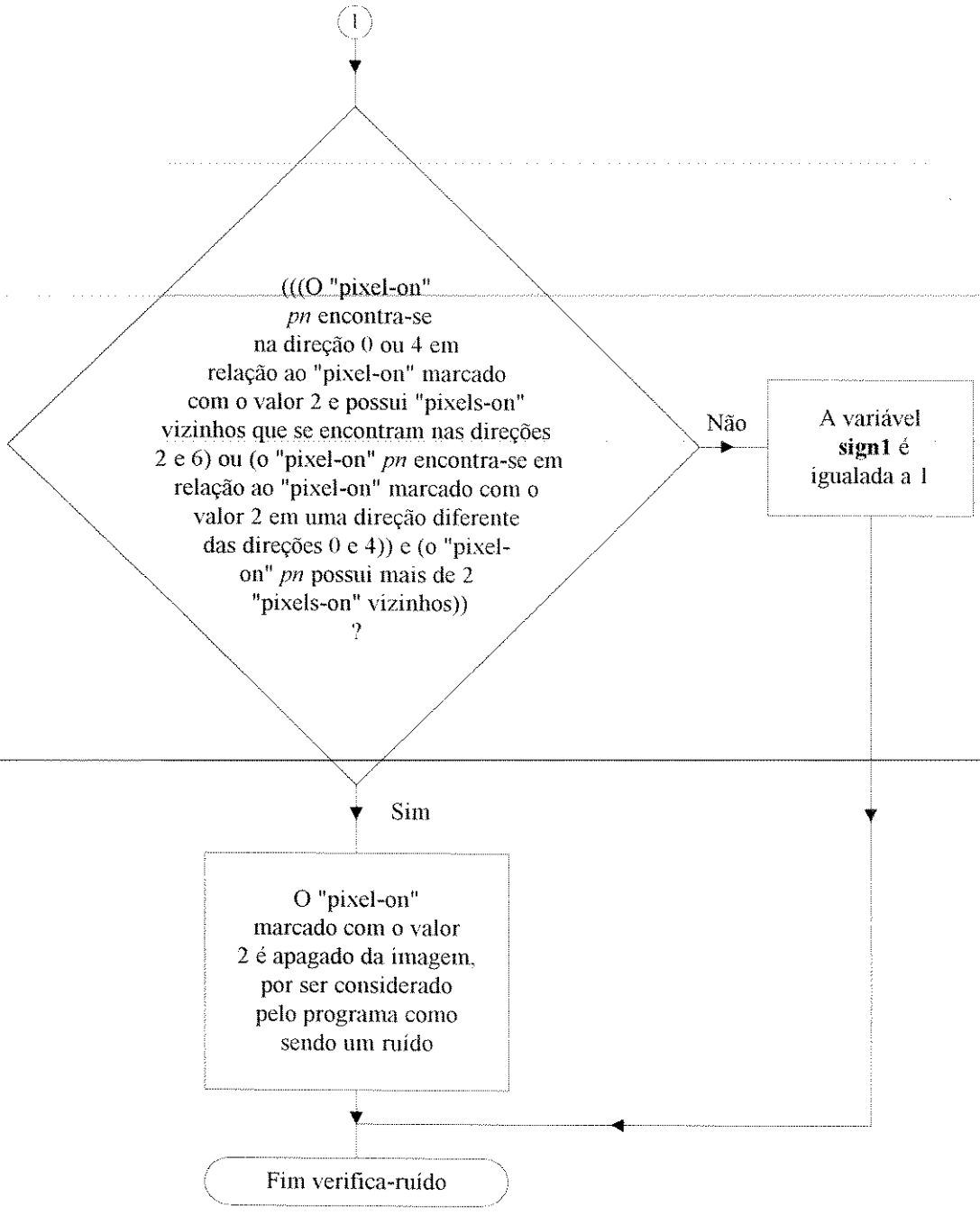


Figura 3.15(b) - Segunda parte do fluxograma da função verifica-ruído.

Fluxograma da função verifica-conectividade

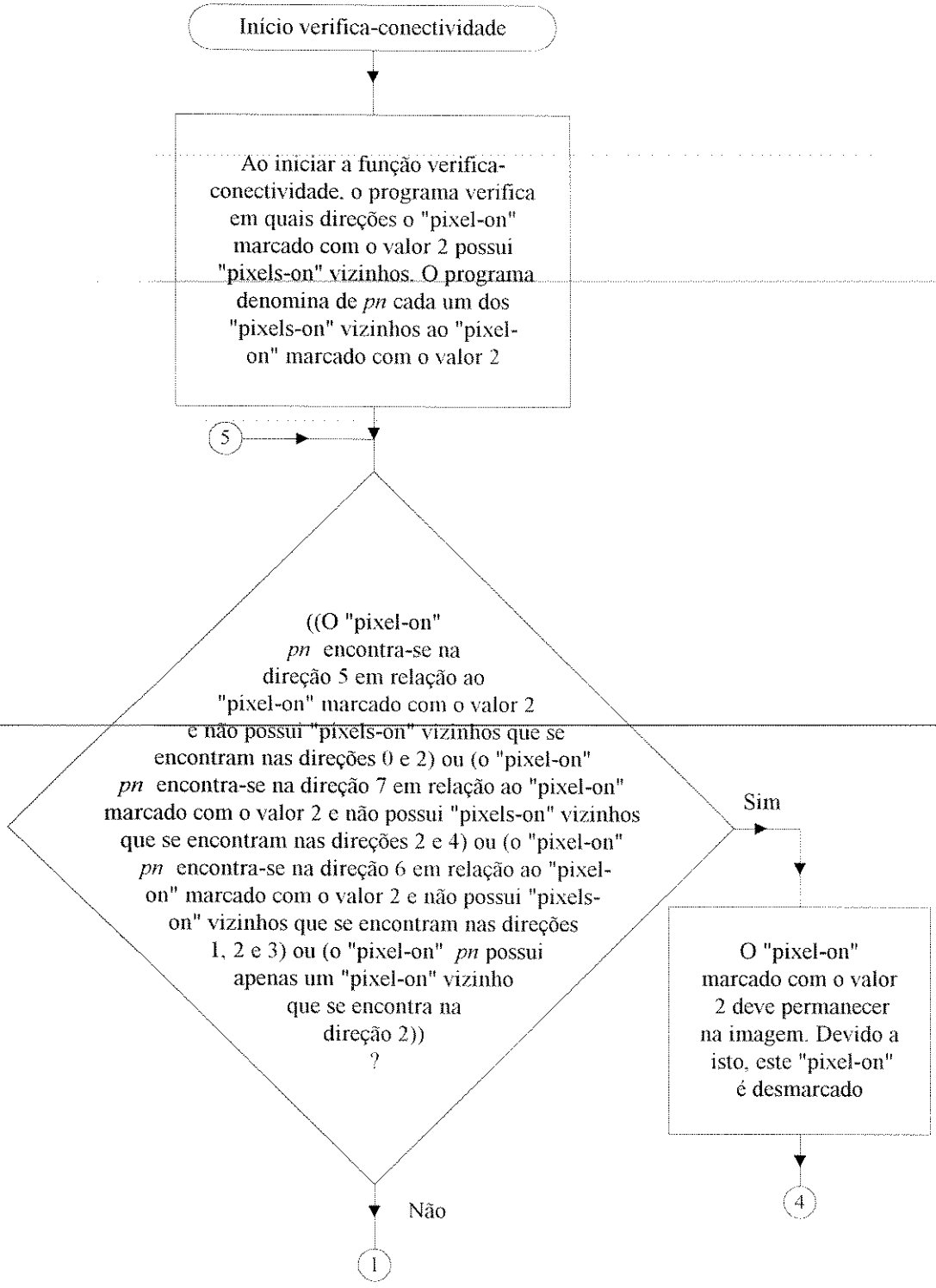


Figura 3.17(a) - Primeira parte do fluxograma da função verifica-conectividade.

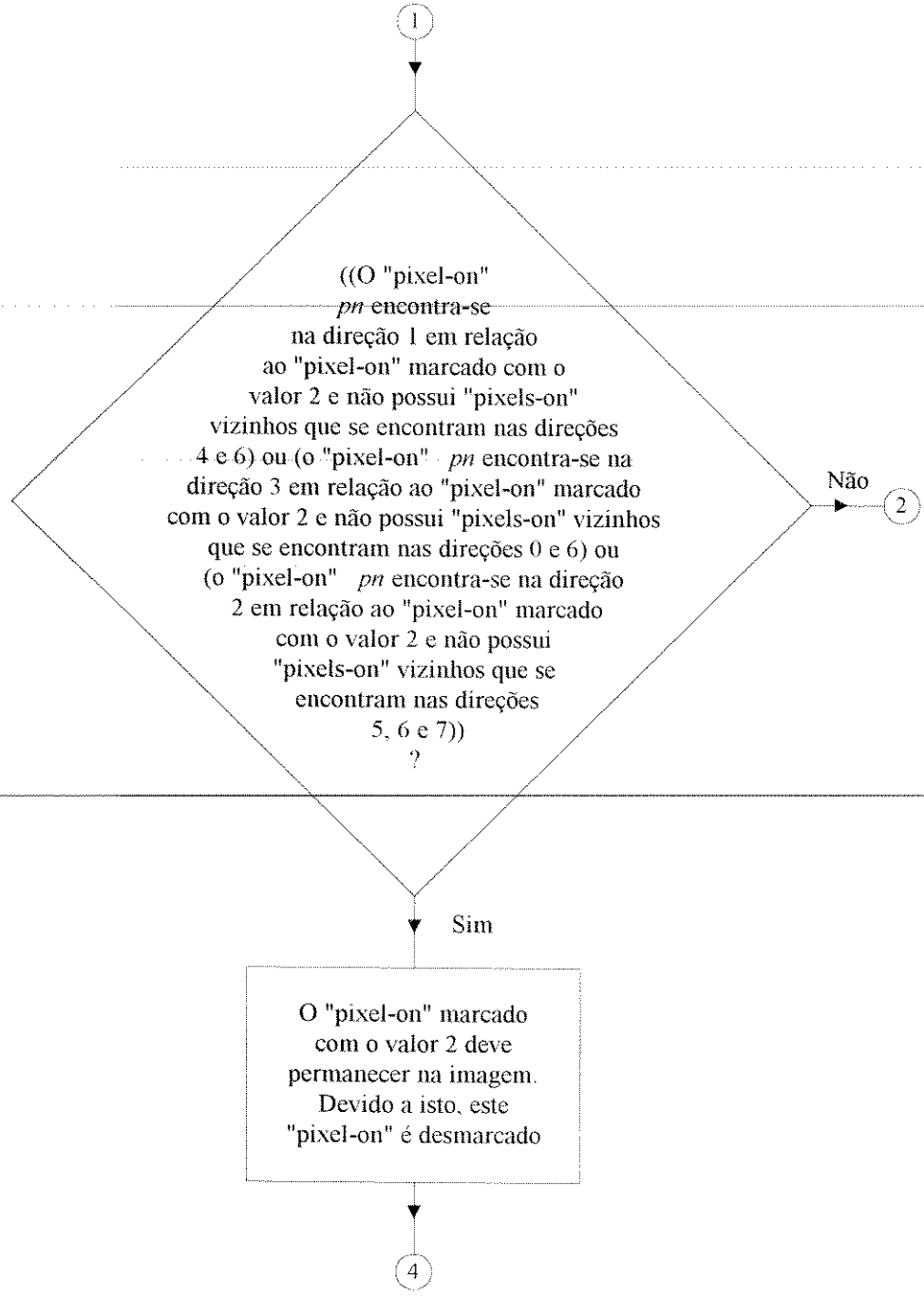


Figura 3.17(b) - Segunda parte do fluxograma da função verifica-conectividade.

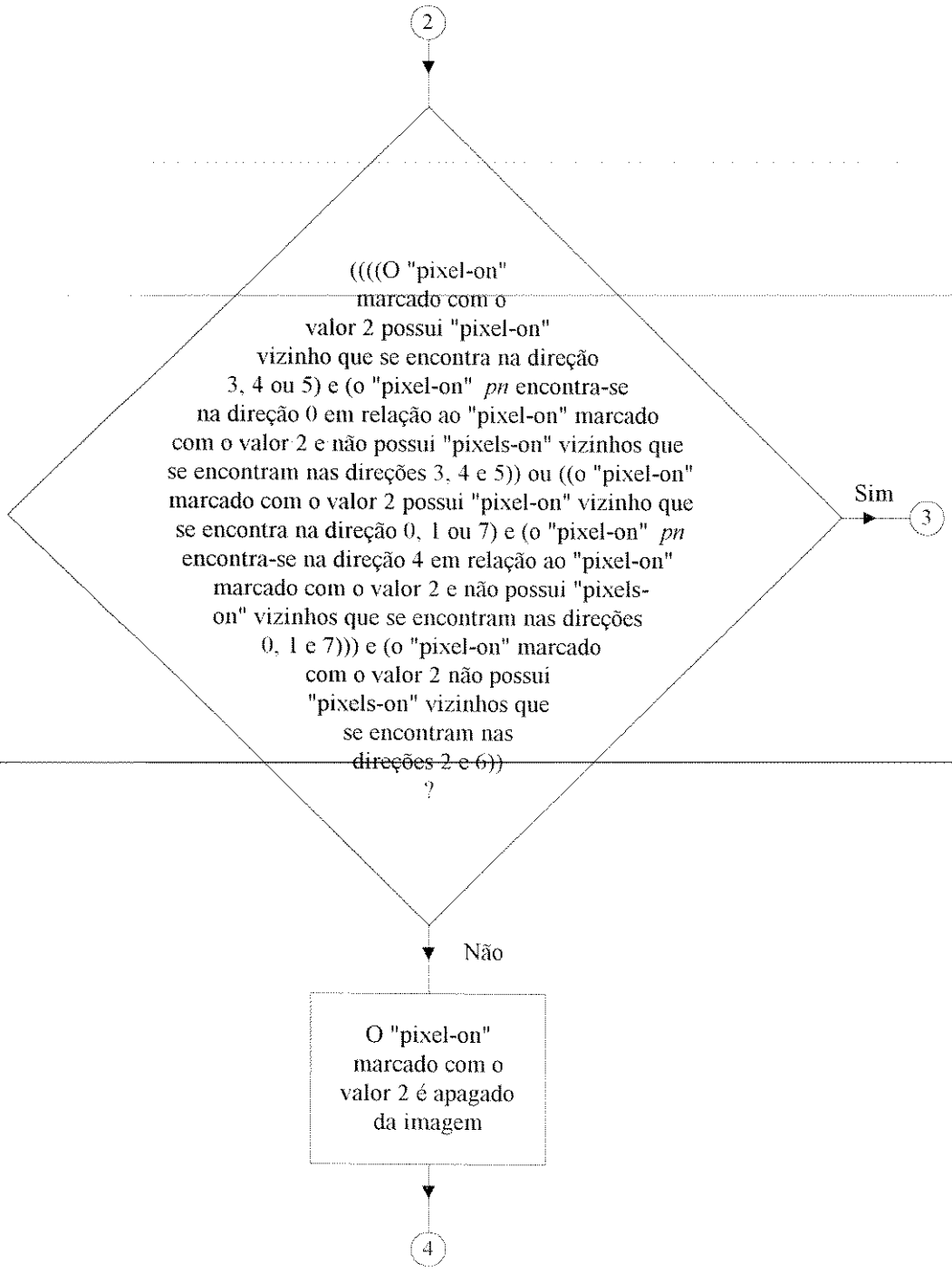


Figura 3.17(c) - Terceira parte do fluxograma da função verifica-conectividade.

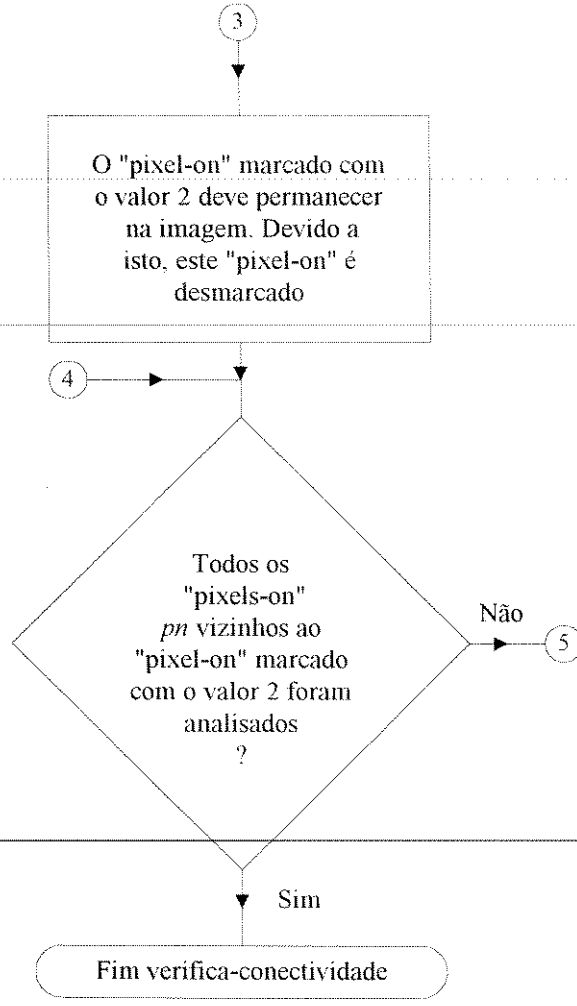


Figura 3.17(d) - Quarta parte do fluxograma da função verifica-conectividade.

Fluxograma da função verifica-desvio

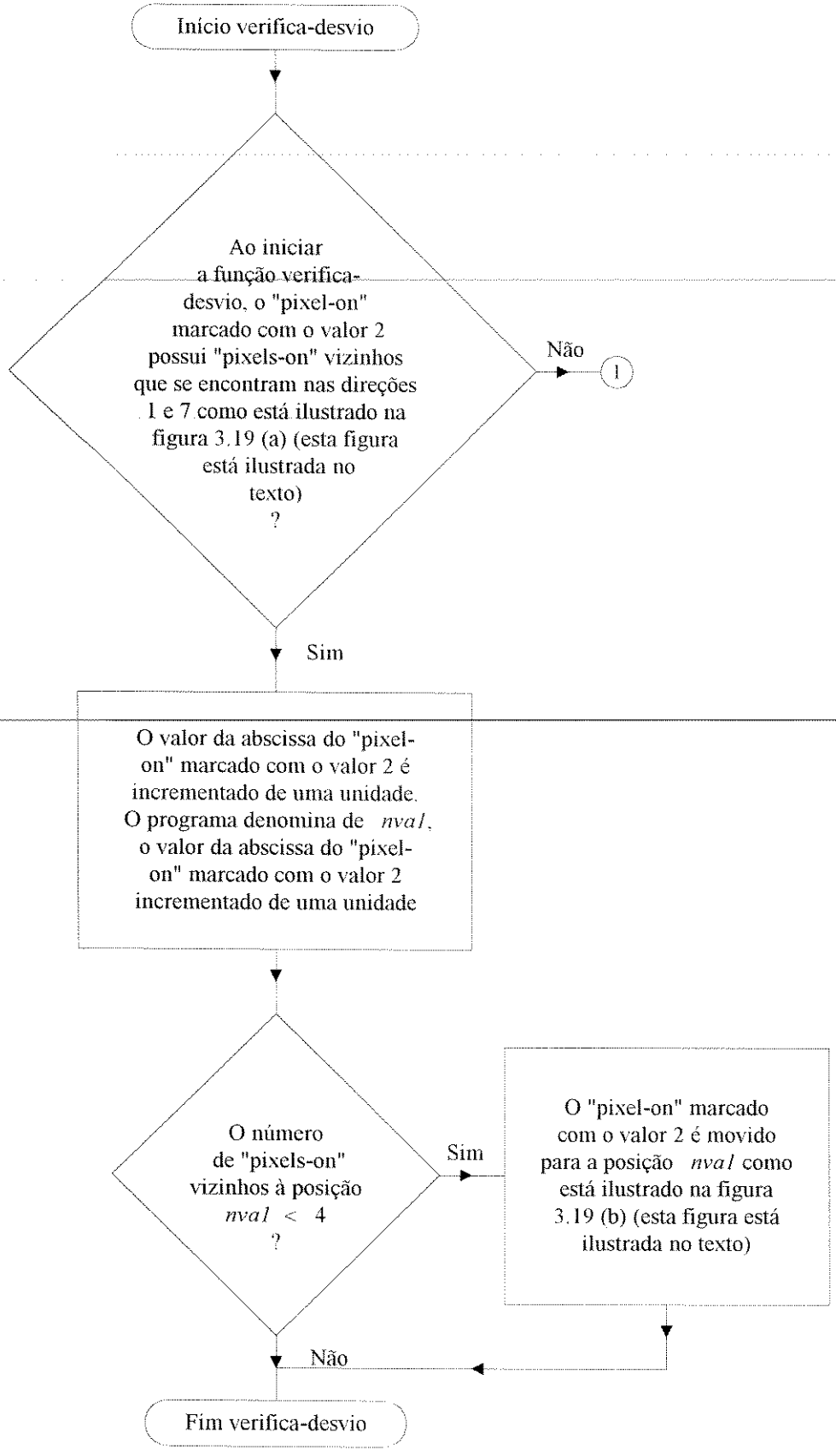


Figura 3.18(a) - Primeira parte do fluxograma da função verifica-desvio.

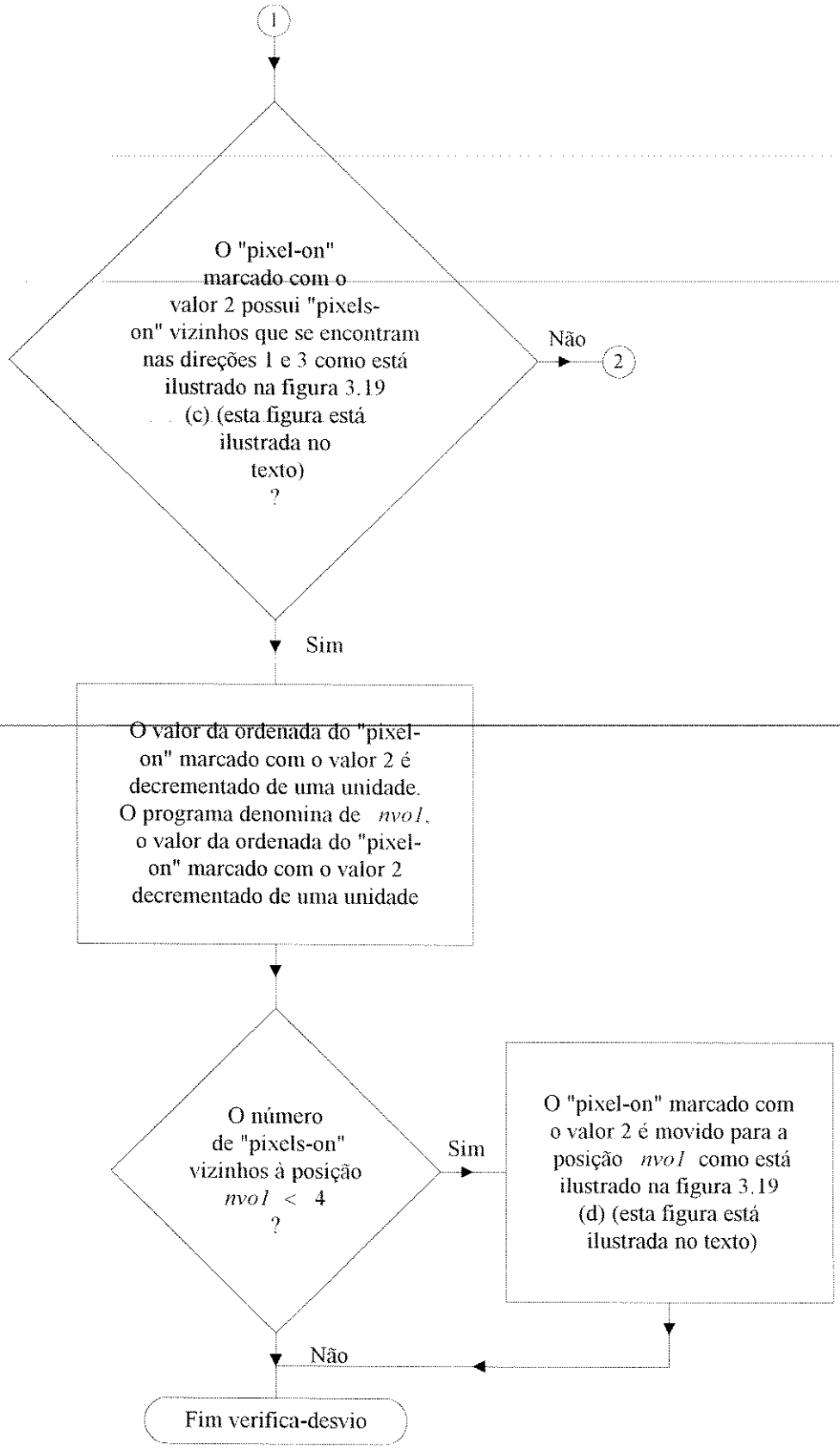


Figura 3.18(b) - Segunda parte do fluxograma da função verifica-desvio.

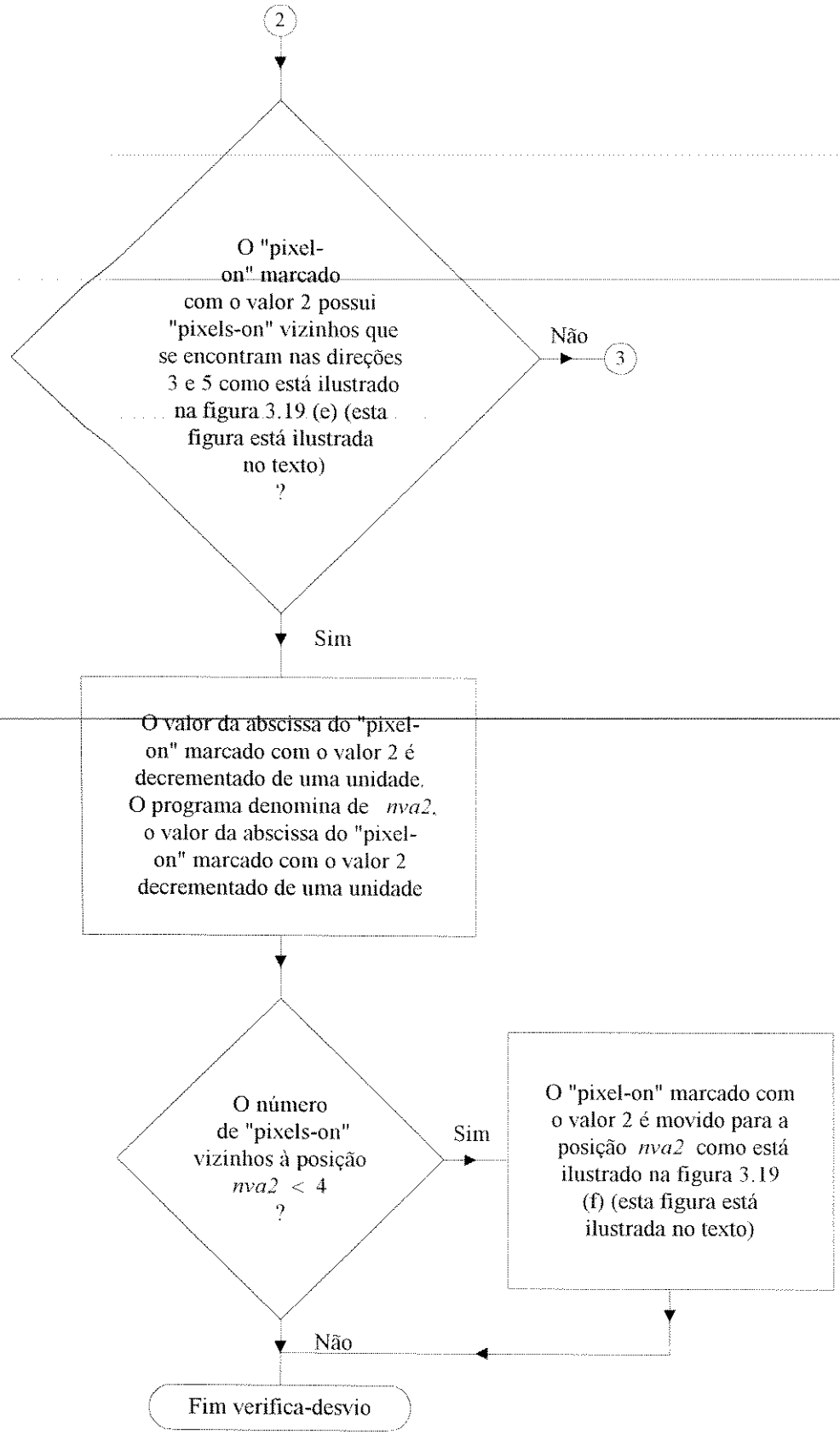


Figura 3.18(c) - Terceira parte do fluxograma da função verifica-desvio.

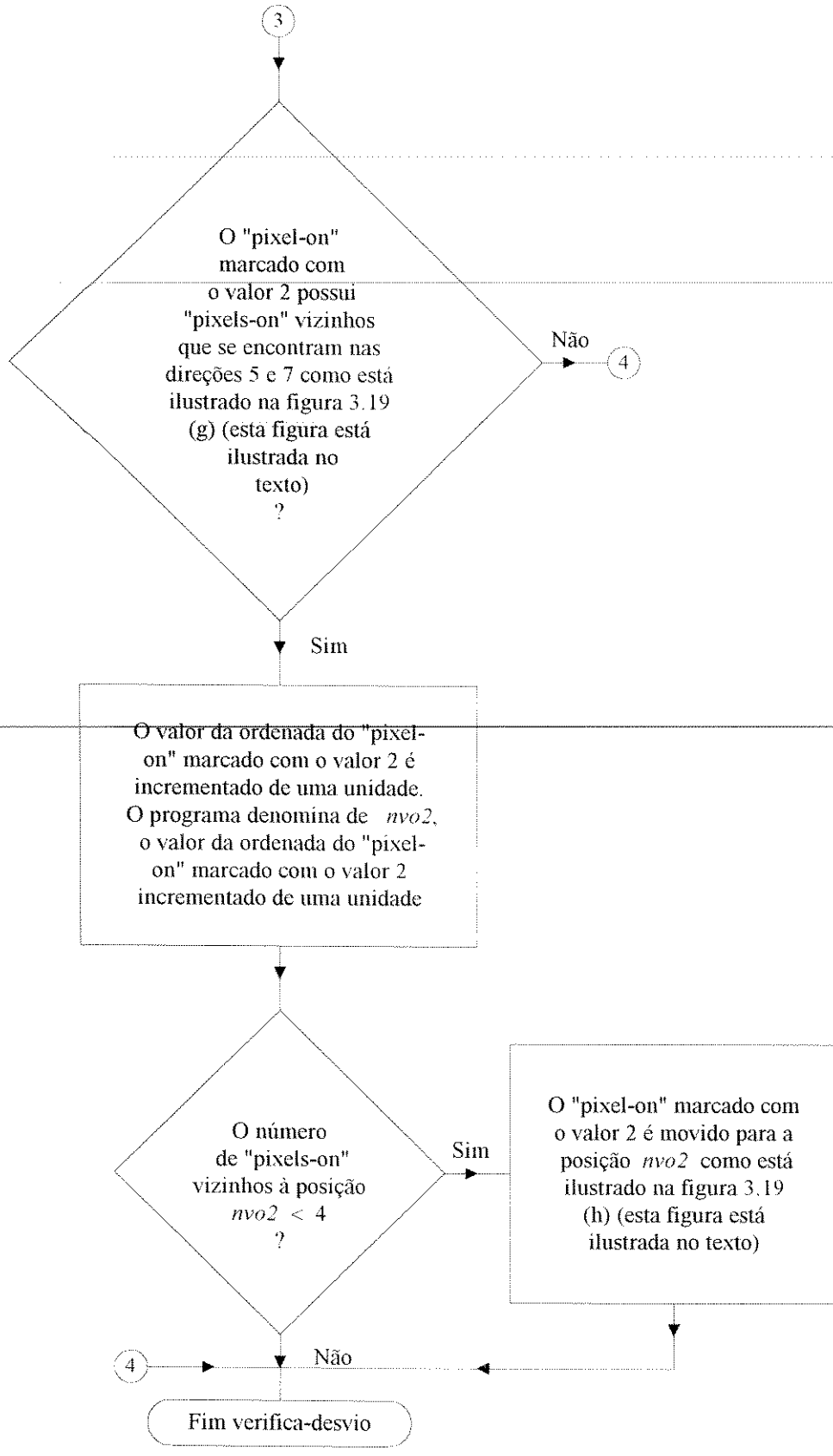


Figura 3.18(d) - Quarta parte do fluxograma da função verifica-desvio.

FLUXOGRAMAS DO CAPÍTULO 4

Fluxograma que realiza a delimitação das coordenadas inicial e final da linha horizontal superior, quando a matriz-original é varrida da esquerda para a direita

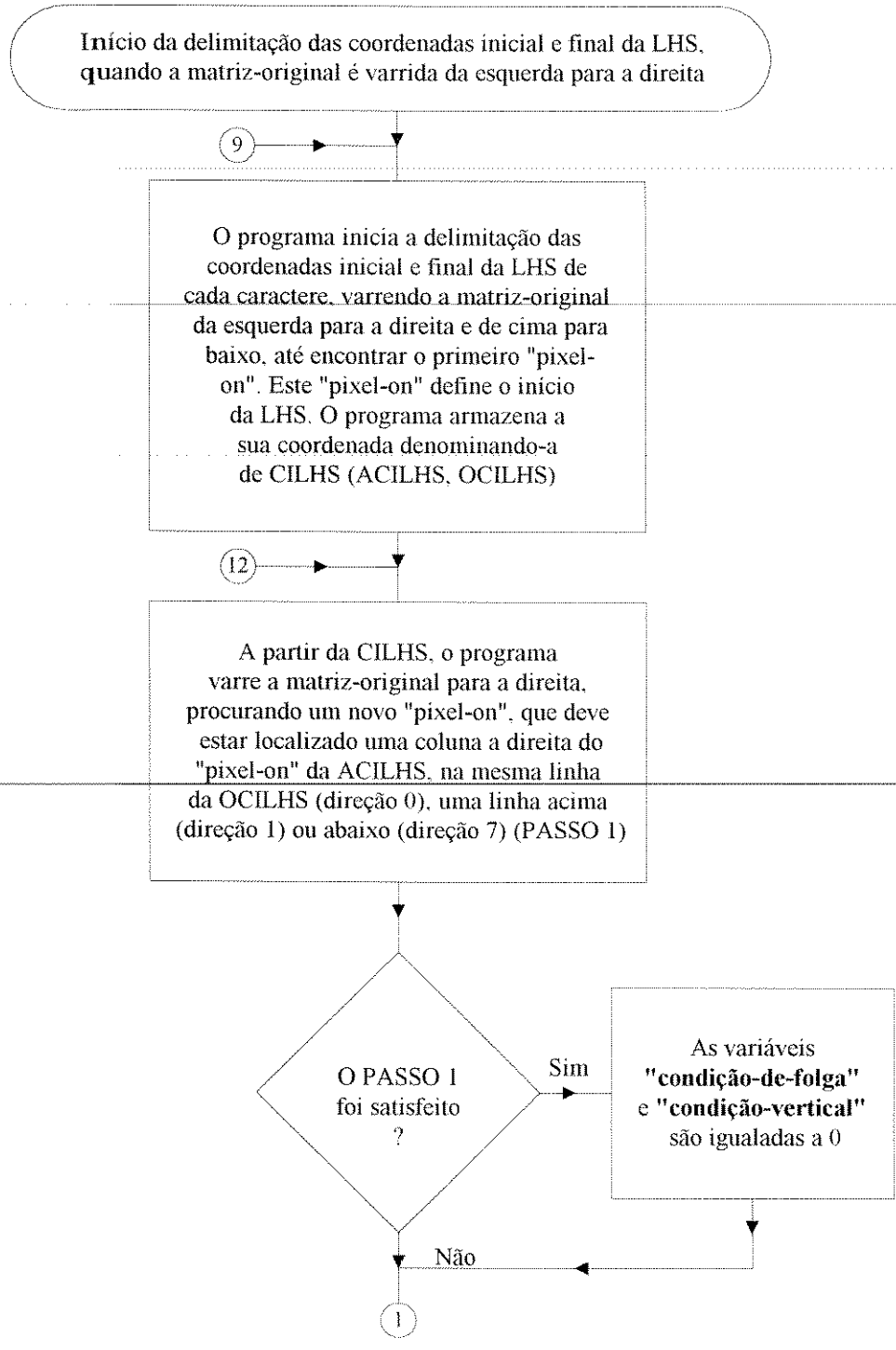


Figura 4.1(a) - Primeira parte do fluxograma que realiza a delimitação das coordenadas inicial e final da linha horizontal superior, quando a matriz-original é varrida da esquerda para a direita.

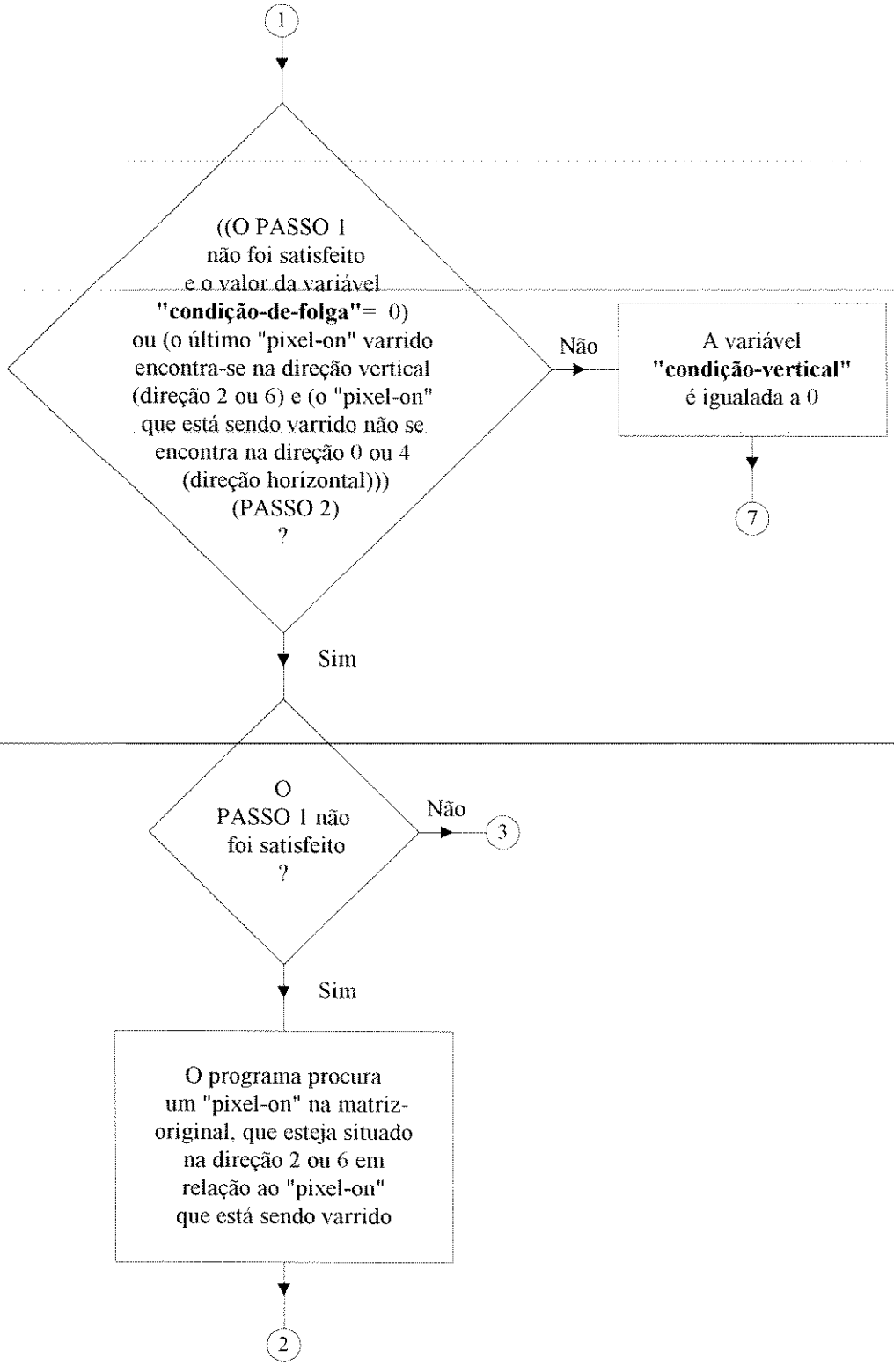


Figura 4.1(b) - Segunda parte do fluxograma que realiza a delimitação das coordenadas inicial e final da linha horizontal superior, quando a matriz-original é varrida da esquerda para a direita.

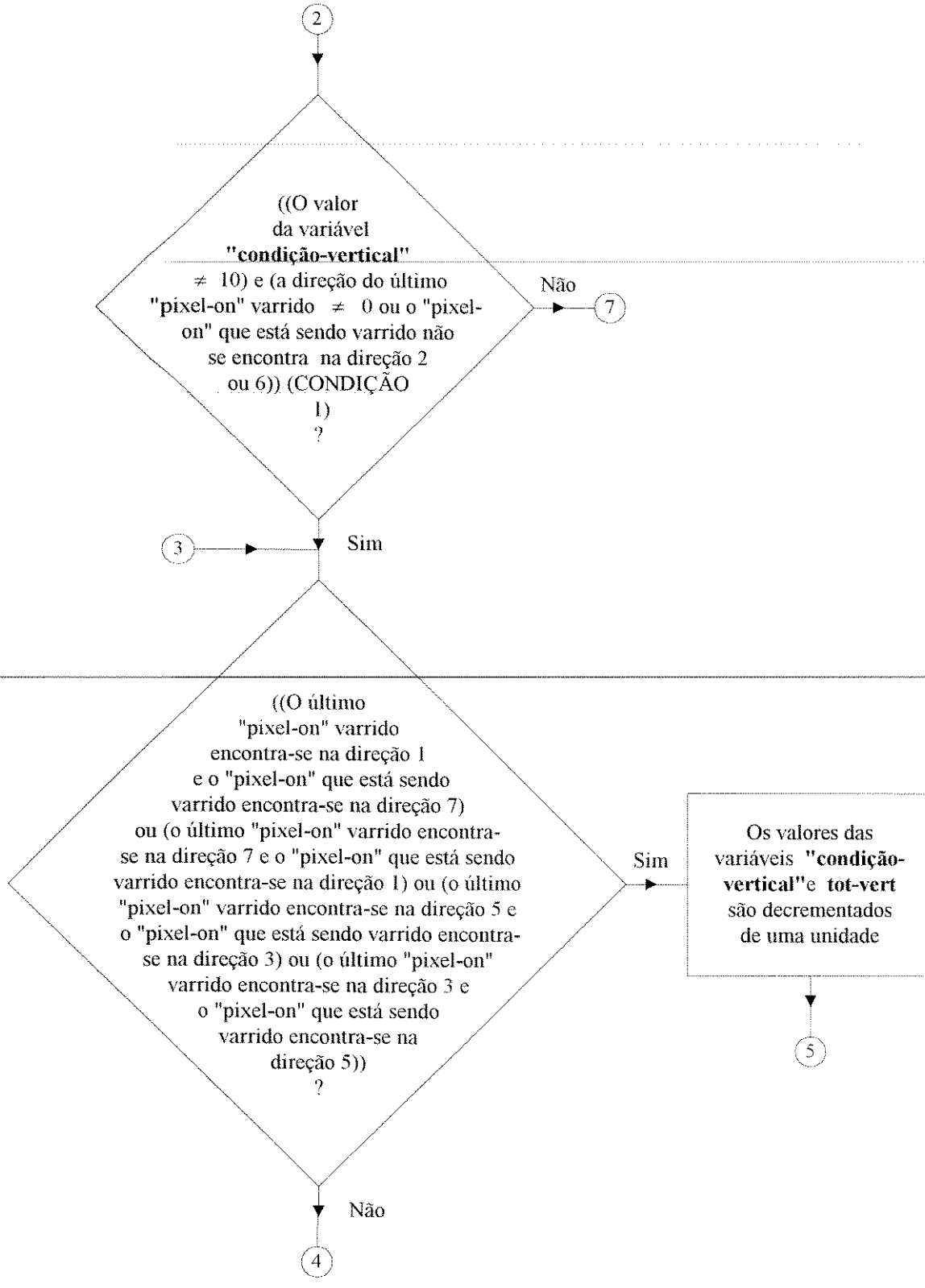


Figura 4.1(c) - Terceira parte do fluxograma que realiza a delimitação das coordenadas inicial e final da linha horizontal superior, quando a matriz-original é varrida da esquerda para a direita.

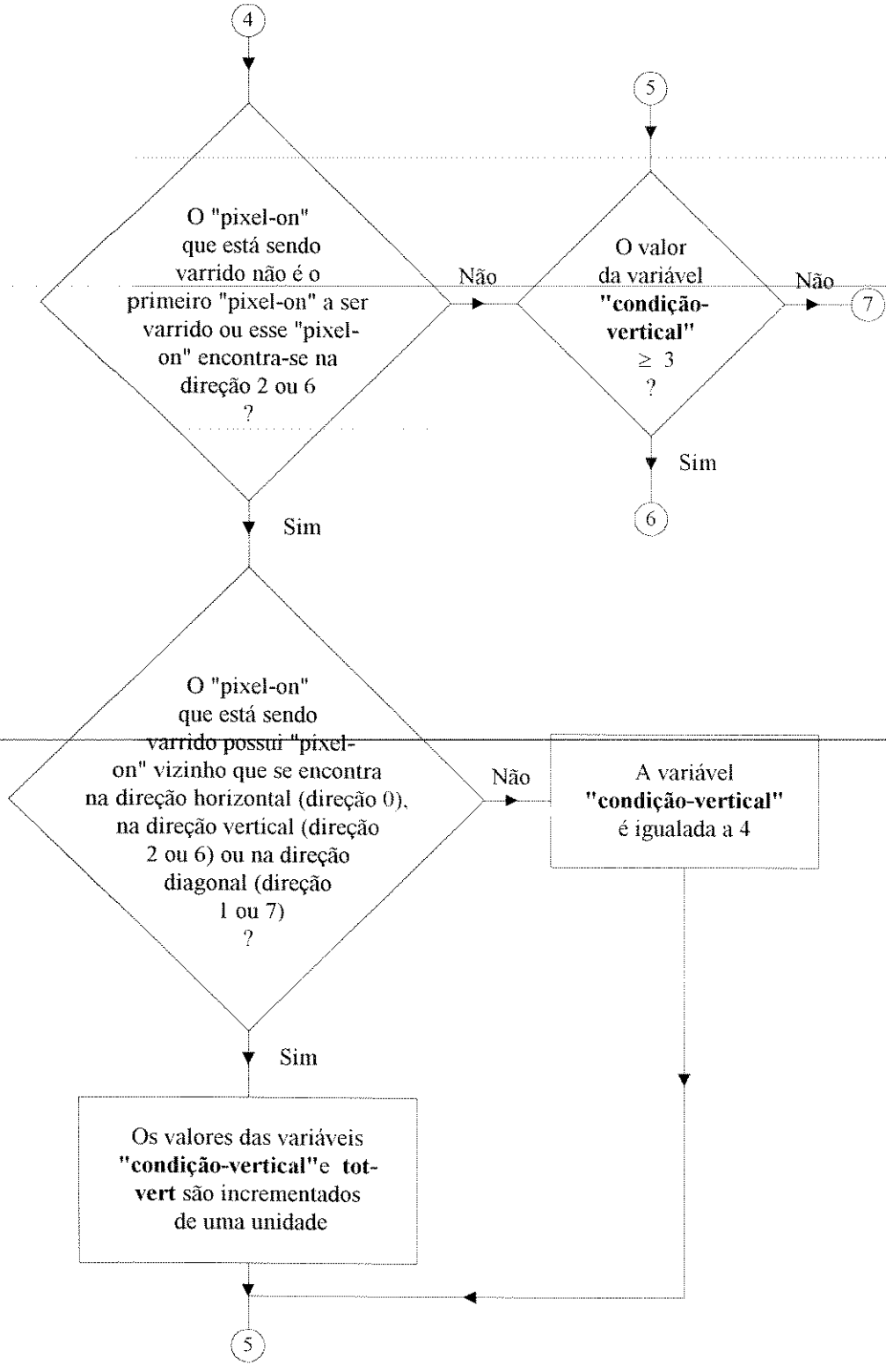


Figura 4.1(d) - Quarta parte do fluxograma que realiza a delimitação das coordenadas inicial e final da linha horizontal superior, quando a matriz-original é varrida da esquerda para a direita.

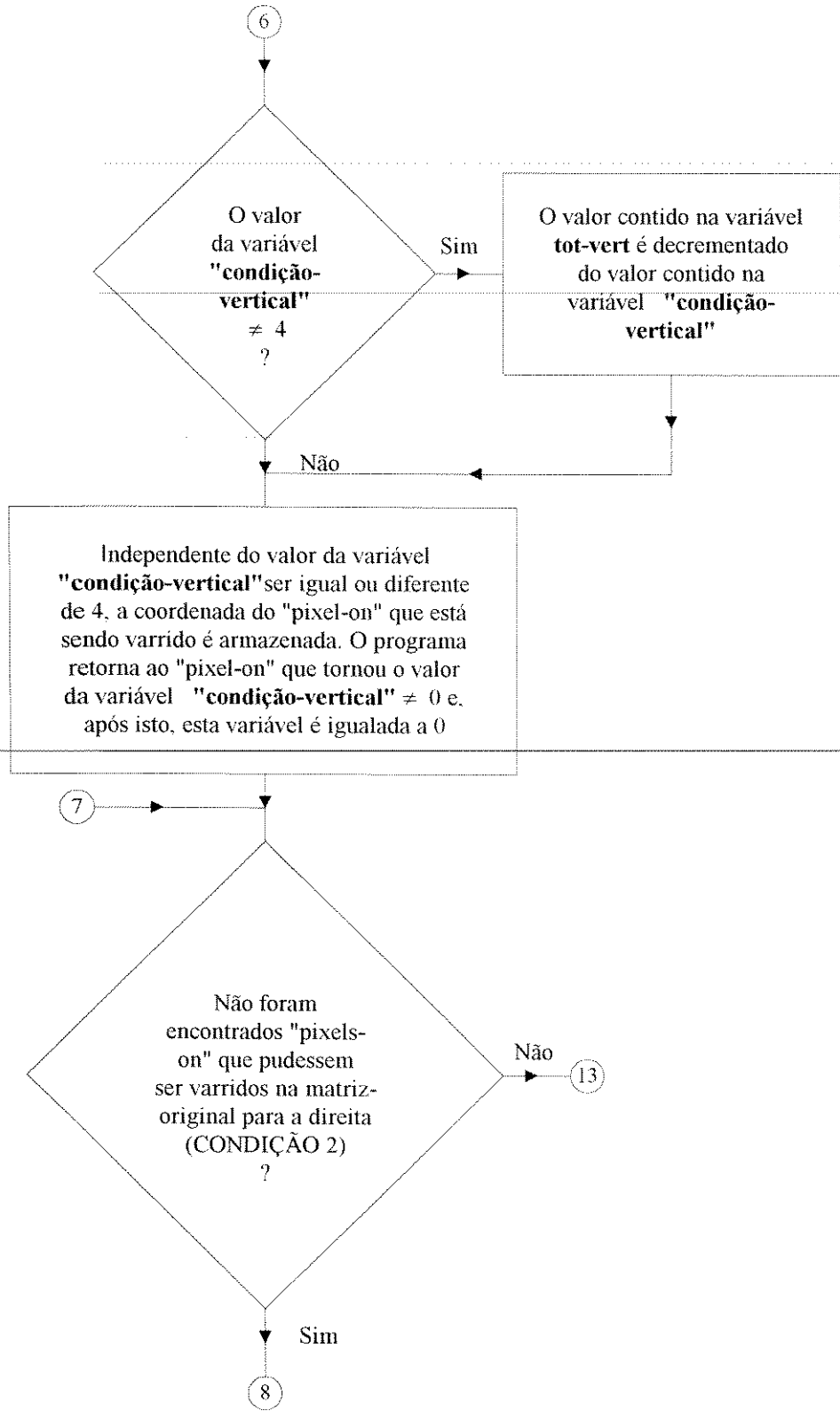


Figura 4.1(e) - Sexta parte do fluxograma que realiza a delimitação das coordenadas inicial e final da linha horizontal superior, quando a matriz-original é varrida da esquerda para a direita.

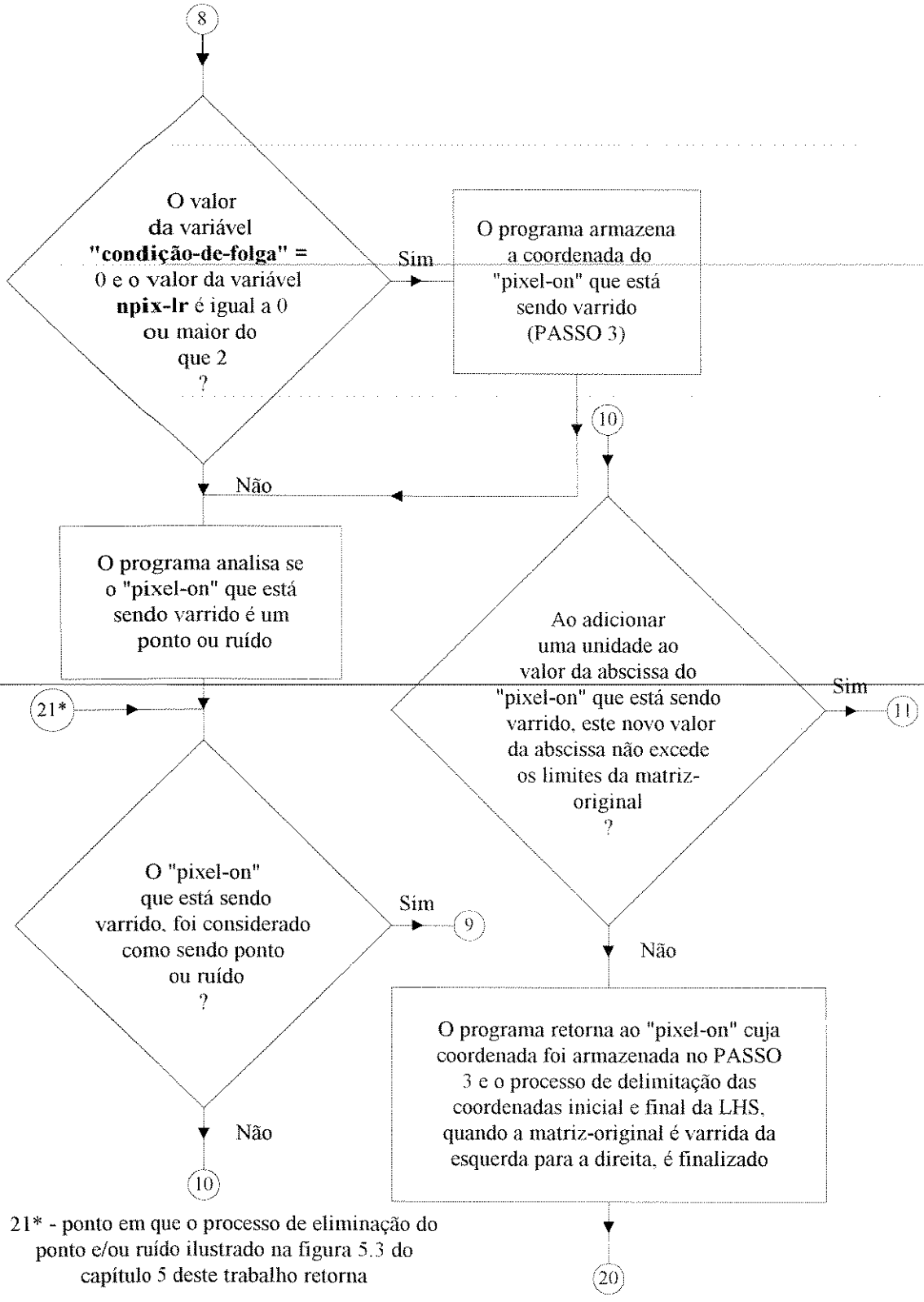


Figura 4.1(f) - Sétima parte do fluxograma que realiza a delimitação das coordenadas inicial e final da linha horizontal superior, quando a matriz-original é varrida da esquerda para a direita.

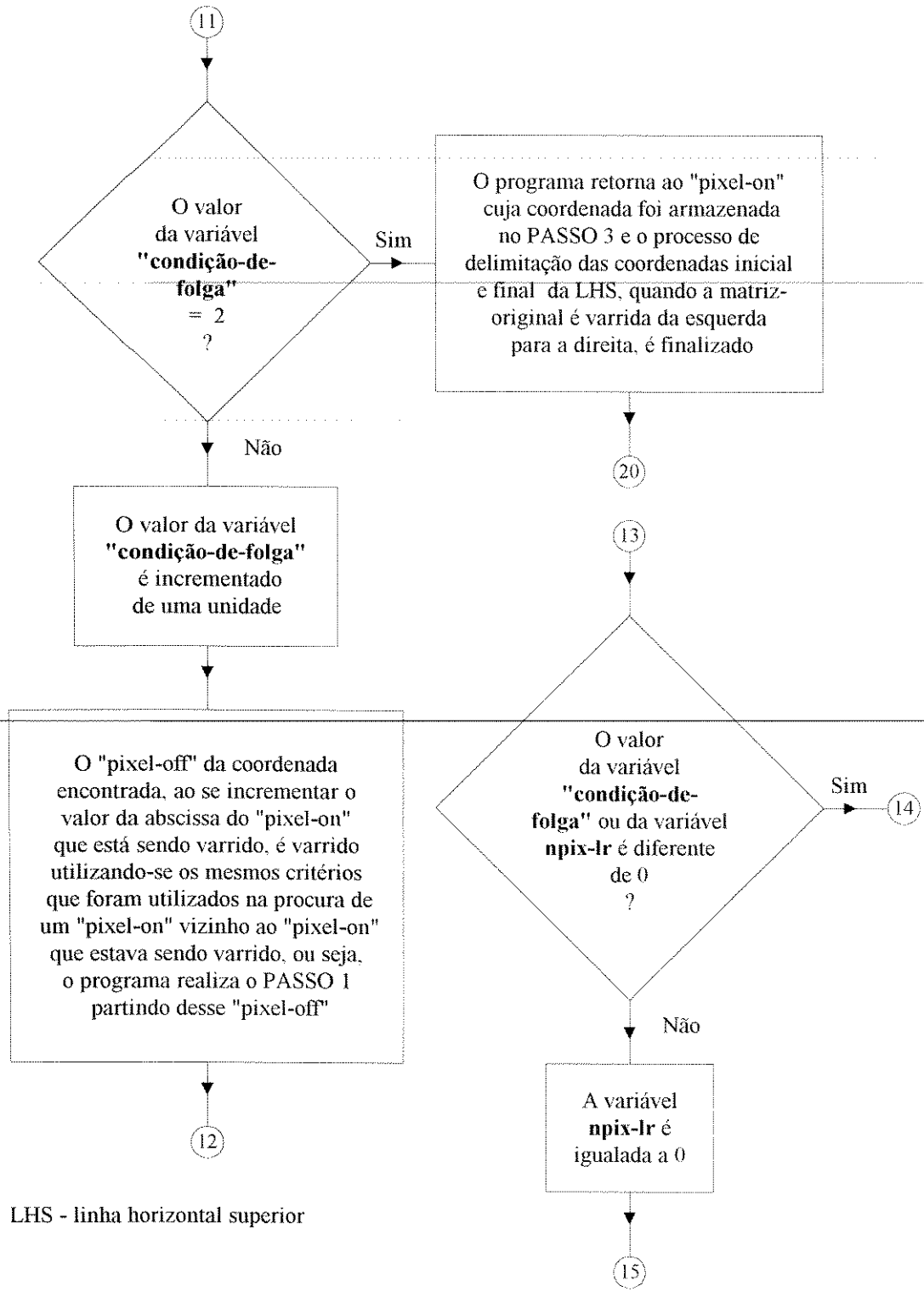


Figura 4.1(g) - Oitava parte do fluxograma que realiza a delimitação das coordenadas inicial e final da linha horizontal superior, quando a matriz-original é varrida da esquerda para a direita.

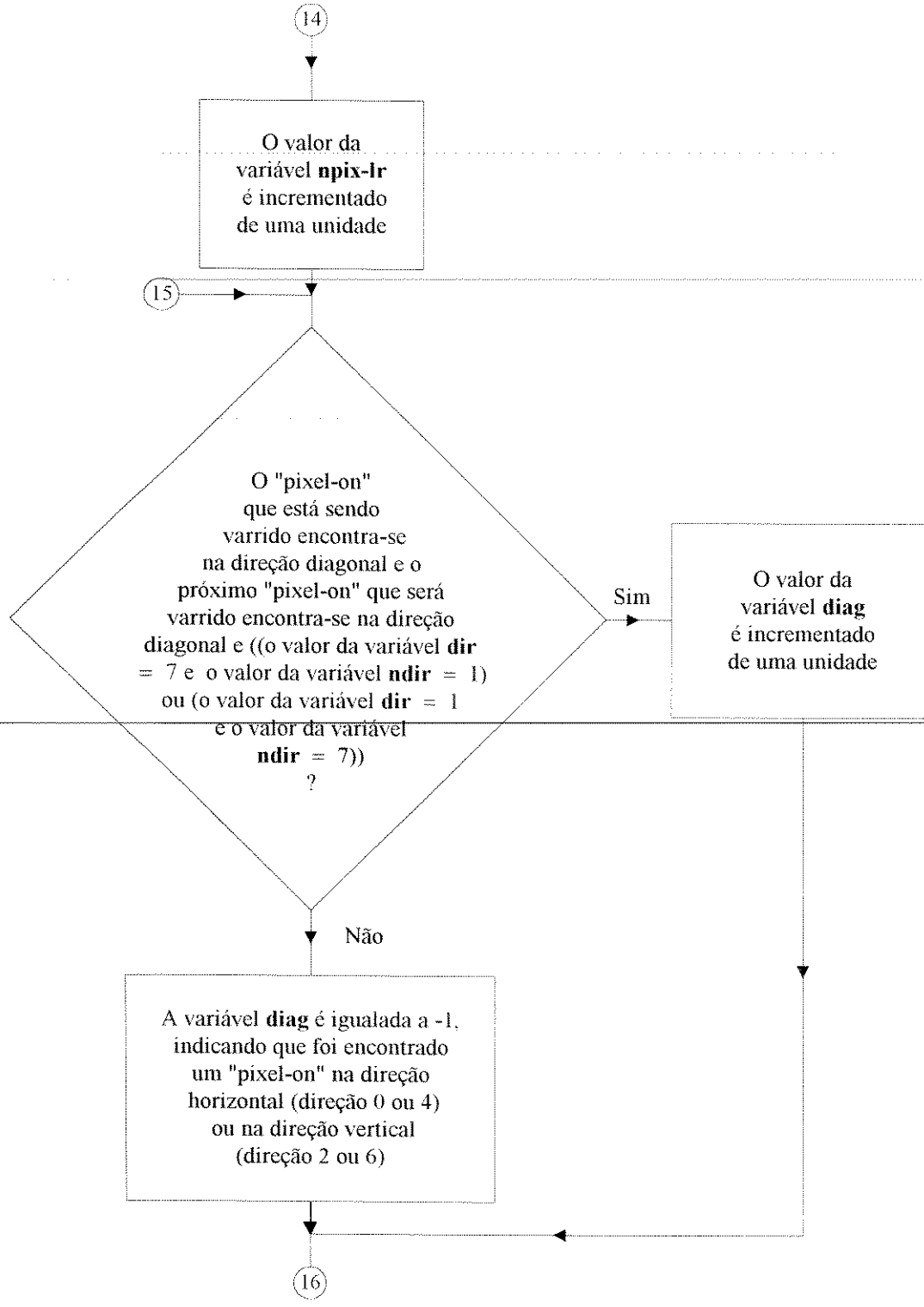
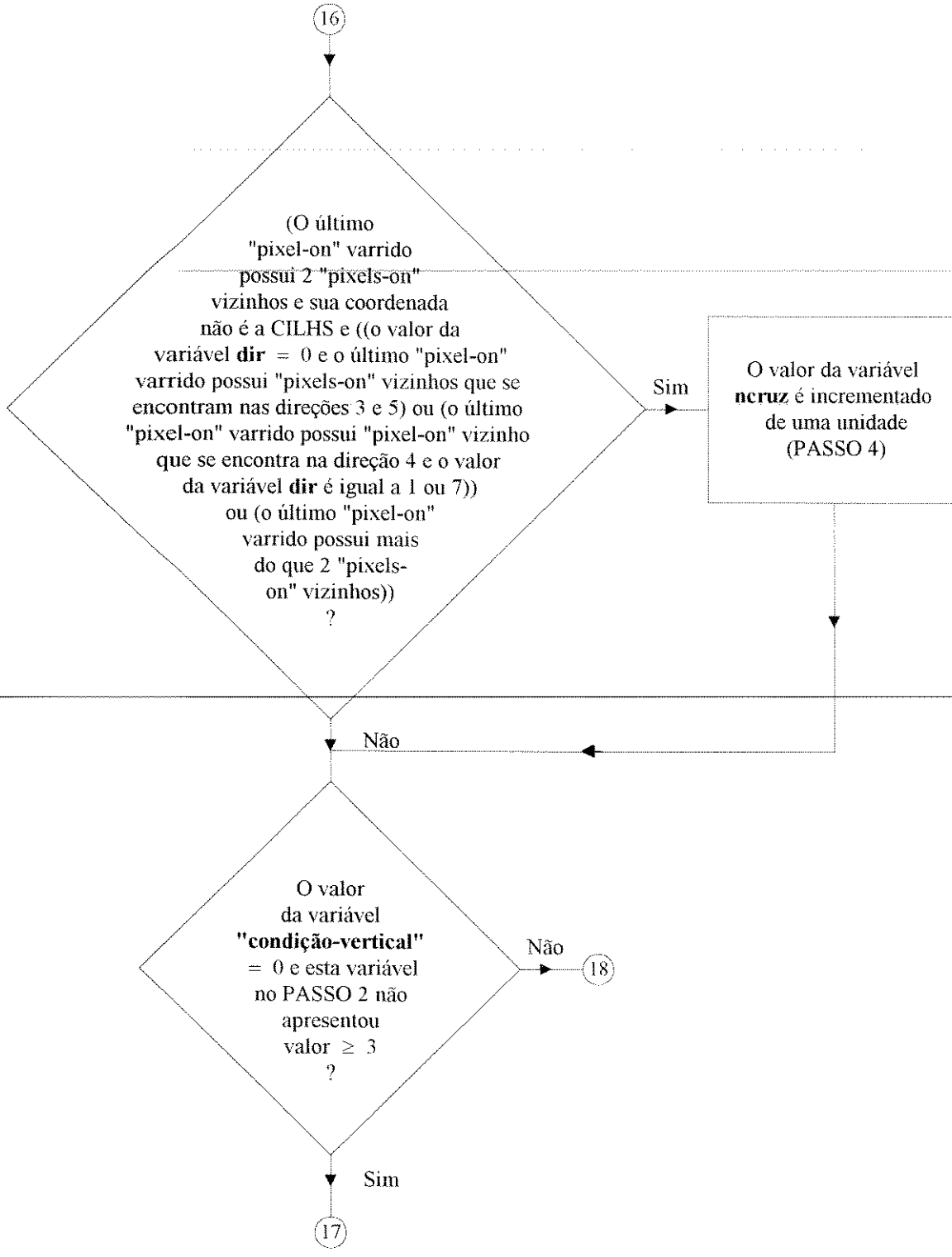
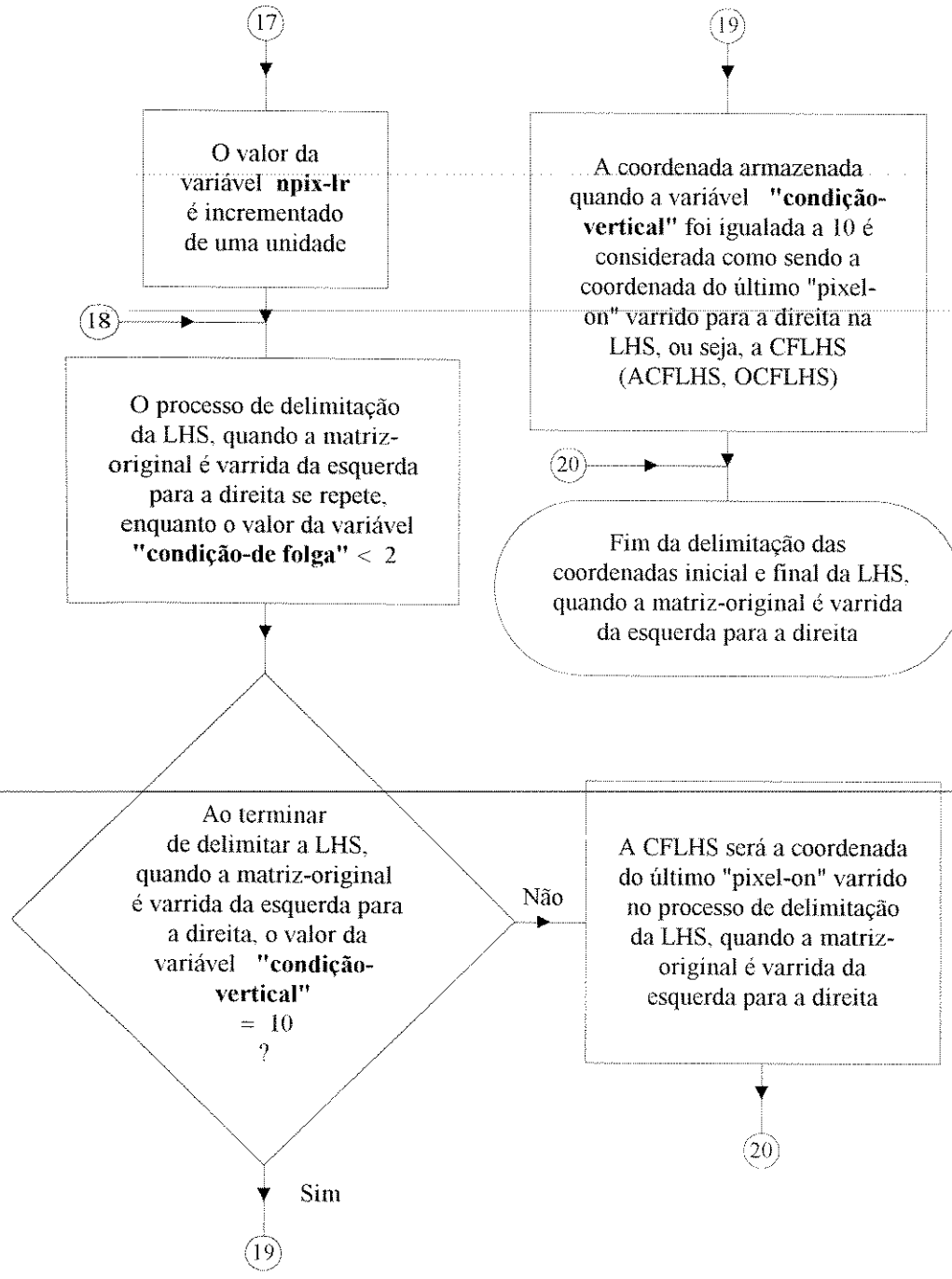


Figura 4.1(h) - Nona parte do fluxograma que realiza a delimitação das coordenadas inicial e final da linha horizontal superior, quando a matriz-original é varrida da esquerda para a direita.



LHS - linha horizontal superior
CILHS - coordenada inicial da LHS

Figura 4.1(i) - Décima parte do fluxograma que realiza a delimitação das coordenadas inicial e final da linha horizontal superior, quando a matriz-original é varrida da esquerda para a direita.



LHS - linha horizontal superior
CFLHS - coordenada final da LHS
ACFLHS - abscissa da coordenada final da LHS
OCFLHS - ordenada da coordenada final da LHS

Figura 4.1(j) - Décima primeira parte do fluxograma que realiza a delimitação das coordenadas inicial e final da linha horizontal superior, quando a matriz-original é varrida da esquerda para a direita

Fluxograma que realiza a delimitação das coordenadas inicial e final da linha horizontal superior, quando a matriz-original é varrida da direita para a esquerda

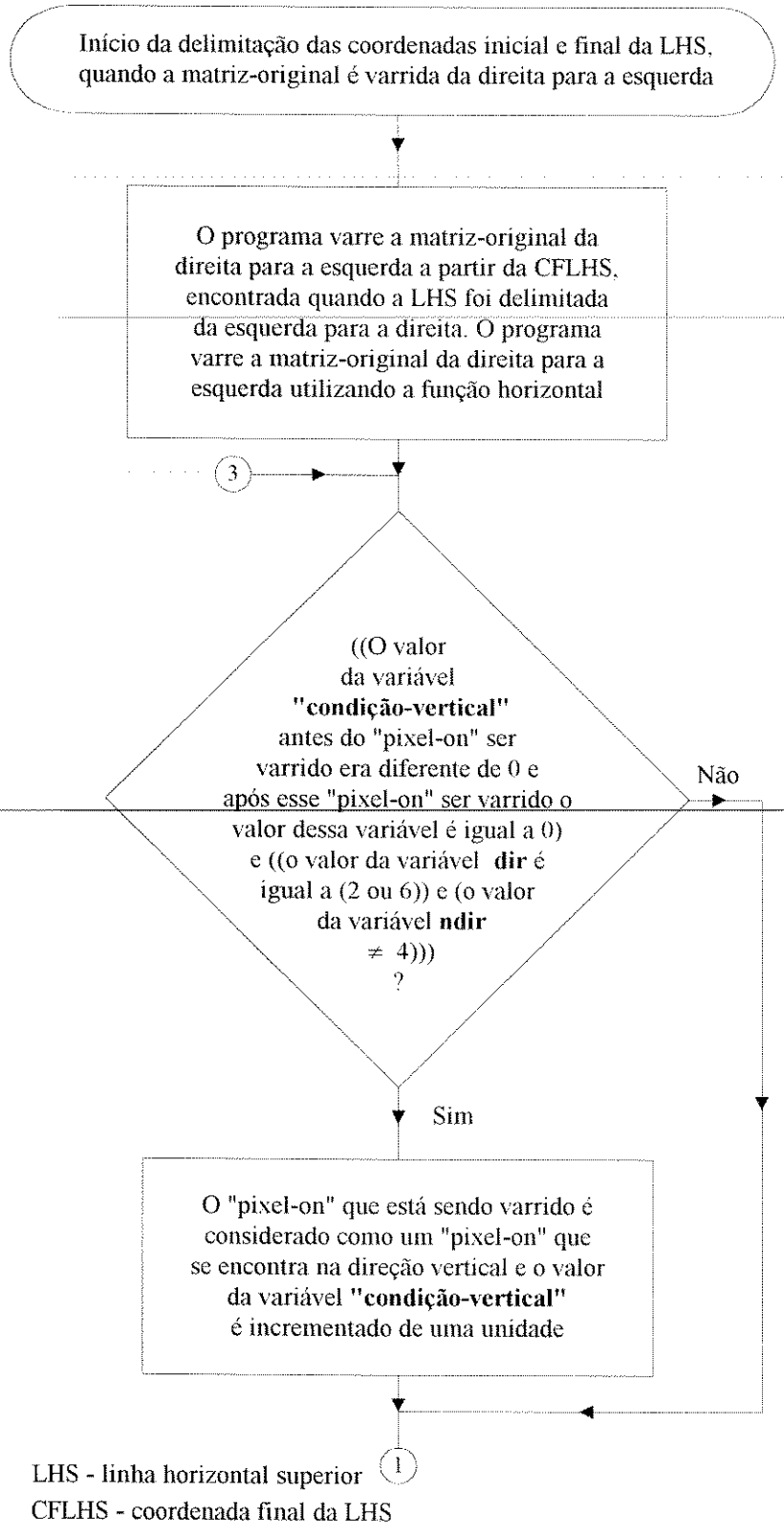


Figura 4.3(a) - Primeira parte do fluxograma que realiza a delimitação das coordenadas inicial e final da linha horizontal superior, quando a matriz-original é varrida da direita para a esquerda.

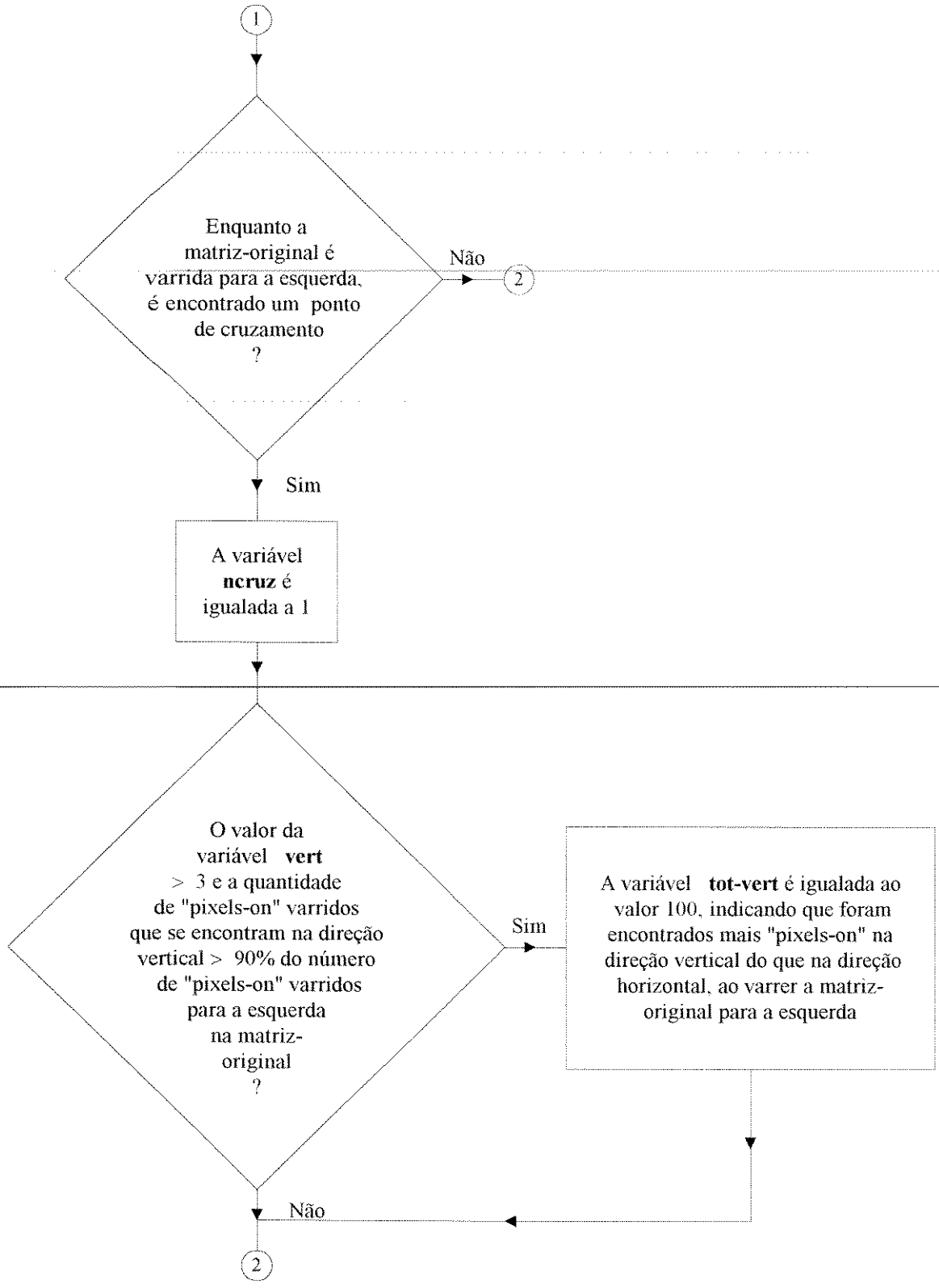


Figura 4.3(b) - Segunda parte do fluxograma que realiza a delimitação das coordenadas inicial e final da linha horizontal superior, quando a matriz-original é varrida da direita para a esquerda.

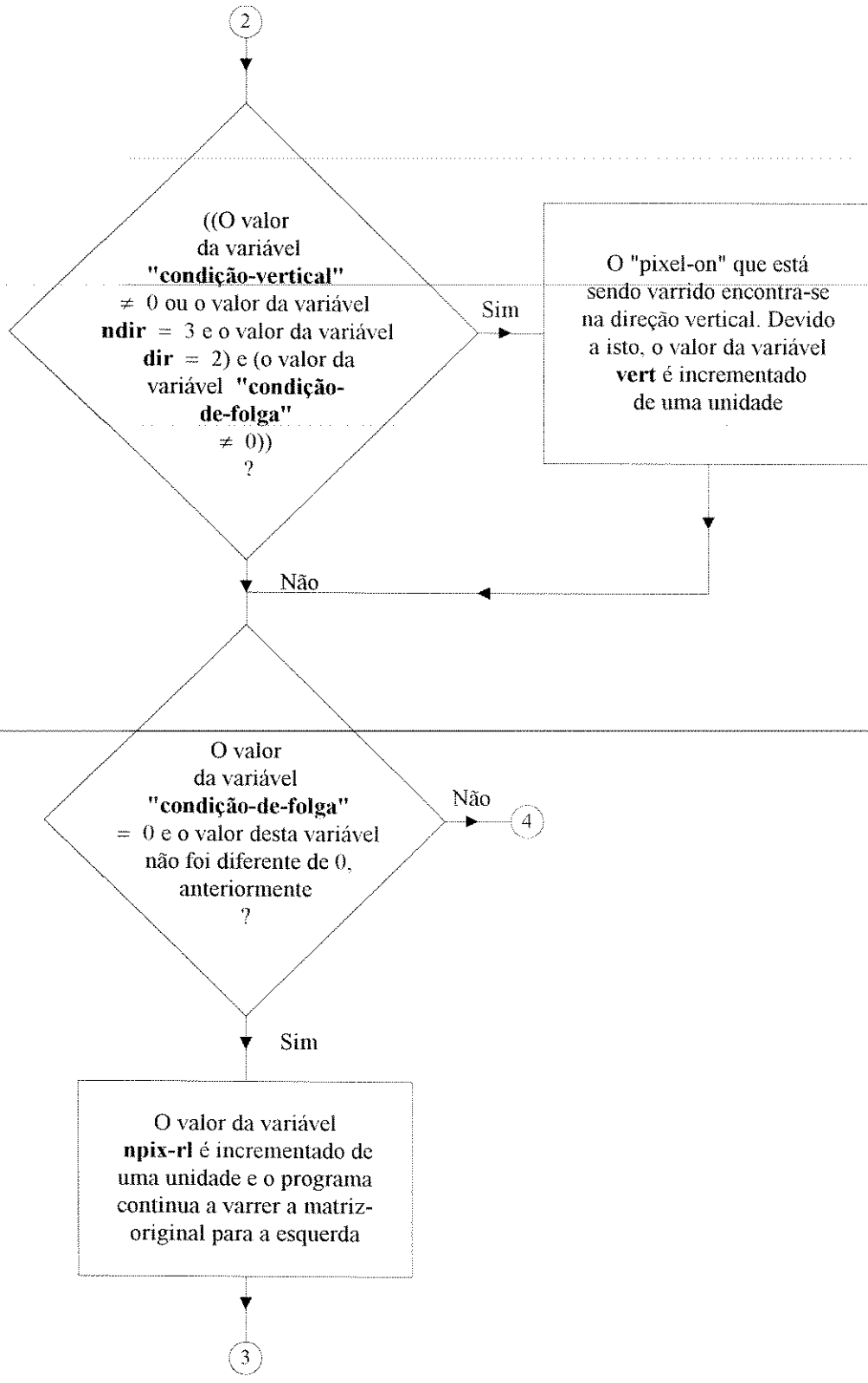


Figura 4.3(c) - Terceira parte do fluxograma que realiza a delimitação das coordenadas inicial e final da linha horizontal superior, quando a matriz-original é varrida da direita para a esquerda.

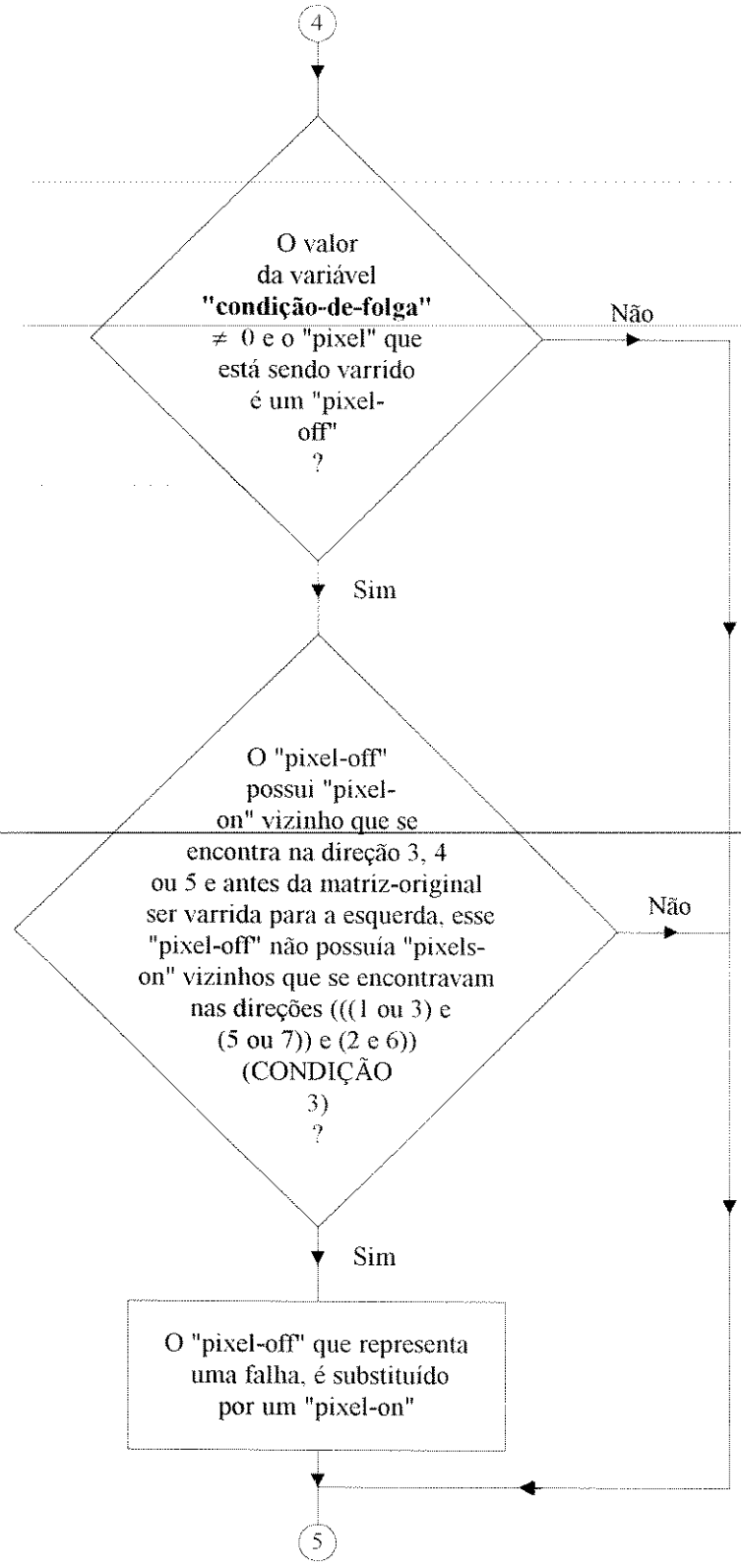


Figura 4.3(d) - Quarta parte do fluxograma que realiza a delimitação das coordenadas inicial e final da linha horizontal superior, quando a matriz-original é varrida da direita para a esquerda.

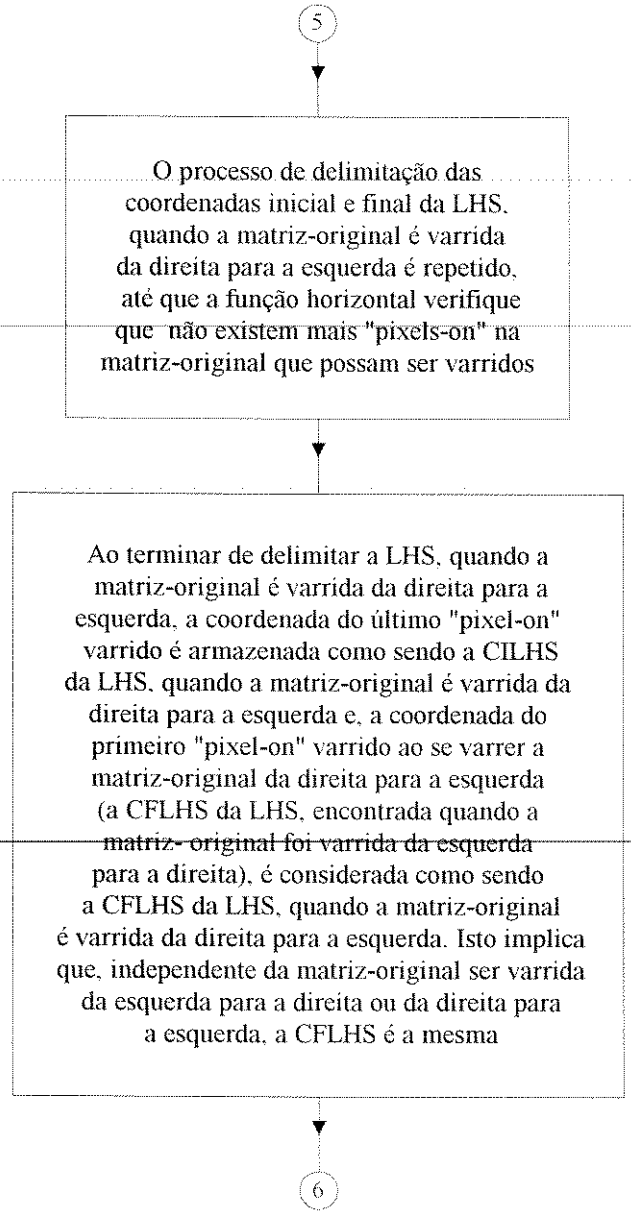


Figura 4.3(e) - Quinta parte do fluxograma que realiza a delimitação das coordenadas inicial e final da linha horizontal superior, quando a matriz-original é varrida da direita para a esquerda.

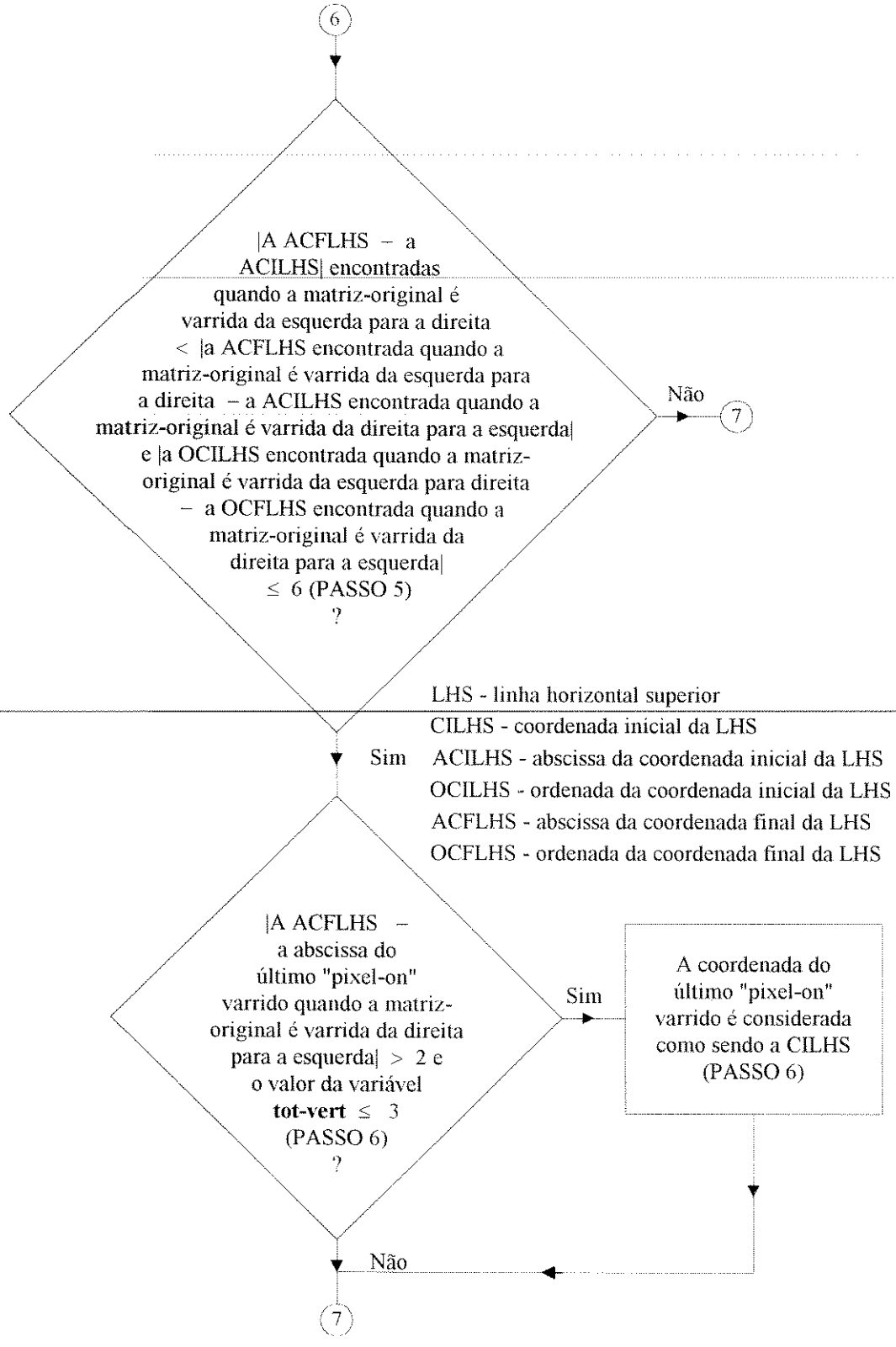


Figura 4.3(f) - Sexta parte do fluxograma que realiza a delimitação das coordenadas inicial e final da linha horizontal superior, quando a matriz-original é varrida da direita para a esquerda.

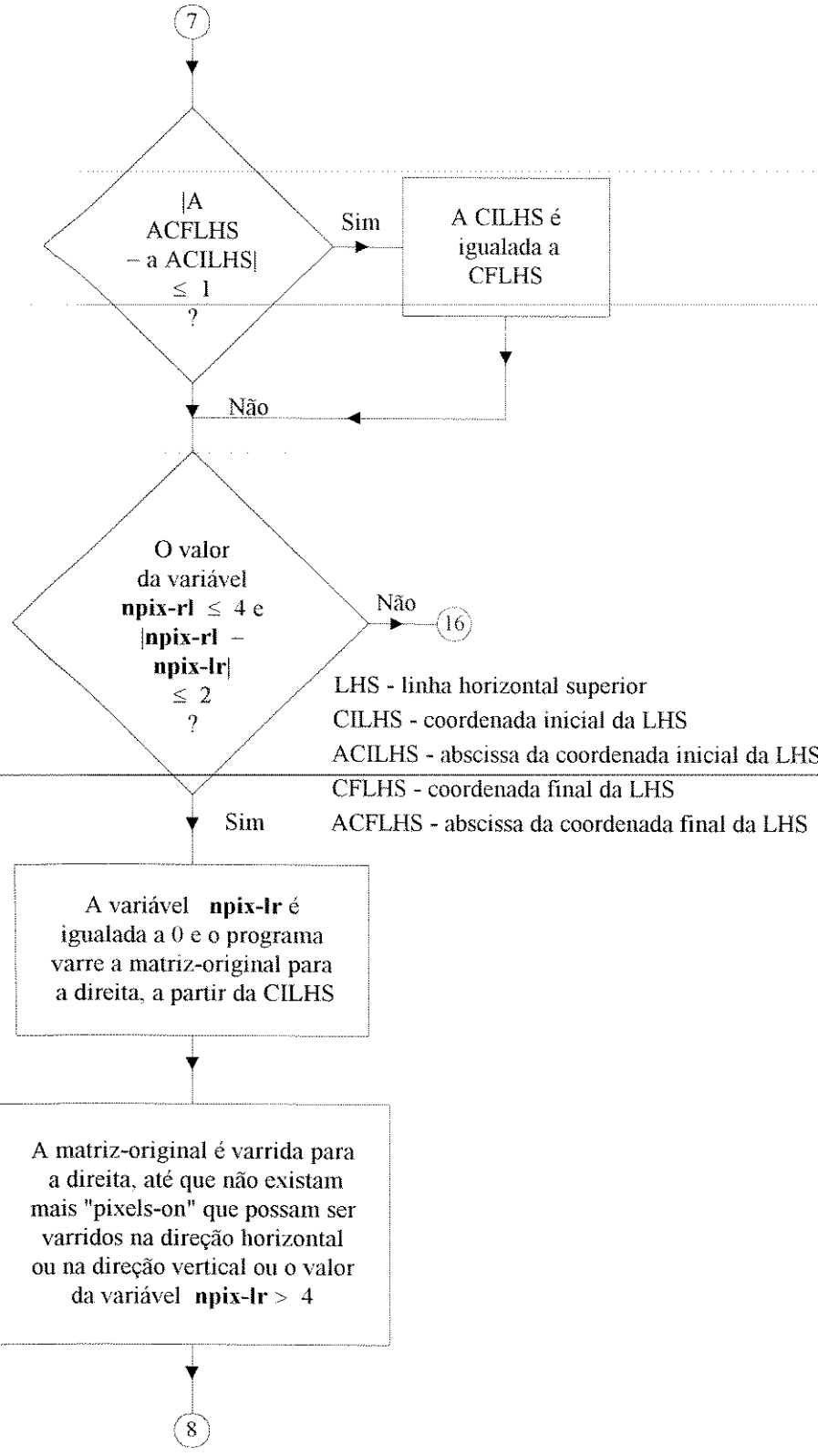


Figura 4.3(g) - Sétima parte do fluxograma que realiza a delimitação das coordenadas inicial e final da linha horizontal superior, quando a matriz-original é varrida da direita para a esquerda.

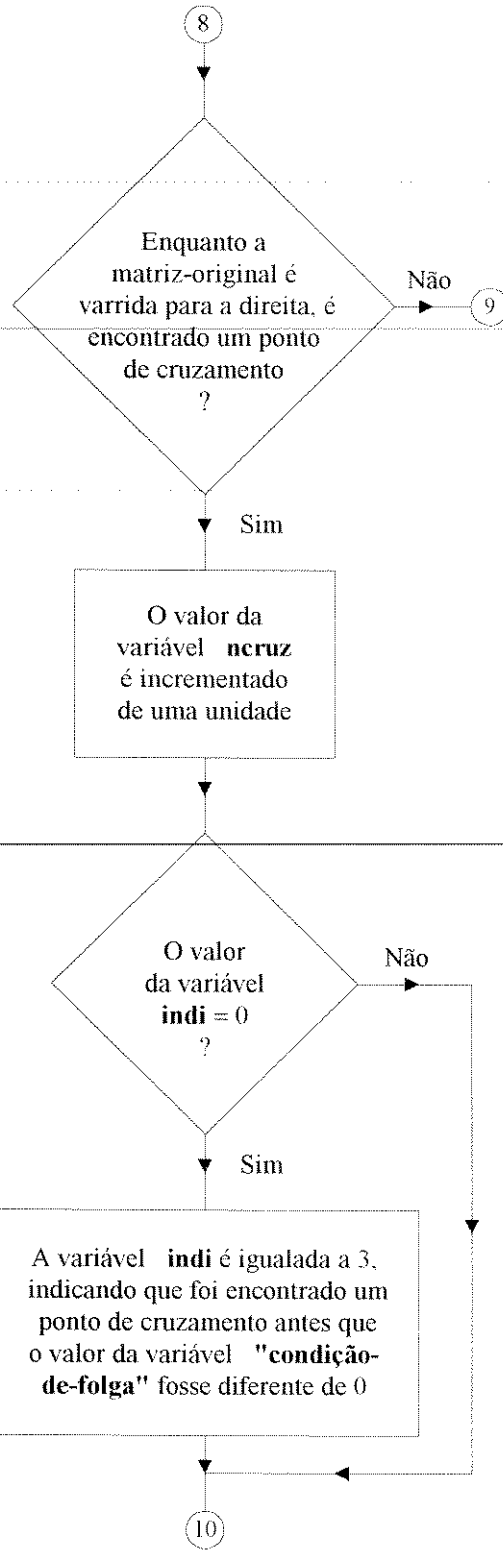


Figura 4.3(h) - Oitava parte do fluxograma que realiza a delimitação das coordenadas inicial e final da linha horizontal superior, quando a matriz-original é varrida da direita para a esquerda.

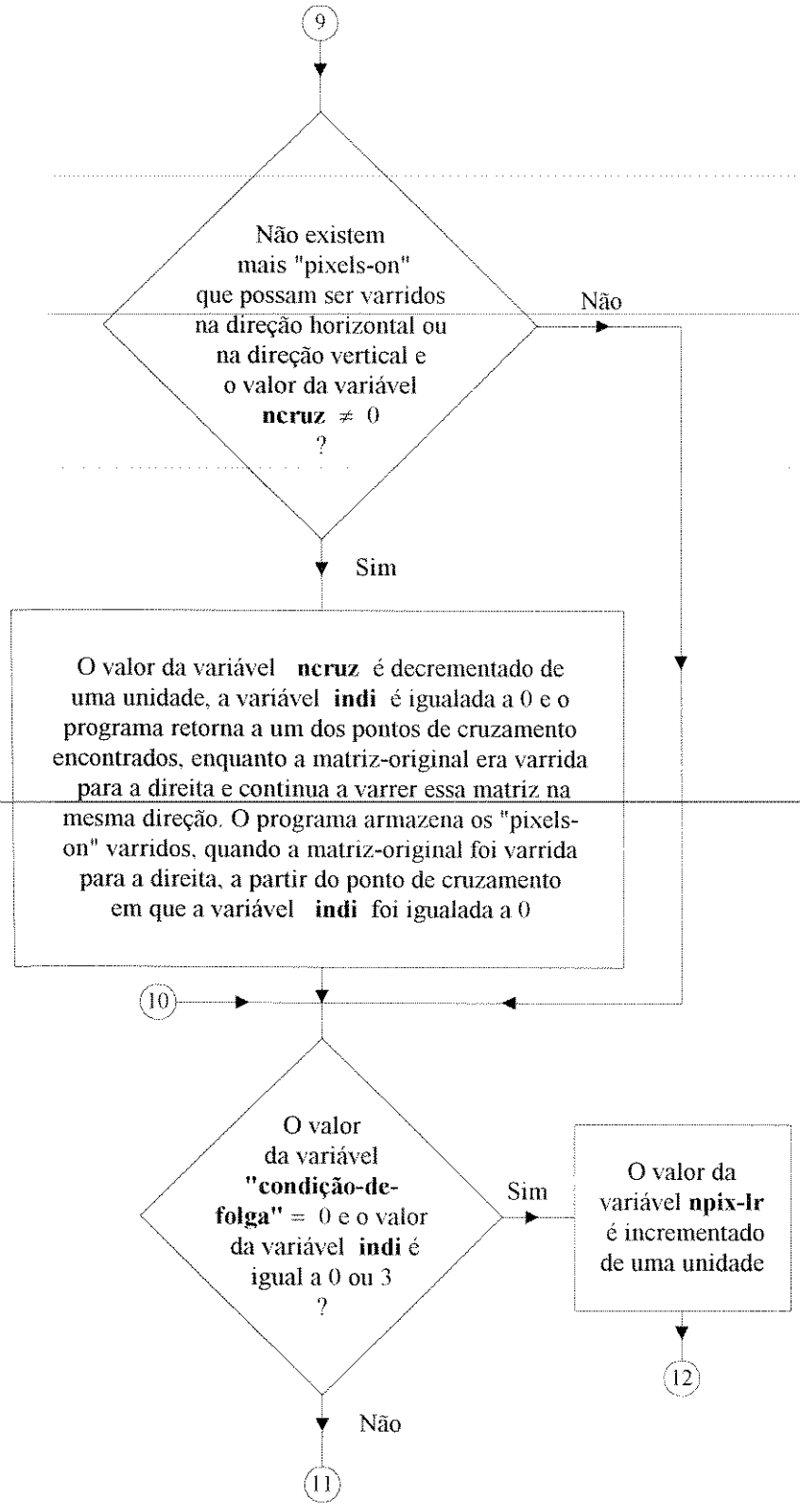


Figura 4.3(i) - Nona parte do fluxograma que realiza a delimitação das coordenadas inicial e final da linha horizontal superior, quando a matriz-original é varrida da direita para a esquerda.

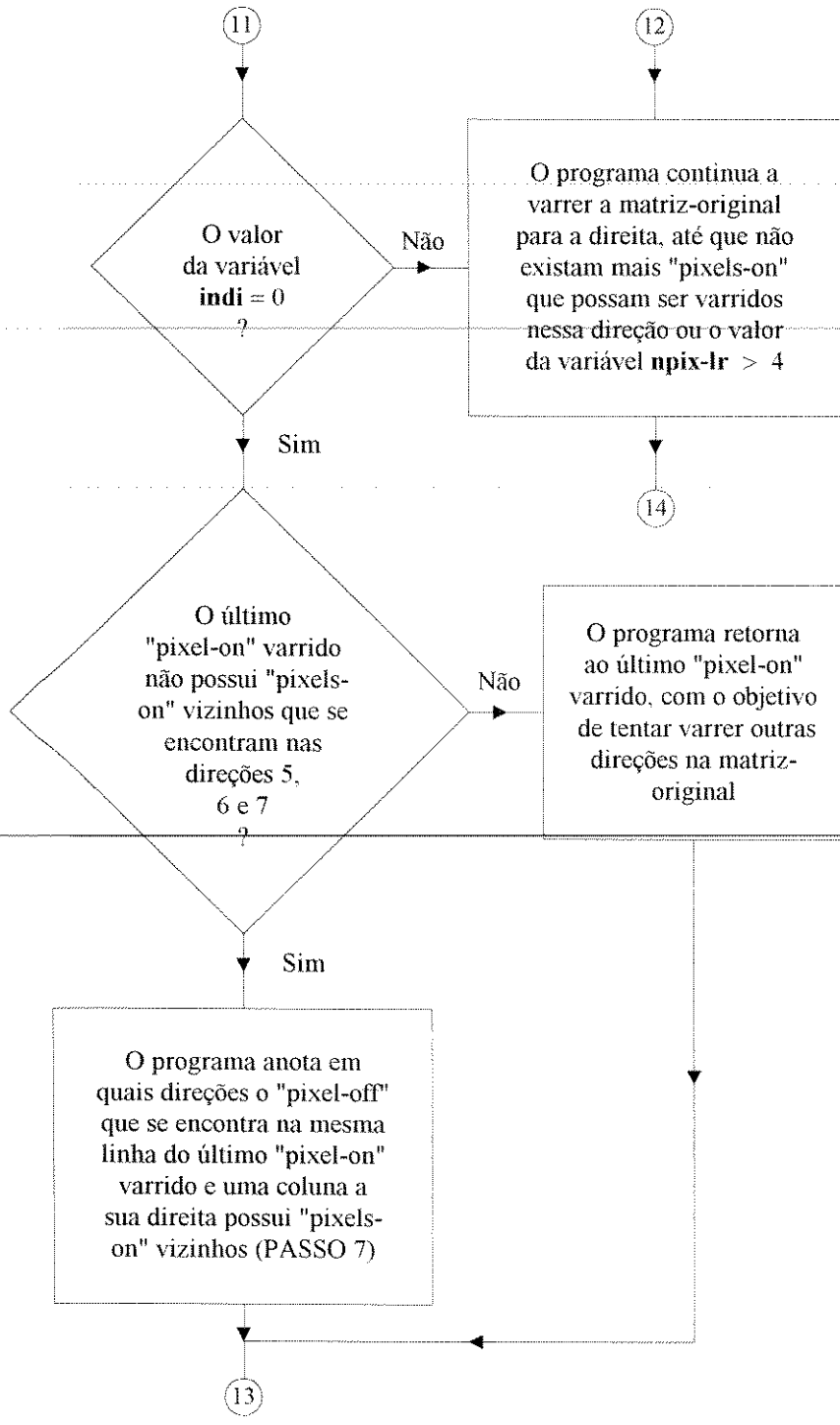


Figura 4.3(j) - Décima parte do fluxograma que realiza a delimitação das coordenadas inicial e final da linha horizontal superior, quando a matriz-original é varrida da direita para a esquerda.

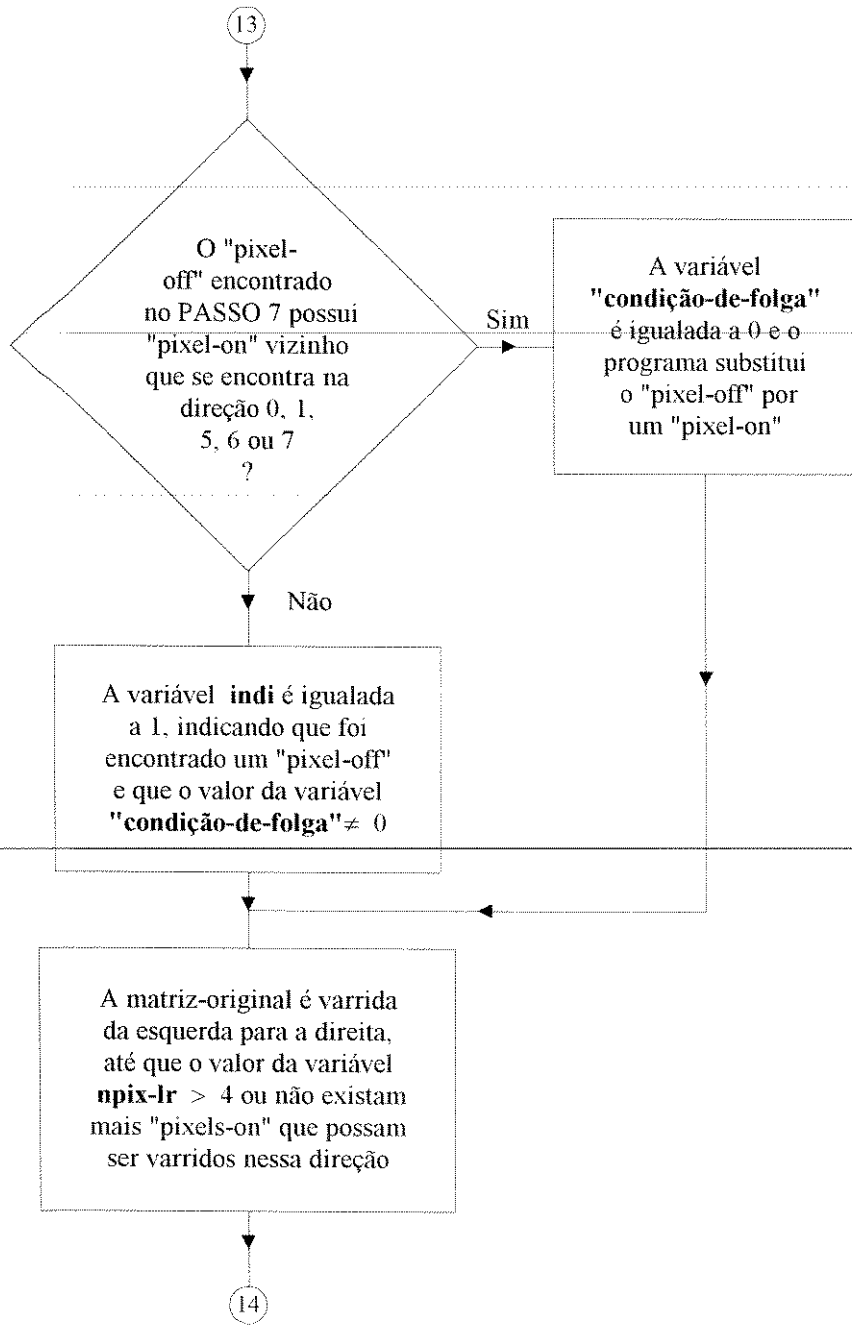


Figura 4.3(k) - Décima primeira parte do fluxograma que realiza a delimitação das coordenadas inicial e final da linha horizontal superior, quando a matriz-original é varrida da direita para a esquerda.

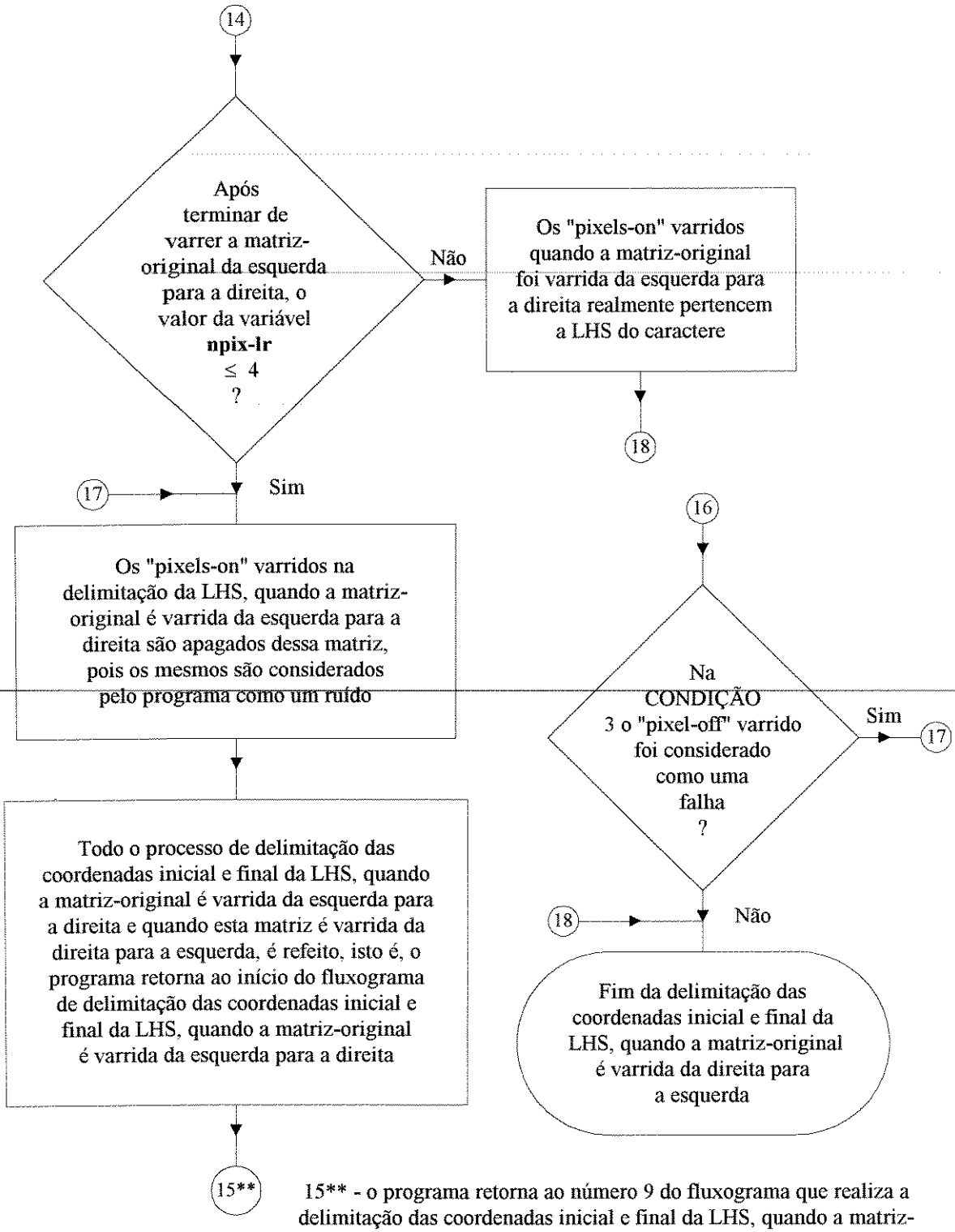
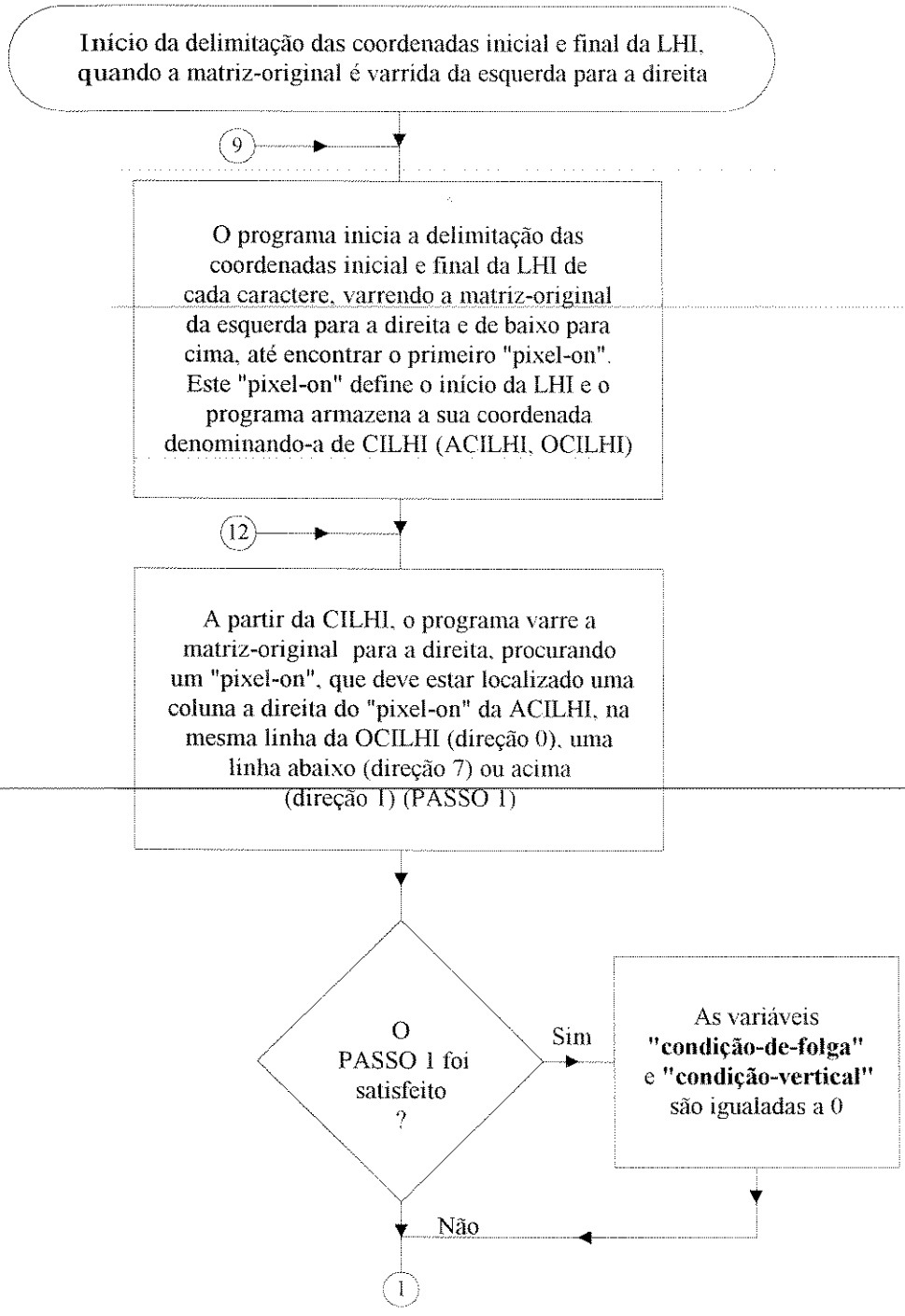


Figura 4.3(l) - Décima segunda parte do fluxograma que realiza a delimitação das coordenadas inicial e final da linha horizontal superior, quando a matriz-original é varrida da direita para a esquerda.

Fluxograma que realiza a delimitação das coordenadas inicial e final da linha horizontal inferior, quando a matriz-original é varrida da esquerda para a direita



LHI - linha horizontal inferior
CILHI - coordenada inicial da LHI
ACILHI - abscissa da coordenada inicial da LHI
OCILHI - ordenada da coordenada inicial da LHI

Figura 4.4(a) - Primeira parte do fluxograma que realiza a delimitação das coordenadas inicial e final da linha horizontal inferior, quando a matriz-original é varrida da esquerda para a direita.

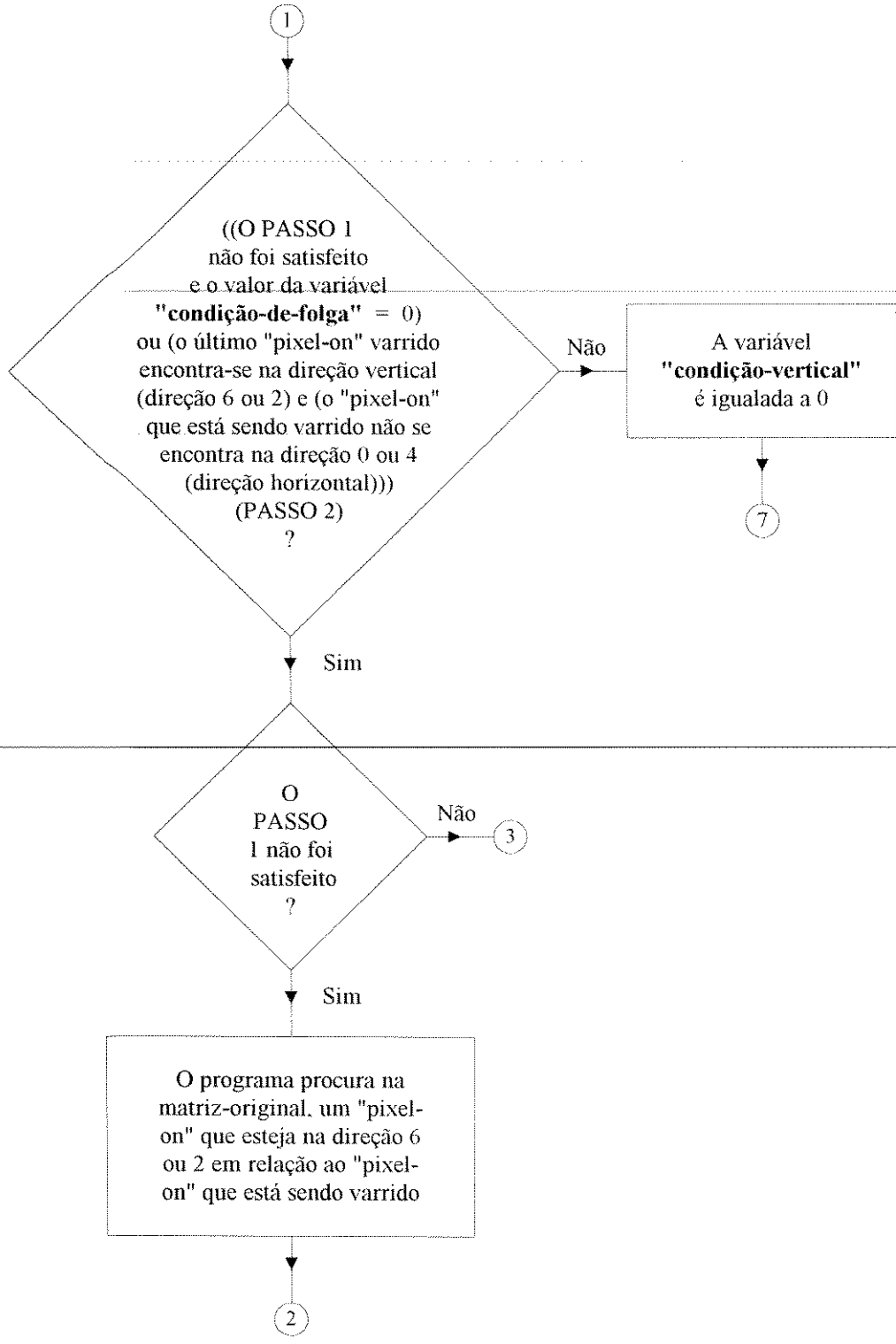


Figura 4.4(b) - Segunda parte do fluxograma que realiza a delimitação das coordenadas inicial e final da linha horizontal inferior, quando a matriz-original é varrida da esquerda para a direita.

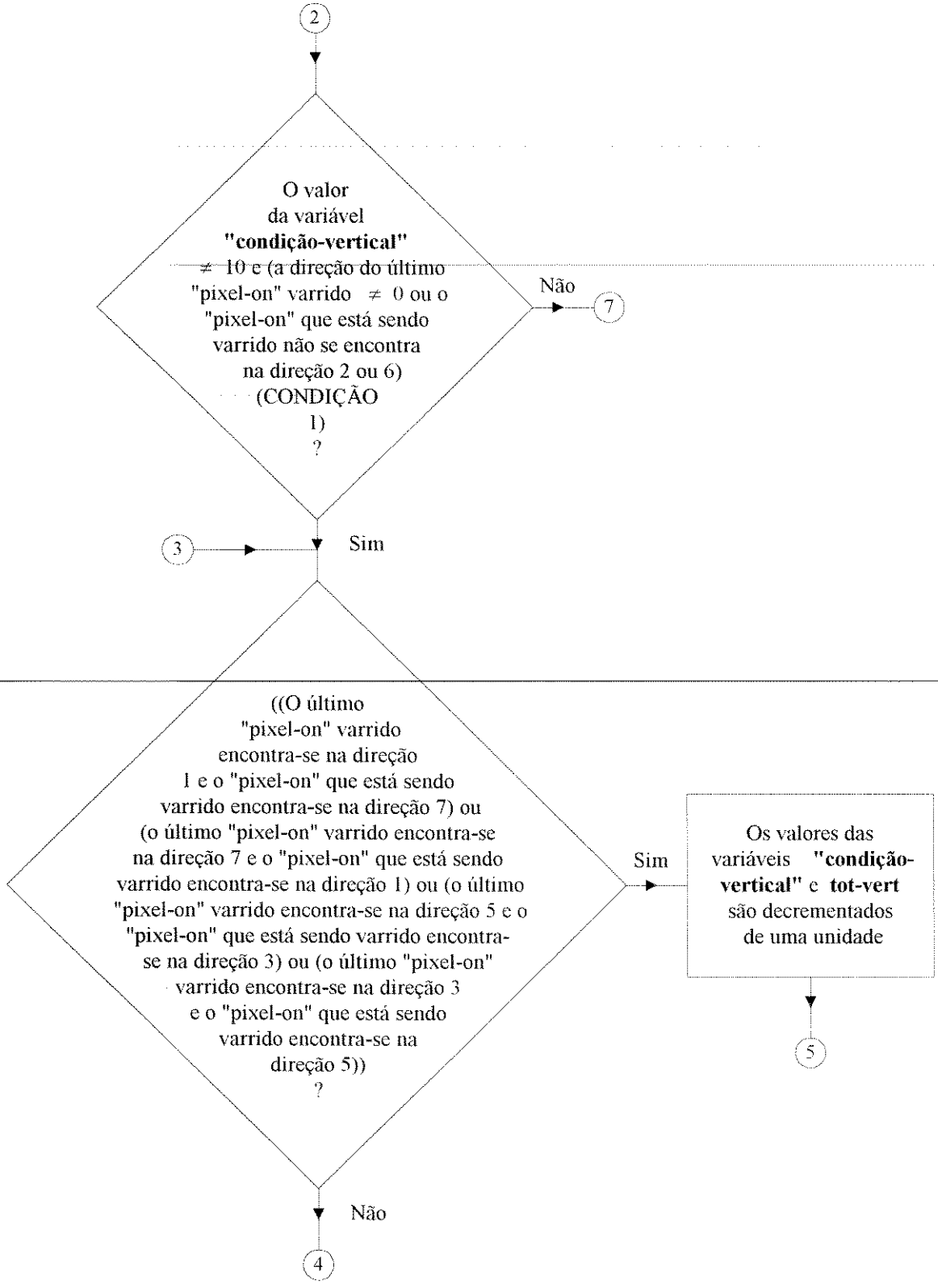


Figura 4.4(c) - Terceira parte do fluxograma que realiza a delimitação das coordenadas inicial e final da linha horizontal inferior, quando a matriz-original é varrida da esquerda para a direita.

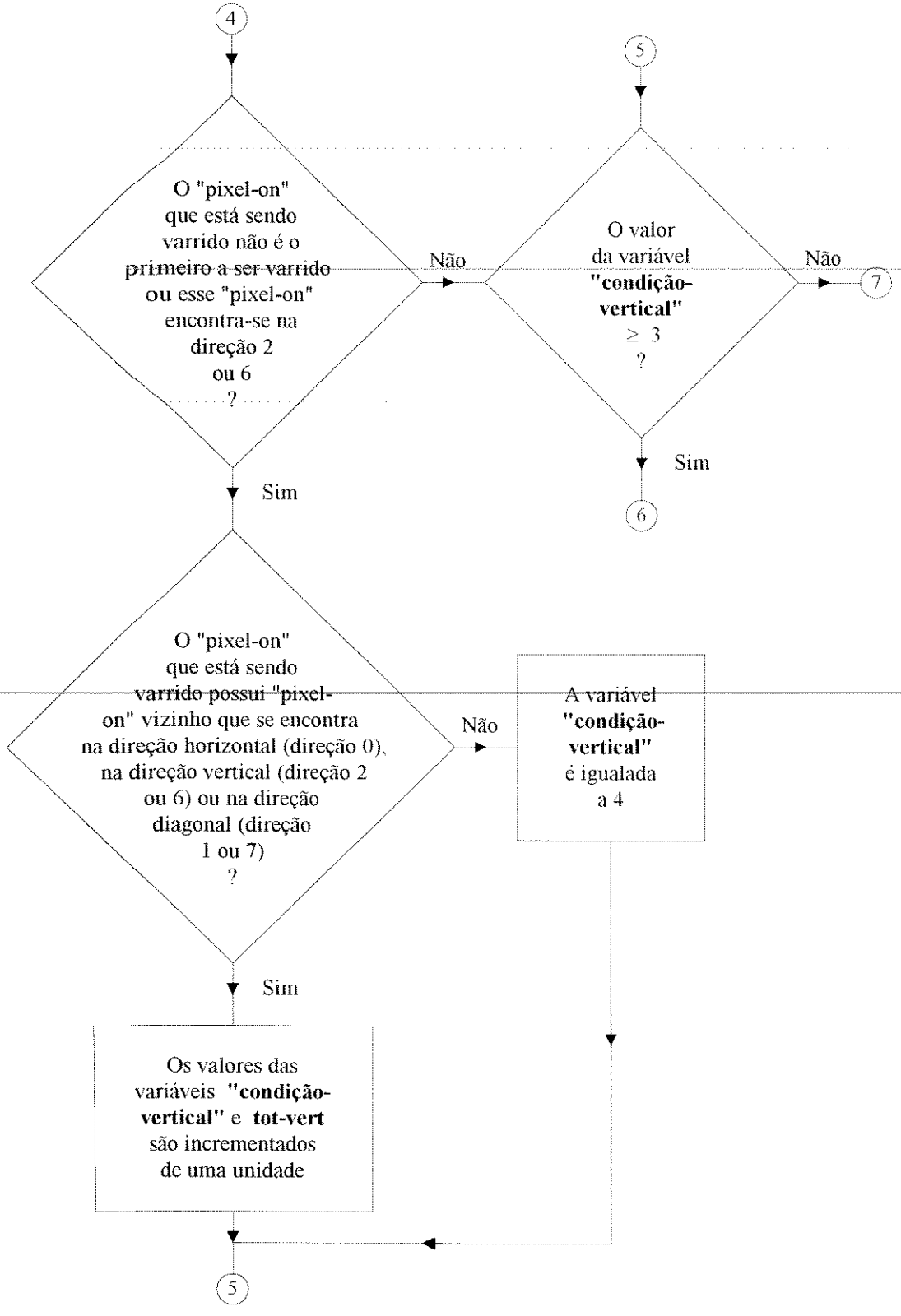


Figura 4.4(d) - Quarta parte do fluxograma que realiza a delimitação das coordenadas inicial e final da linha horizontal inferior, quando a matriz-original é varrida da esquerda para a direita.

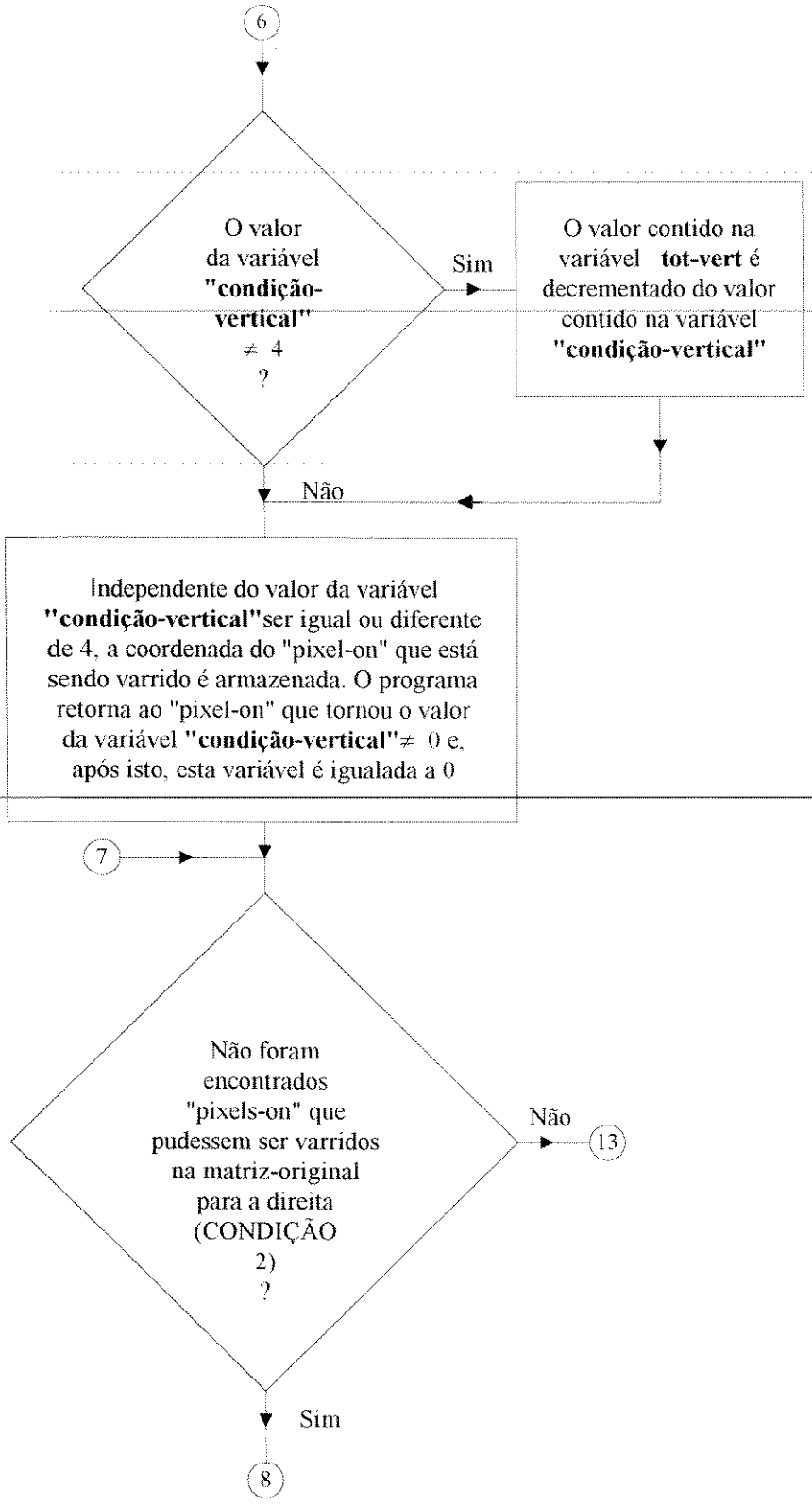
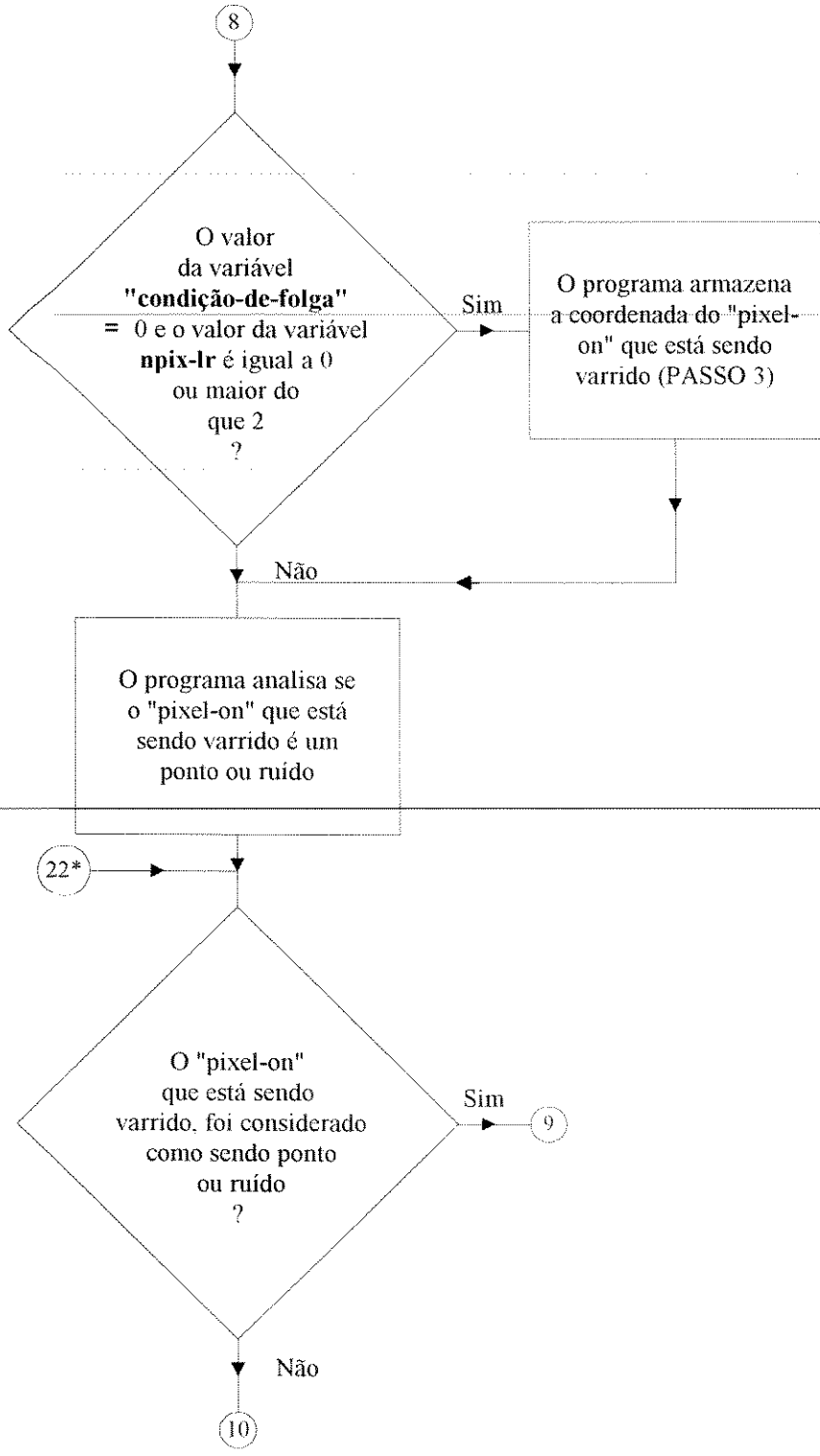


Figura 4.4(e) - Quinta parte do fluxograma que realiza a delimitação das coordenadas inicial e final da linha horizontal inferior, quando a matriz-original é varrida da esquerda para a direita.



22* - ponto em que o processo de eliminação do ponto e/ou ruído ilustrado na figura 5.3 do capítulo 5 deste trabalho retorna

Figura 4.4(f) - Sexta parte do fluxograma que realiza a delimitação das coordenadas inicial e final da linha horizontal inferior, quando a matriz-original é varrida da esquerda para a direita.

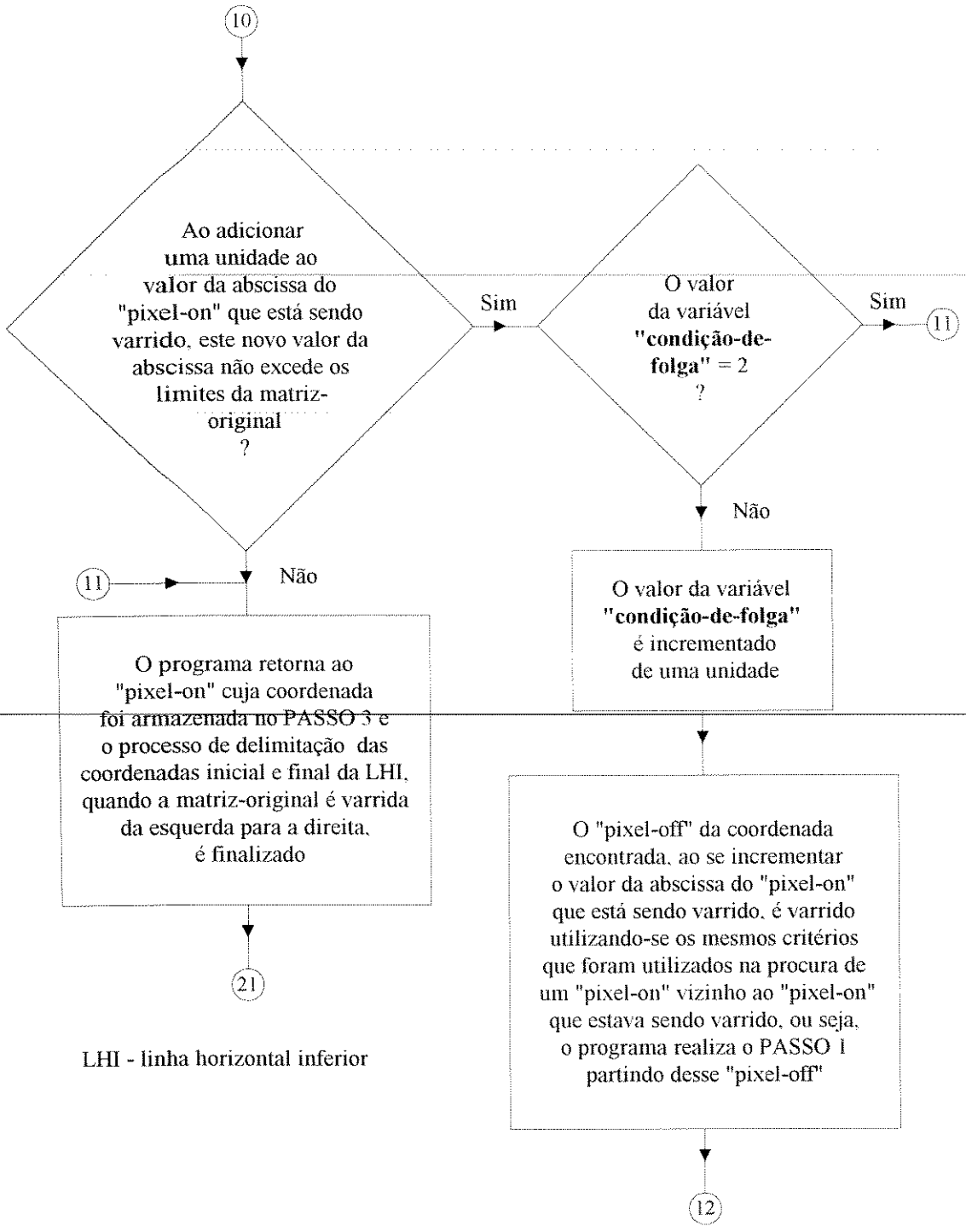


Figura 4.4(g) - Sétima parte do fluxograma que realiza a delimitação das coordenadas inicial e final da linha horizontal inferior, quando a matriz-original é varrida da esquerda para a direita.

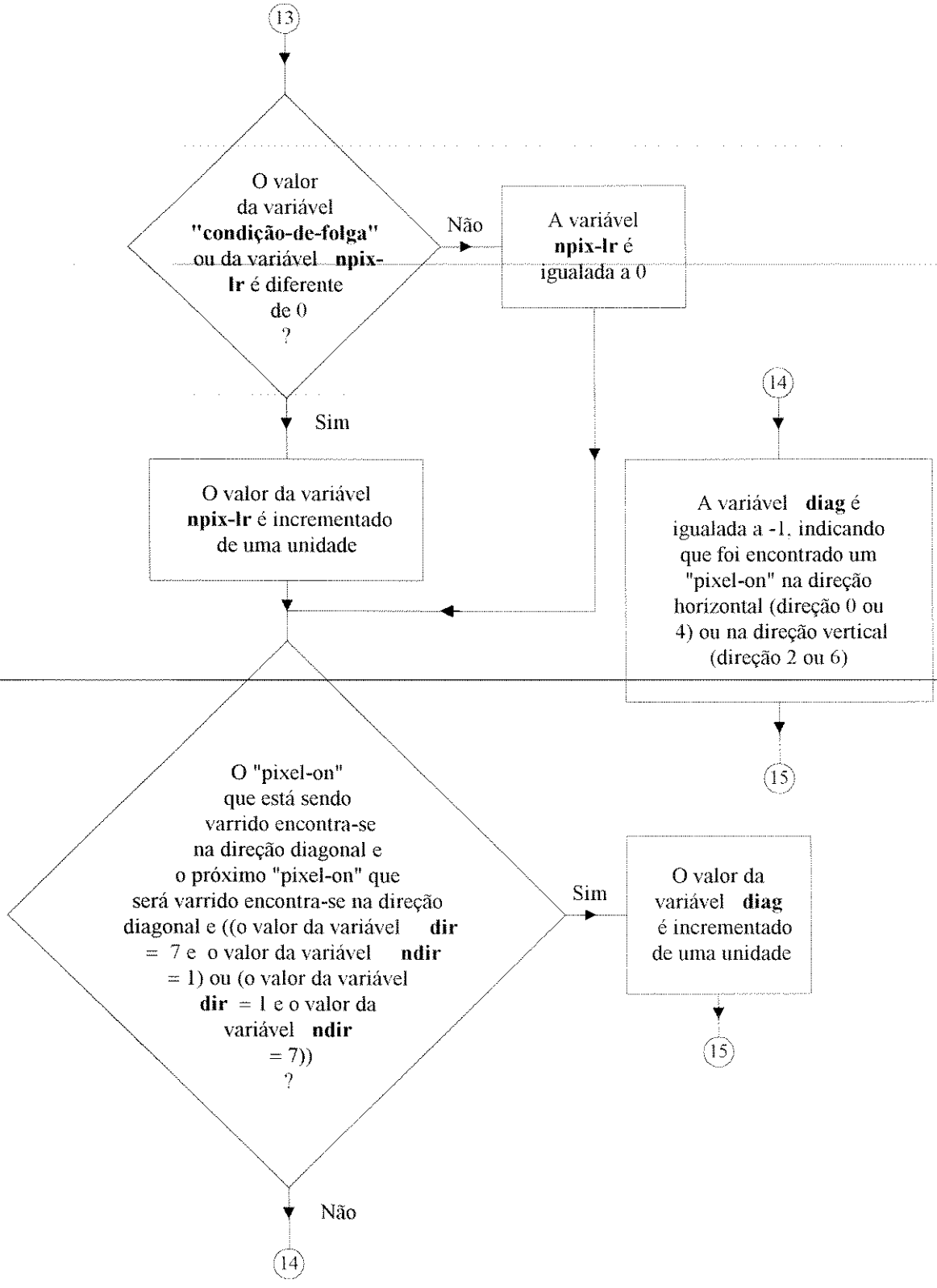
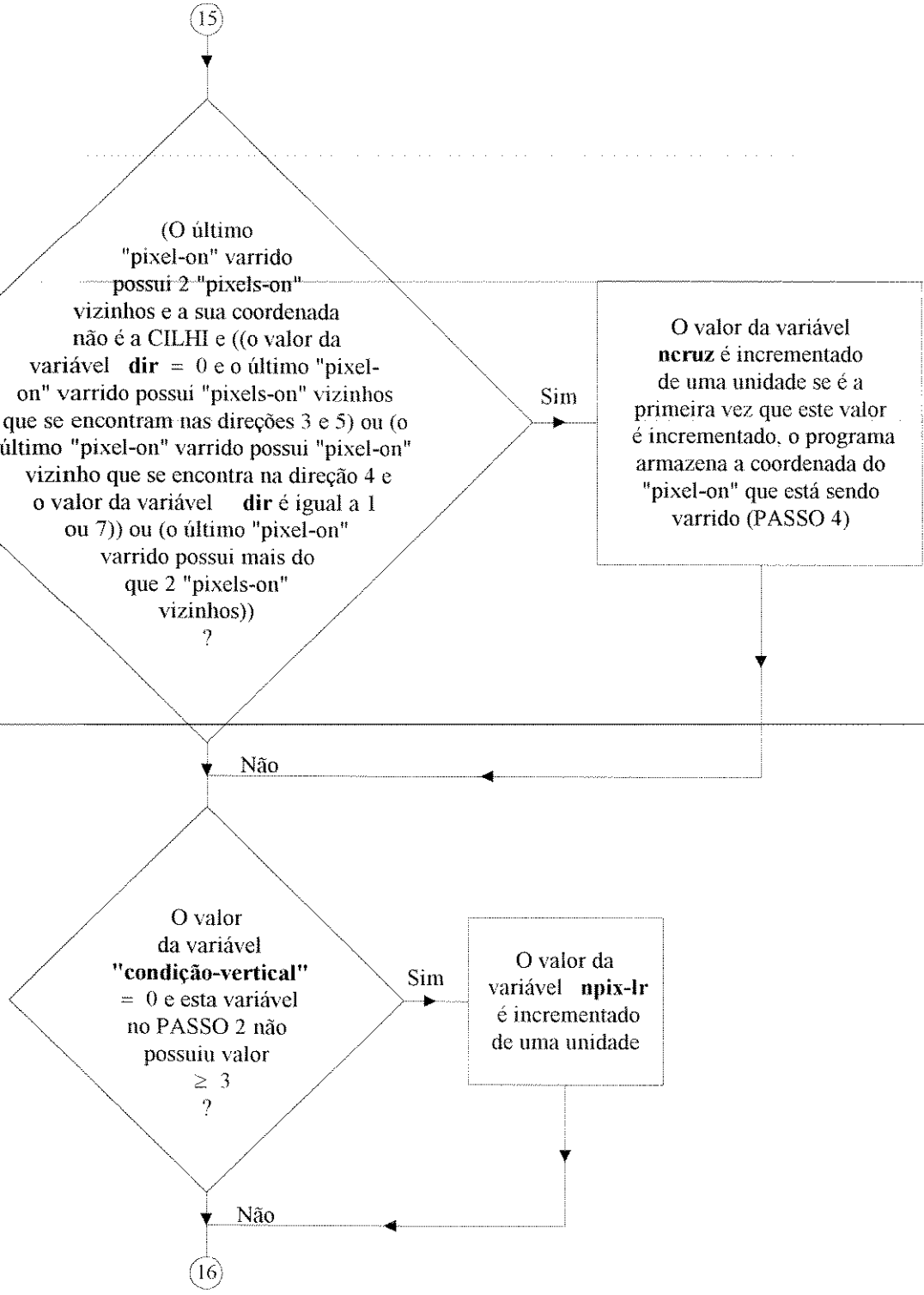


Figura 4.4(h) - Oitava parte do fluxograma que realiza a delimitação das coordenadas inicial e final da linha horizontal inferior, quando a matriz-original é varrida da esquerda para a direita.



LHI - linha horizontal inferior
CILHI - coordenada inicial da LHI

Figura 4.4(i) - Nona parte do fluxograma que realiza a delimitação das coordenadas inicial e final da linha horizontal inferior, quando a matriz-original é varrida da esquerda para a direita.

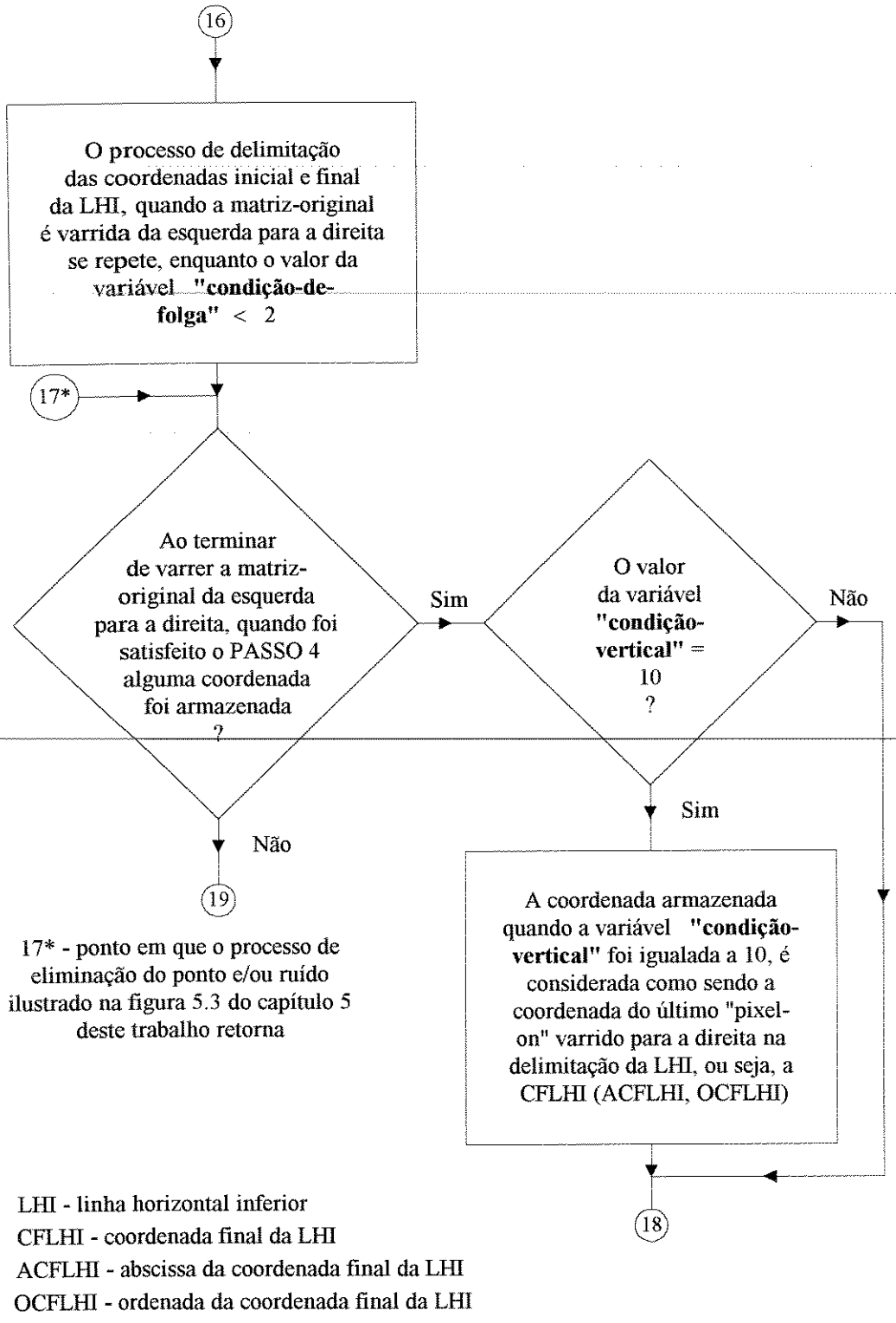


Figura 4.4(j) - Décima parte do fluxograma que realiza a delimitação das coordenadas inicial e final da linha horizontal inferior, quando a matriz-original é varrida da esquerda para a direita.

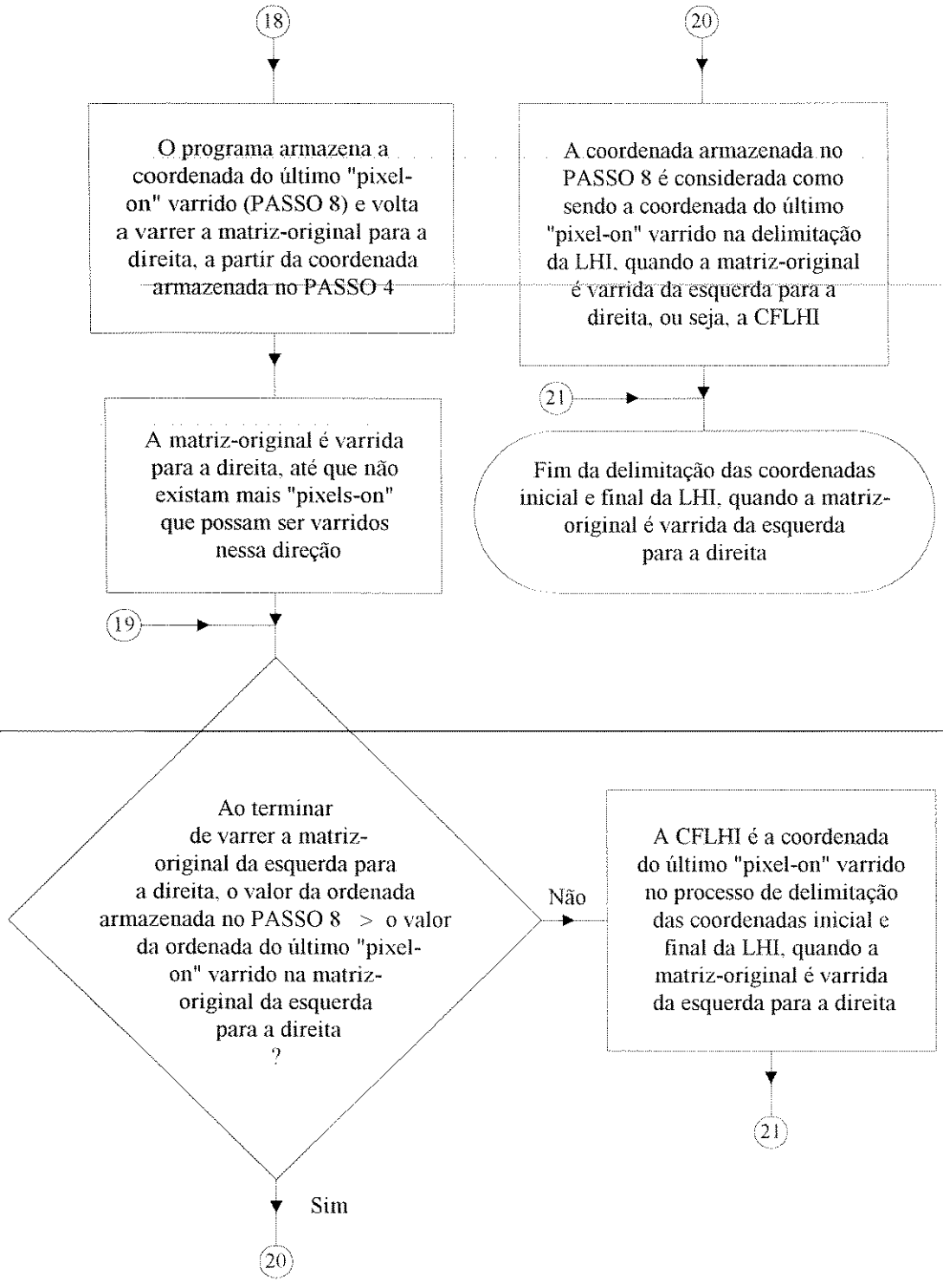


Figura 4.4(k) - Décima primeira parte do fluxograma que realiza a delimitação das coordenadas inicial e final da linha horizontal inferior, quando a matriz-original é varrida da esquerda para a direita.

Fluxograma que realiza a delimitação das coordenadas inicial e final da linha horizontal inferior, quando a matriz-original é varrida da direita para a esquerda

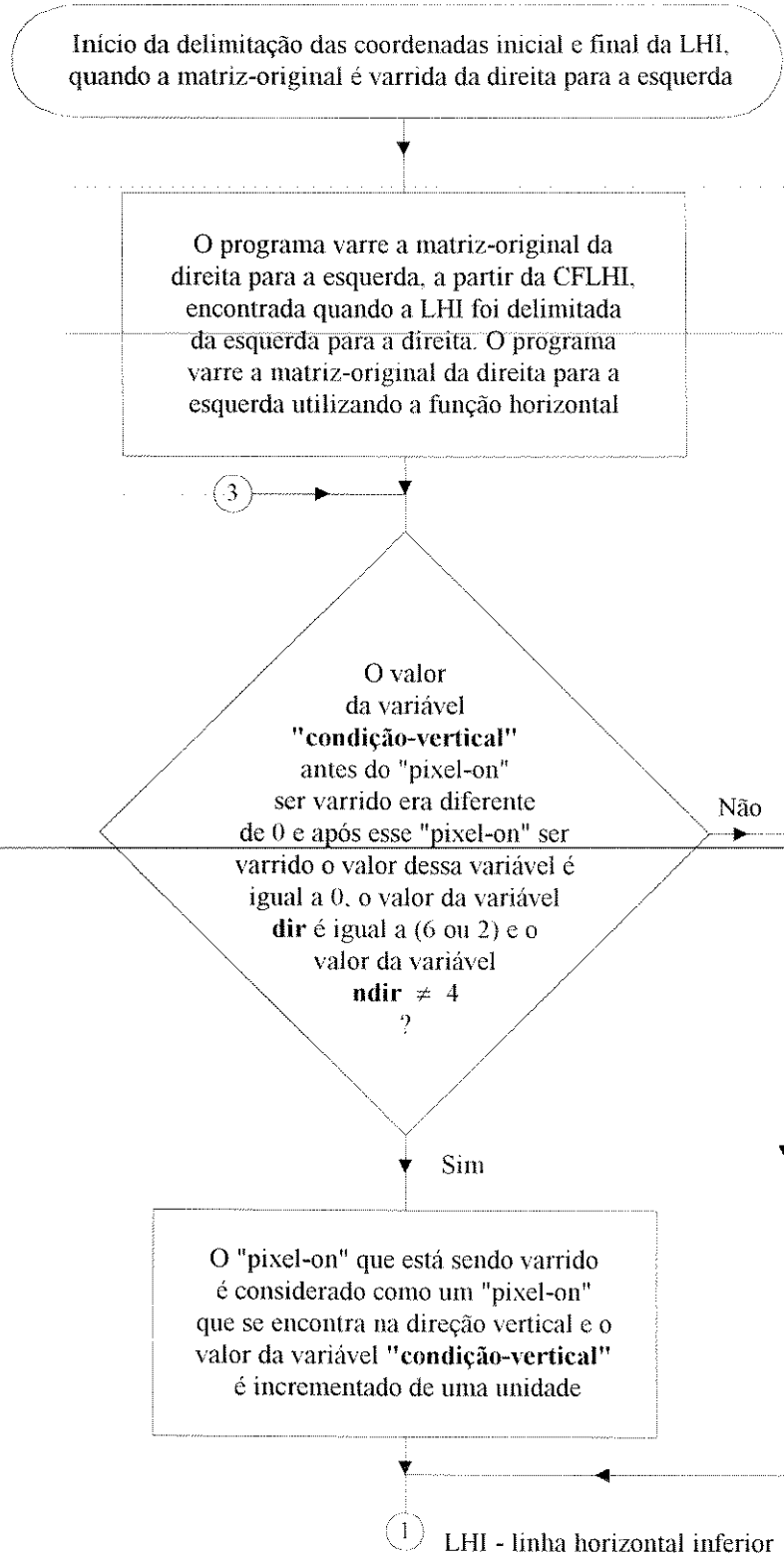


Figura 4.6(a) - Primeira parte do fluxograma que realiza a delimitação das coordenadas inicial e final da linha horizontal inferior, quando a matriz-original é varrida da direita para a esquerda.

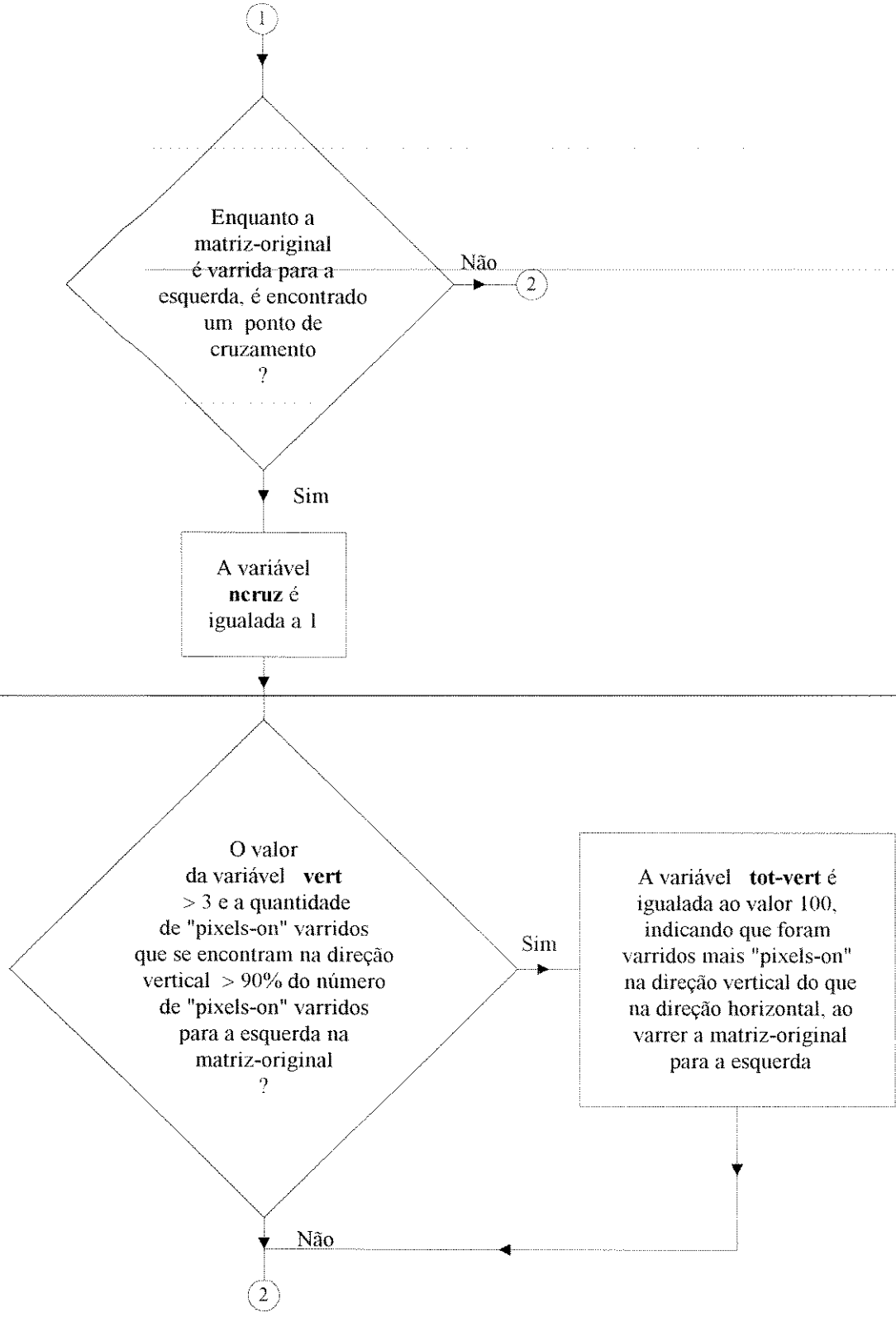


Figura 4.6(b) - Segunda parte do fluxograma que realiza a delimitação das coordenadas inicial e final da linha horizontal inferior, quando a matriz-original é varrida da direita para a esquerda.

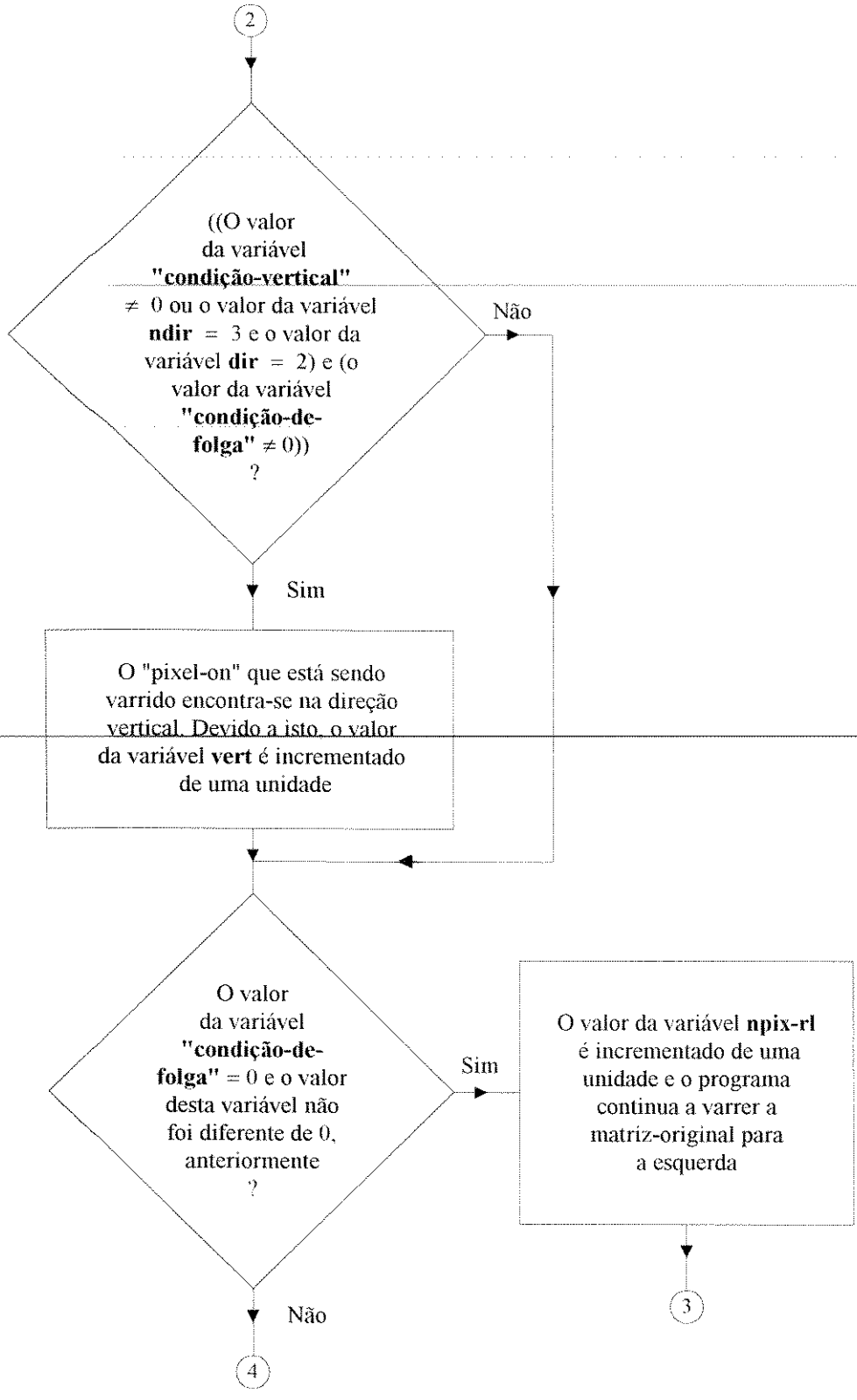


Figura 4.6(c) - Terceira parte do fluxograma que realiza a delimitação das coordenadas inicial e final da linha horizontal inferior, quando a matriz-original é varrida da direita para a esquerda.

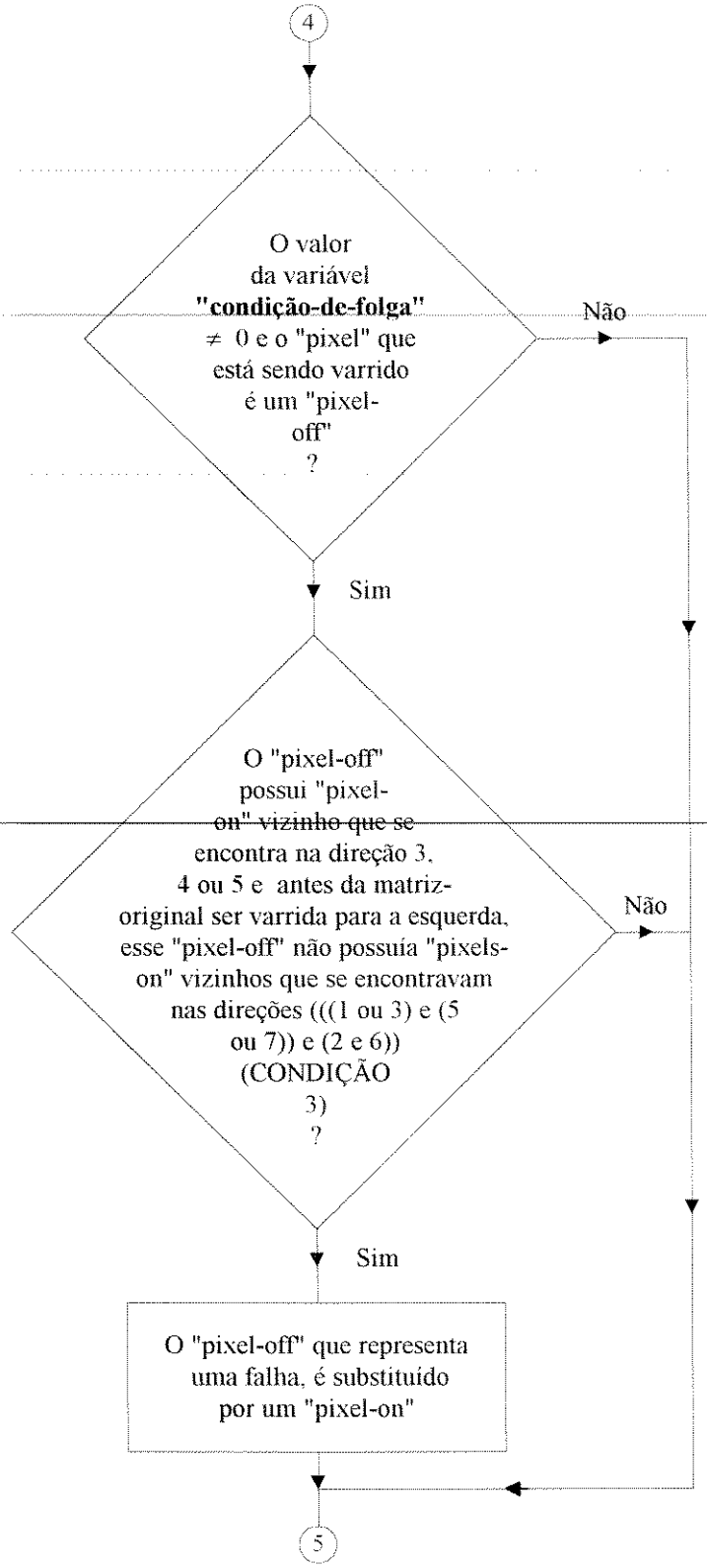


Figura 4.6(d) - Quarta parte do fluxograma que realiza a delimitação das coordenadas inicial e final da linha horizontal inferior, quando a matriz-original é varrida da direita para a esquerda.

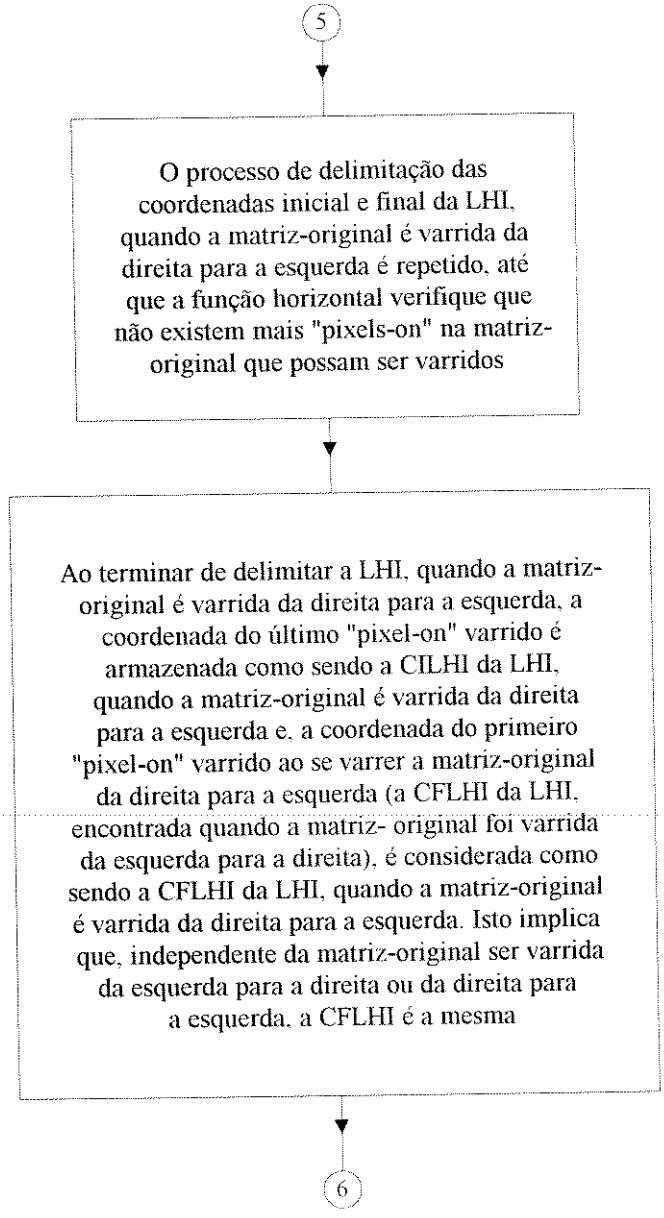
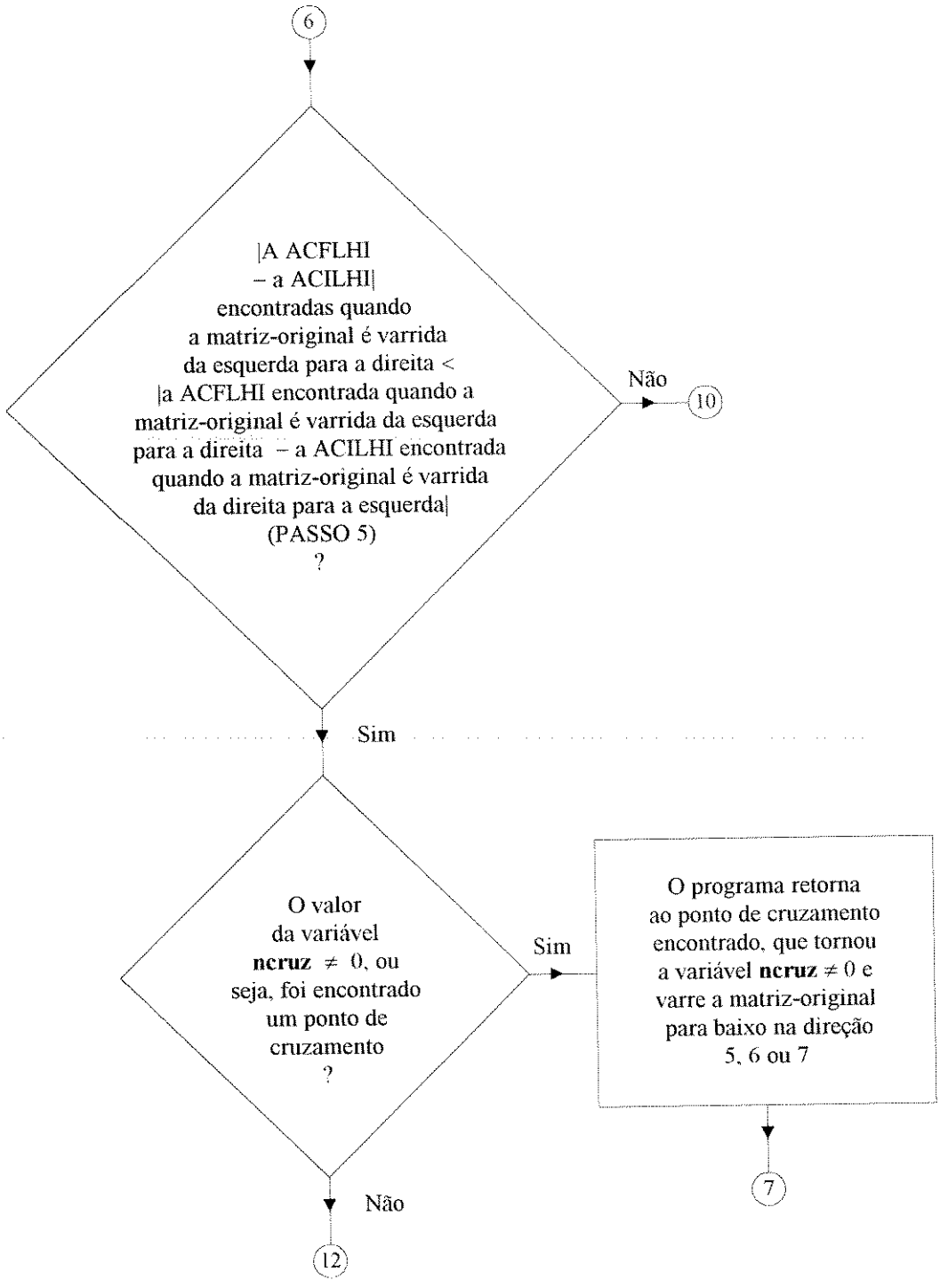


Figura 4.6(e) - Quinta parte do fluxograma que realiza a delimitação das coordenadas inicial e final da linha horizontal inferior, quando a matriz-original é varrida da direita para a esquerda.



LHI - linha horizontal inferior
ACILHI - abscissa da coordenada inicial da LHI
ACFLHI - abscissa da coordenada final da LHI

Figura 4.6(f) - Sexta parte do fluxograma que realiza a delimitação das coordenadas inicial e final da linha horizontal inferior, quando a matriz-original é varrida da direita para a esquerda.

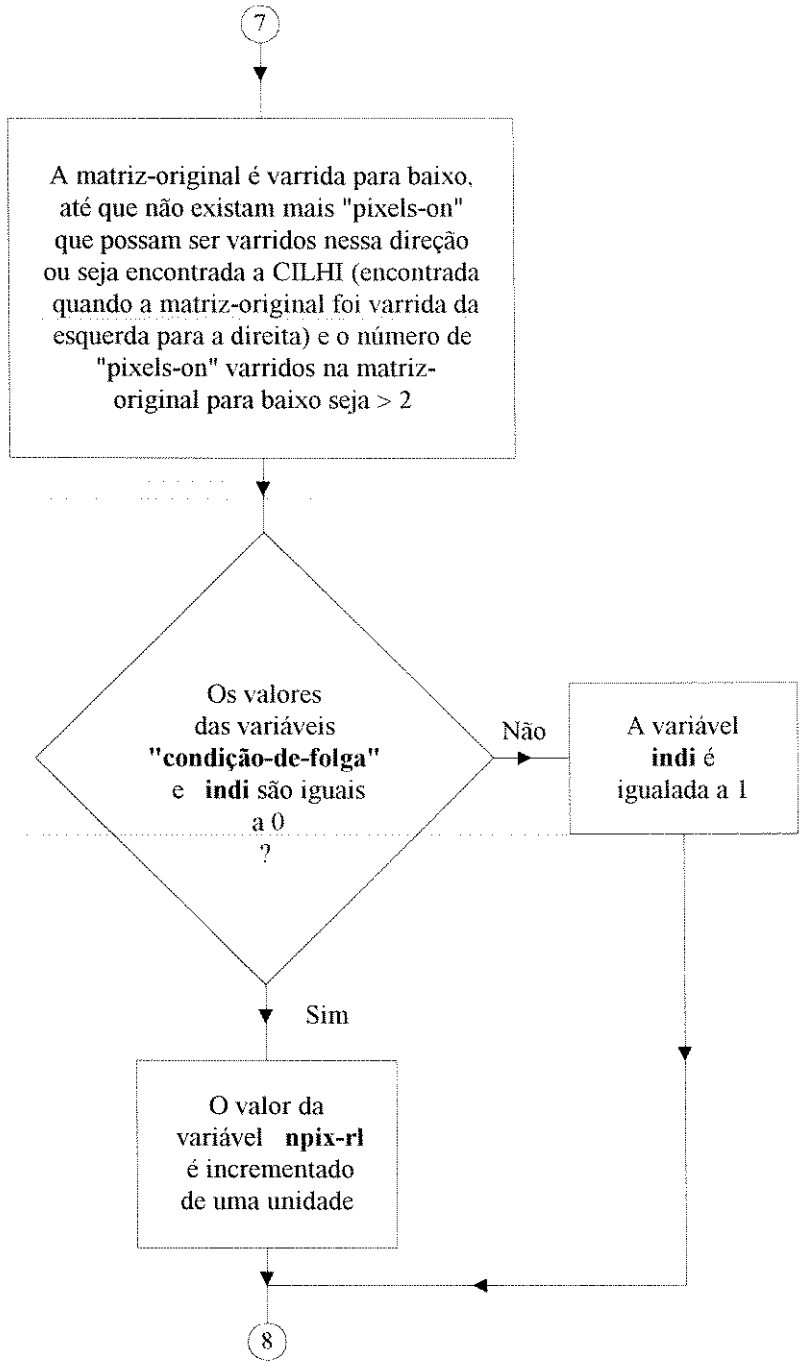
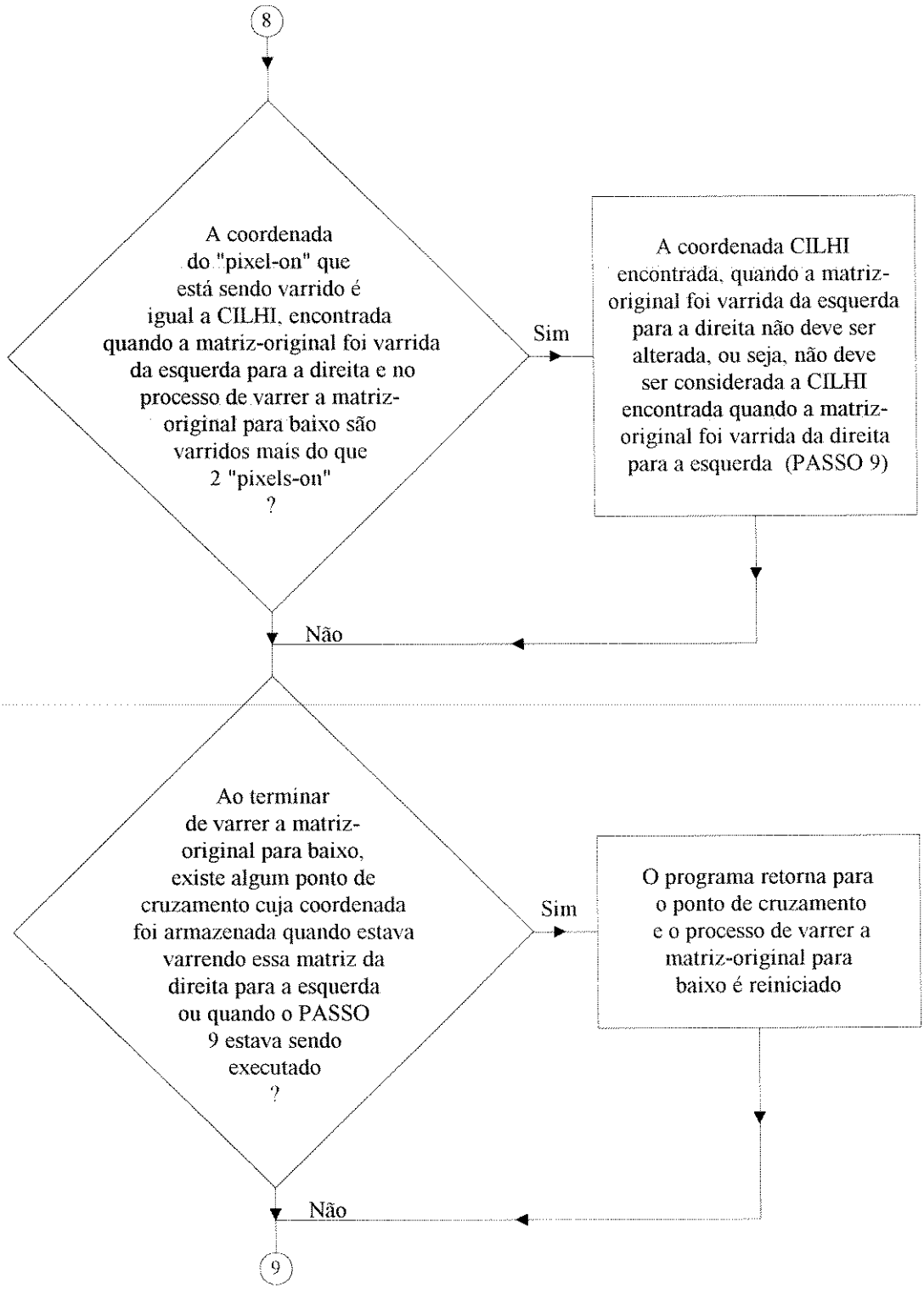


Figura 4.6(g) - Sétima parte do fluxograma que realiza a delimitação das coordenadas inicial e final da linha horizontal inferior, quando a matriz-original é varrida da direita para a esquerda.



LHI - linha horizontal inferior
CILHI - coordenada inicial da LHI

Figura 4.6(h) - Oitava parte do fluxograma que realiza a delimitação das coordenadas inicial e final da linha horizontal inferior, quando a matriz-original é varrida da direita para a esquerda.

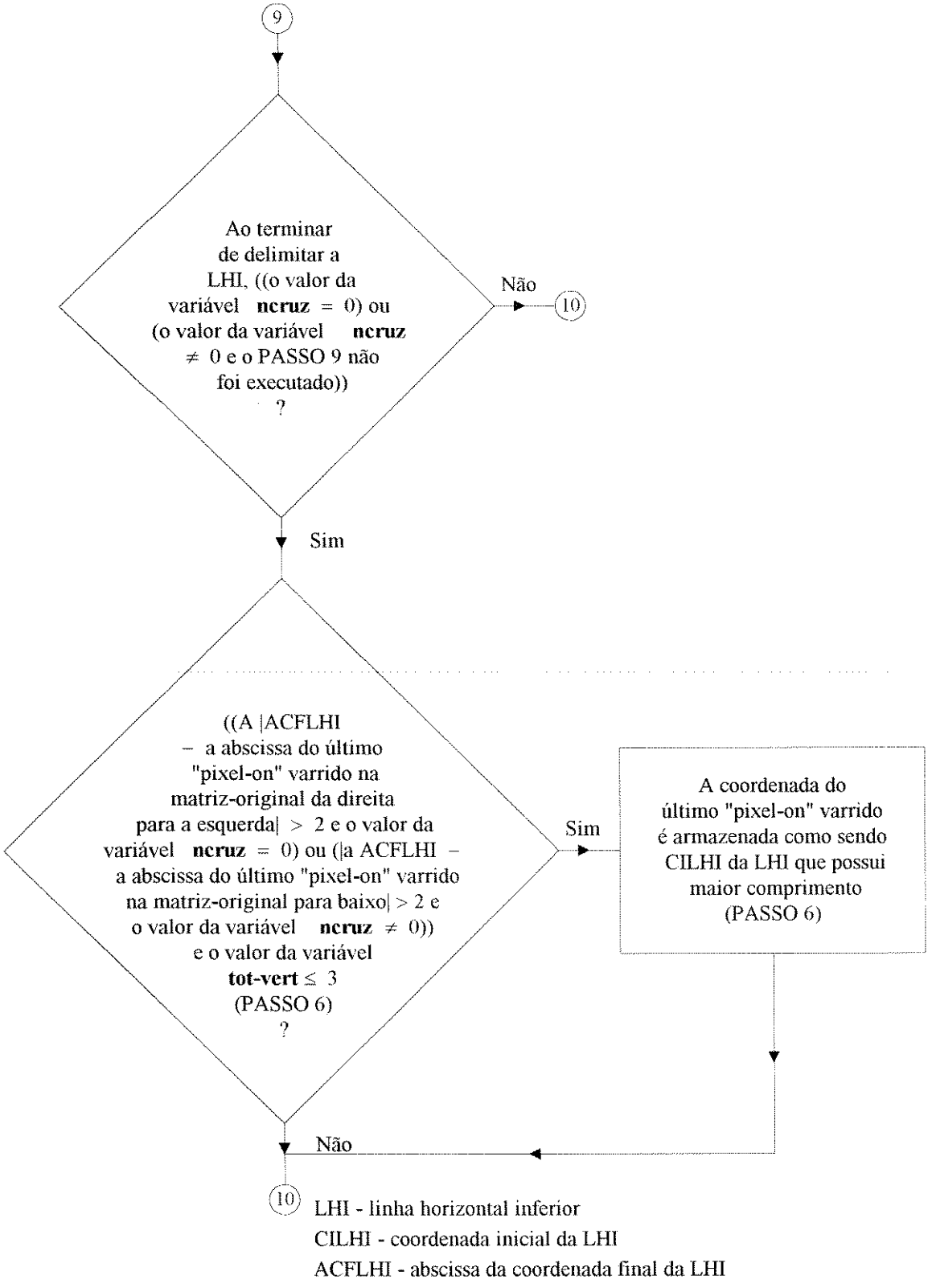


Figura 4.6(i) - Nona parte do fluxograma que realiza a delimitação das coordenadas inicial e final da linha horizontal inferior, quando a matriz-original é varrida da direita para a esquerda.

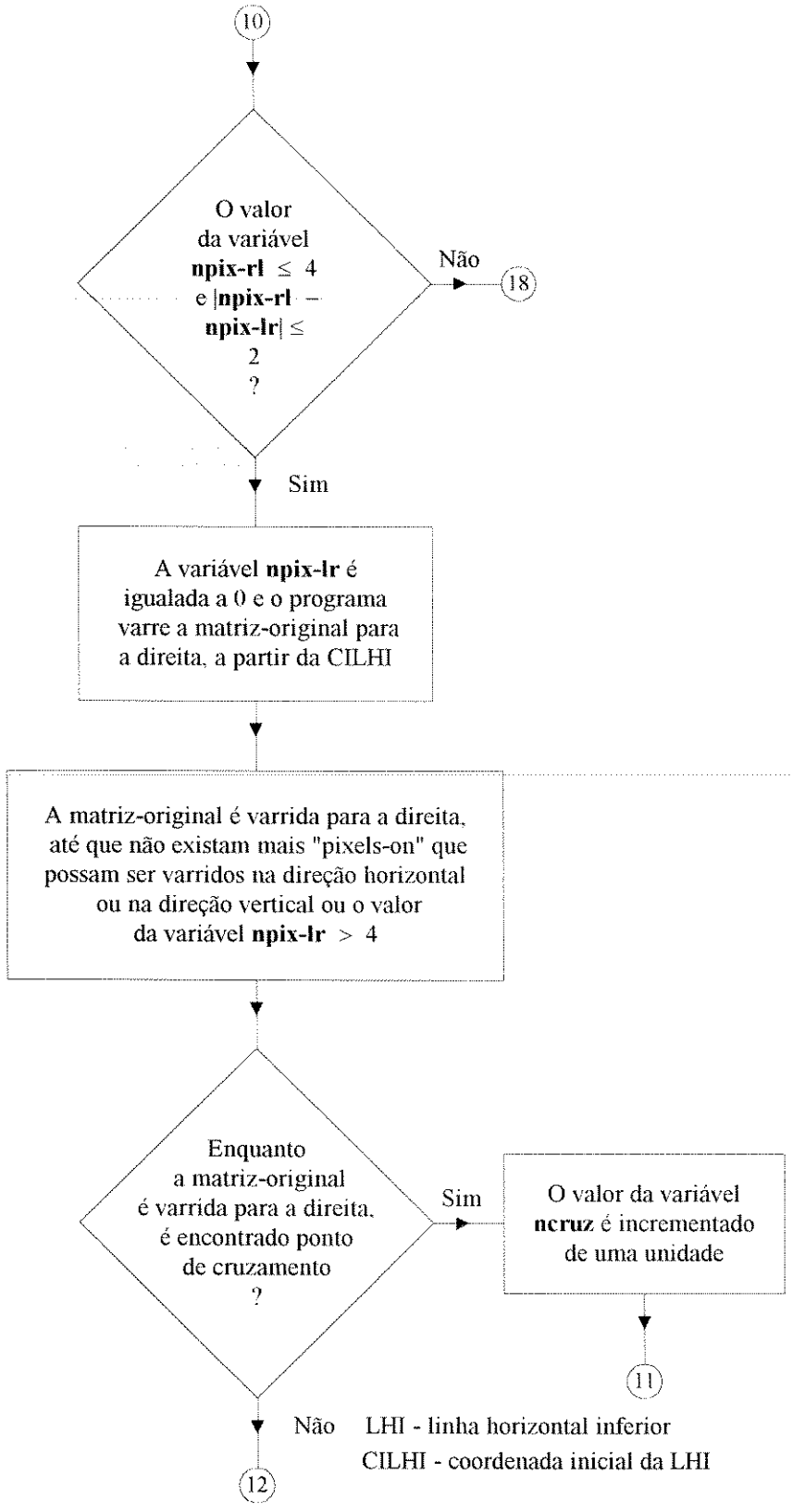


Figura 4.6(j) - Décima parte do fluxograma que realiza a delimitação das coordenadas inicial e final da linha horizontal inferior, quando a matriz-original é varrida da direita para a esquerda.

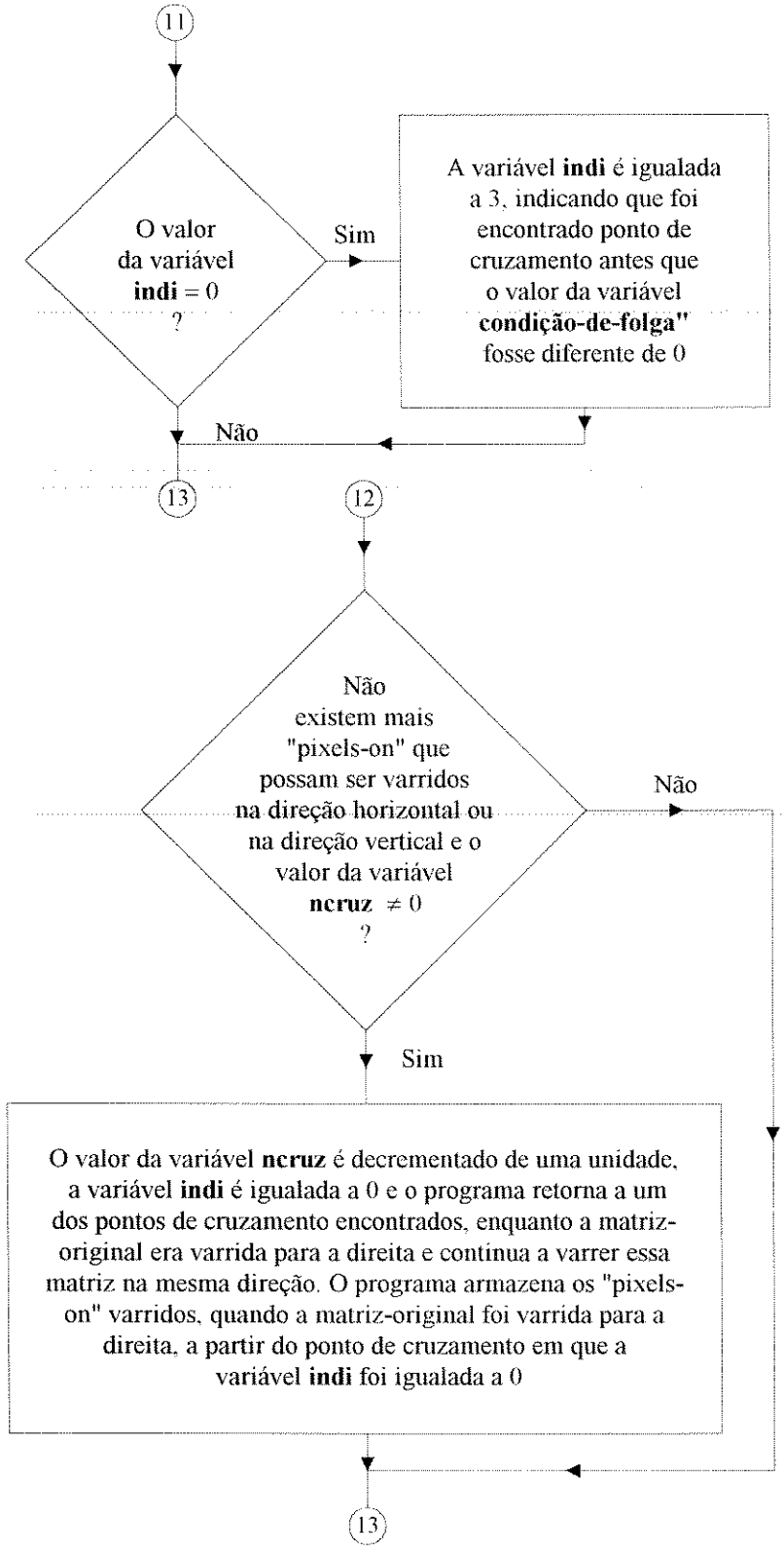


Figura 4.6(k)- Décima primeira parte do fluxograma que realiza a delimitação das coordenadas inicial e final da linha horizontal inferior, quando a matriz-original é varrida da direita para a esquerda.

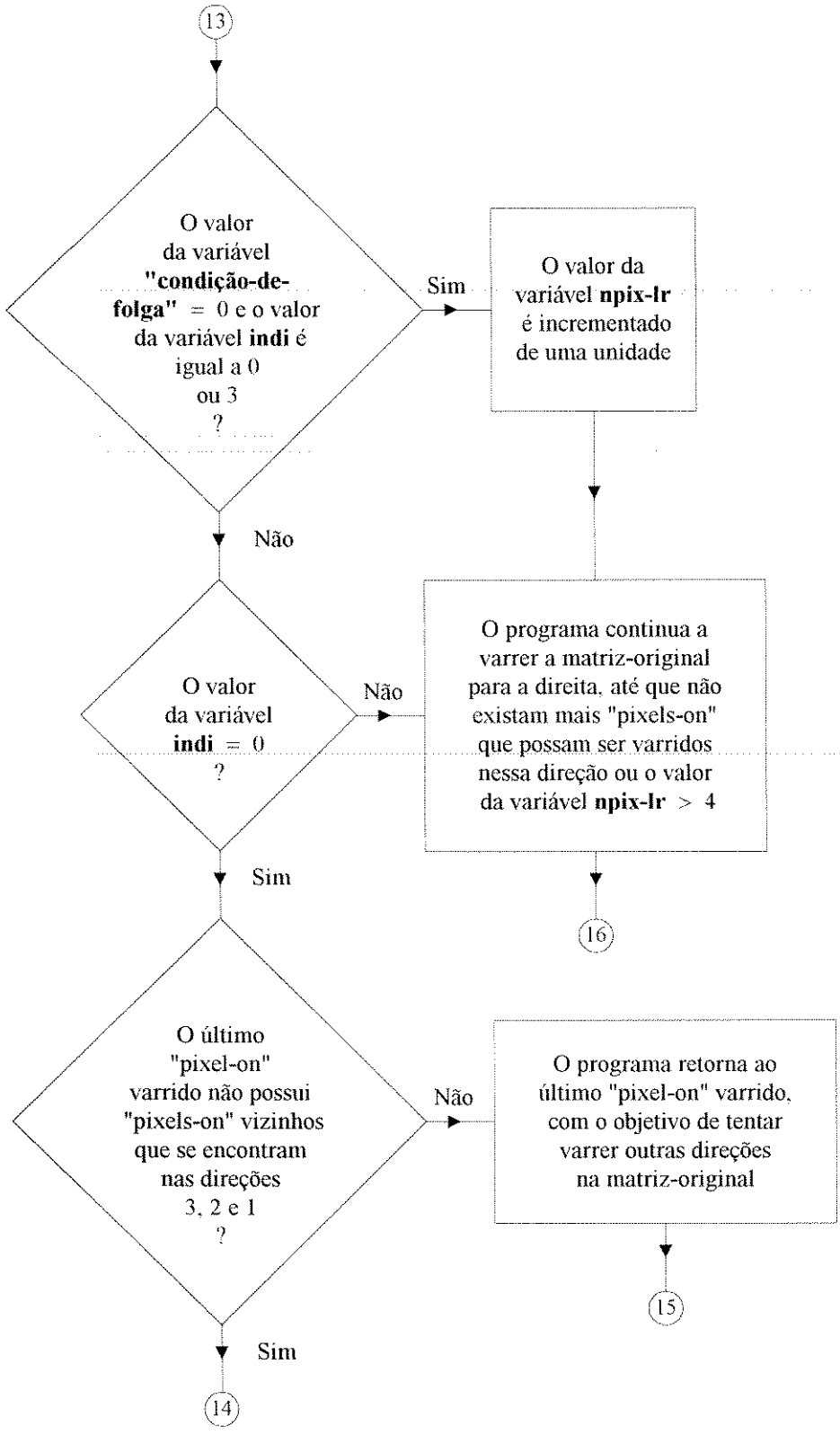


Figura 4.6(l)- Décima segunda parte do fluxograma que realiza a delimitação das coordenadas inicial e final da linha horizontal inferior, quando a matriz-original é varrida da direita para a esquerda.

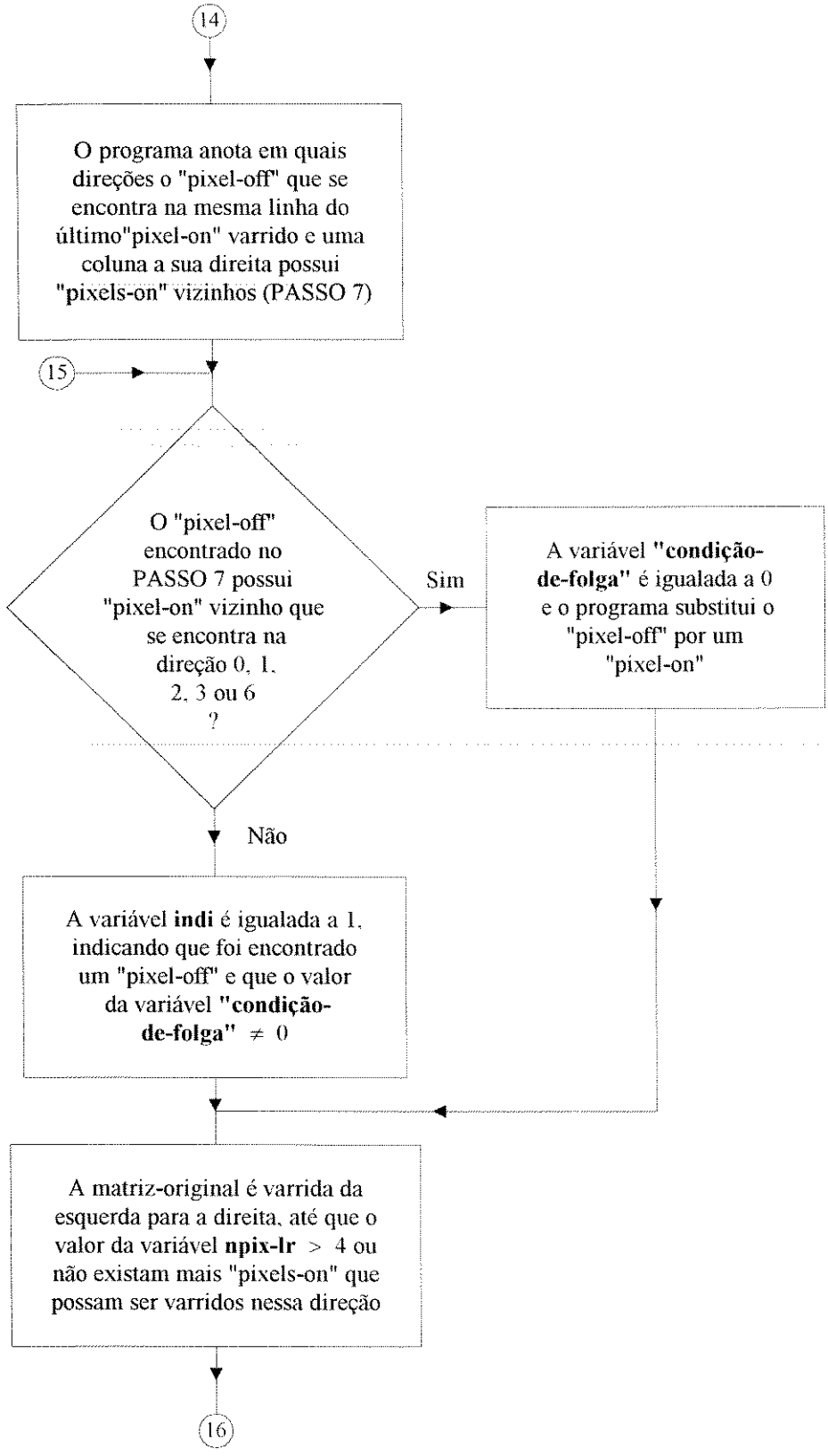


Figura 4.6(m)- Décima terceira parte do fluxograma que realiza a delimitação das coordenadas inicial e final da linha horizontal inferior, quando a matriz-original é varrida da direita para a esquerda.

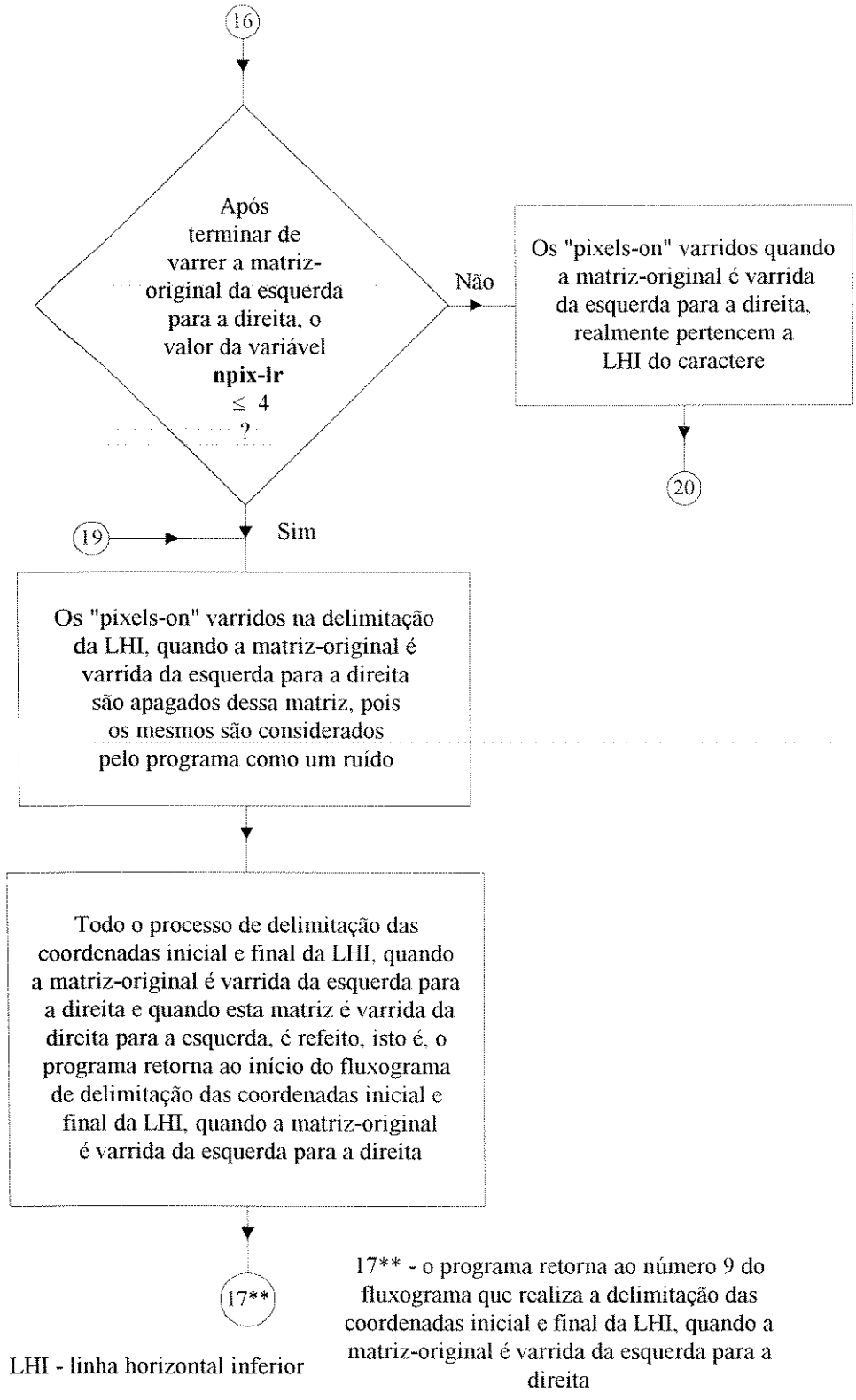
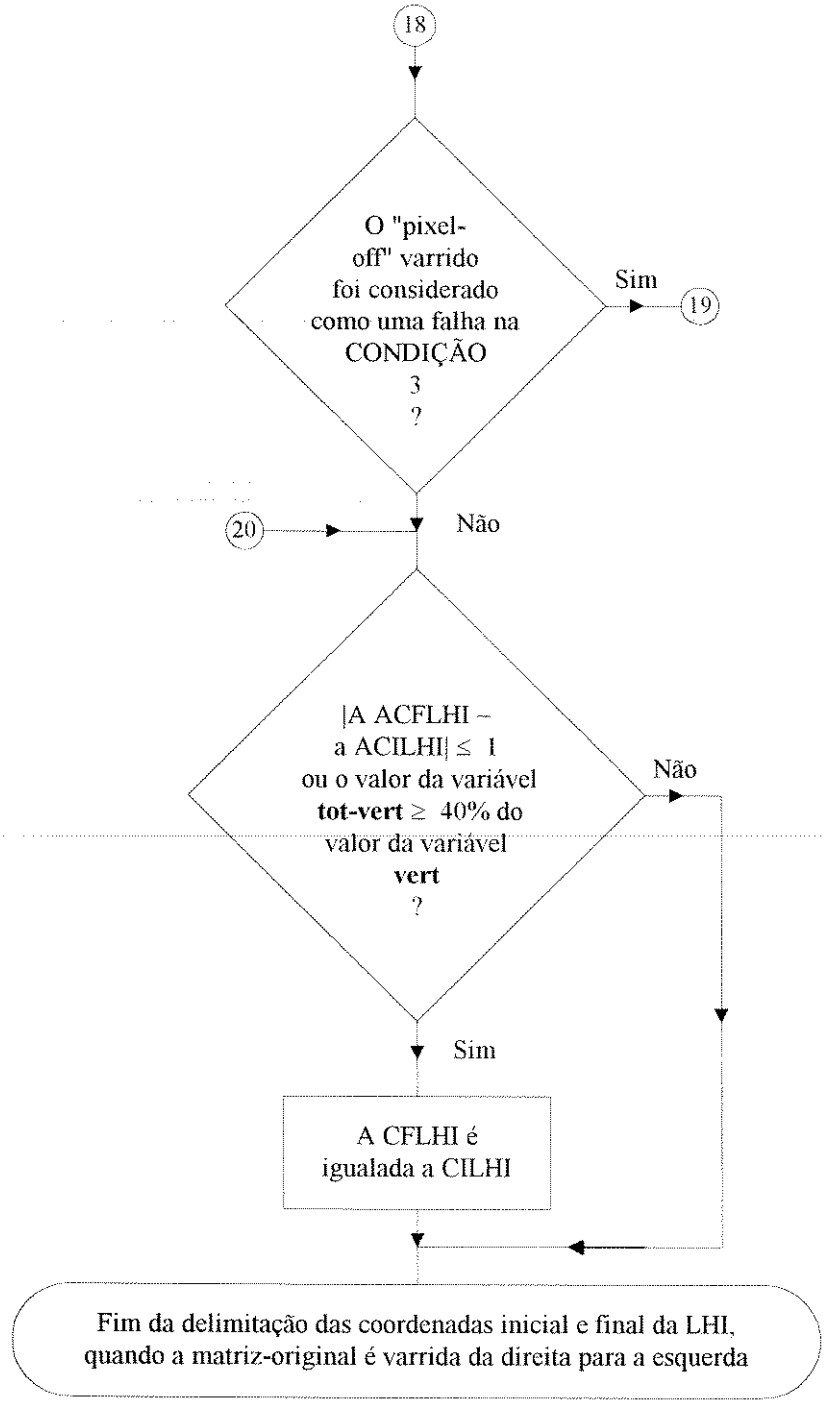


Figura 4.6(n)- Décima quarta parte do fluxograma que realiza a delimitação das coordenadas inicial e final da linha horizontal inferior, quando a matriz-original é varrida da direita para a esquerda.



LHI - linha horizontal inferior
CILHI - coordenada inicial da LHI
ACILHI - abscissa da coordenada inicial da LHI
CFLHI - coordenada final da LHI
ACFLHI abscissa da coordenada final da LHI

Figura 4.6(o)- Décima quinta parte do fluxograma que realiza a delimitação ds coordenadas inicial e final da linha horizontal inferior, quando a matriz-original é varrida da direita para a esquerda.

Fluxograma que realiza a delimitação das coordenadas inicial e final da linha vertical esquerda

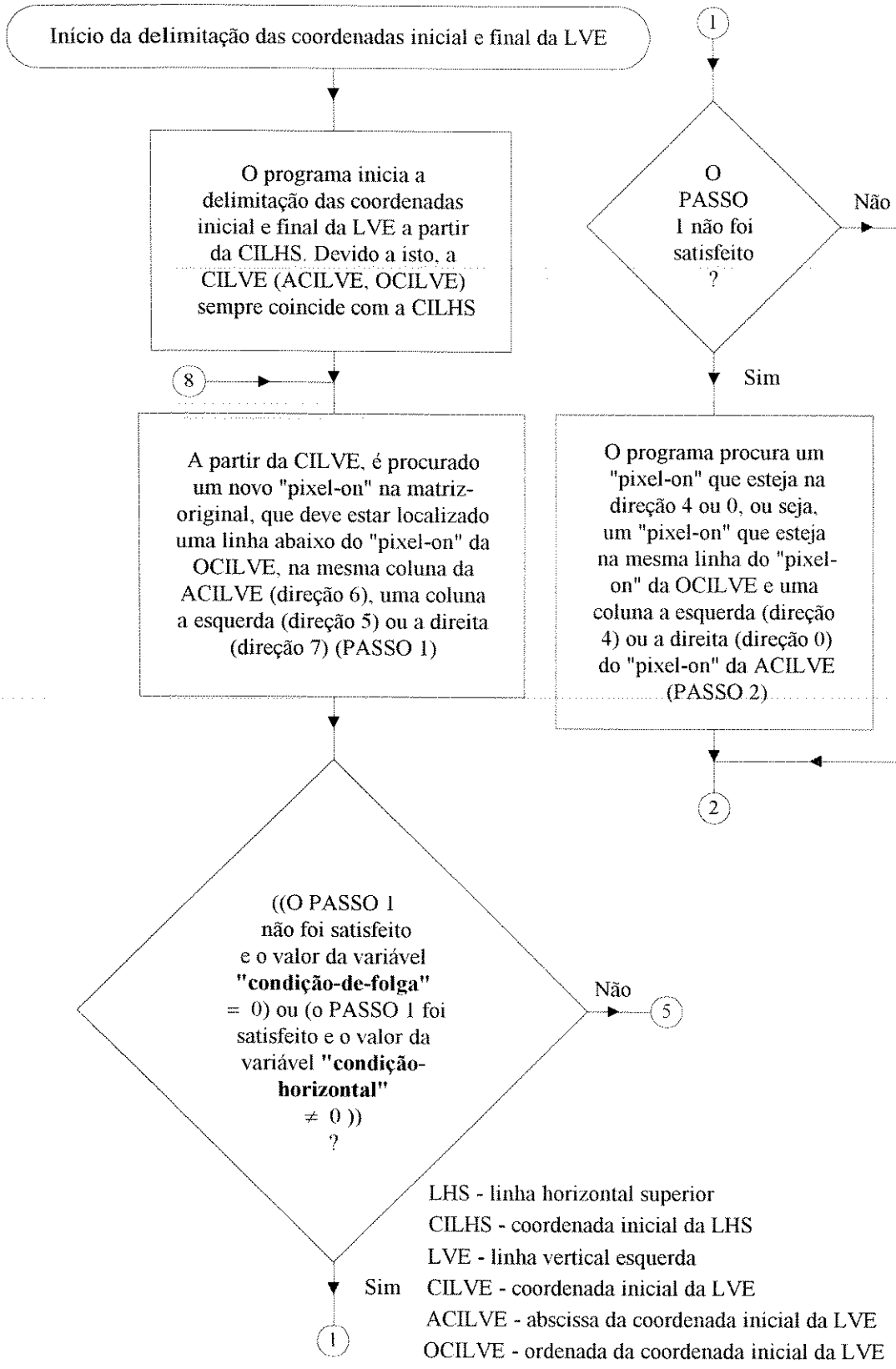


Figura 4.7(a) - Primeira parte do fluxograma que realiza a delimitação das coordenadas inicial e final da linha vertical esquerda.

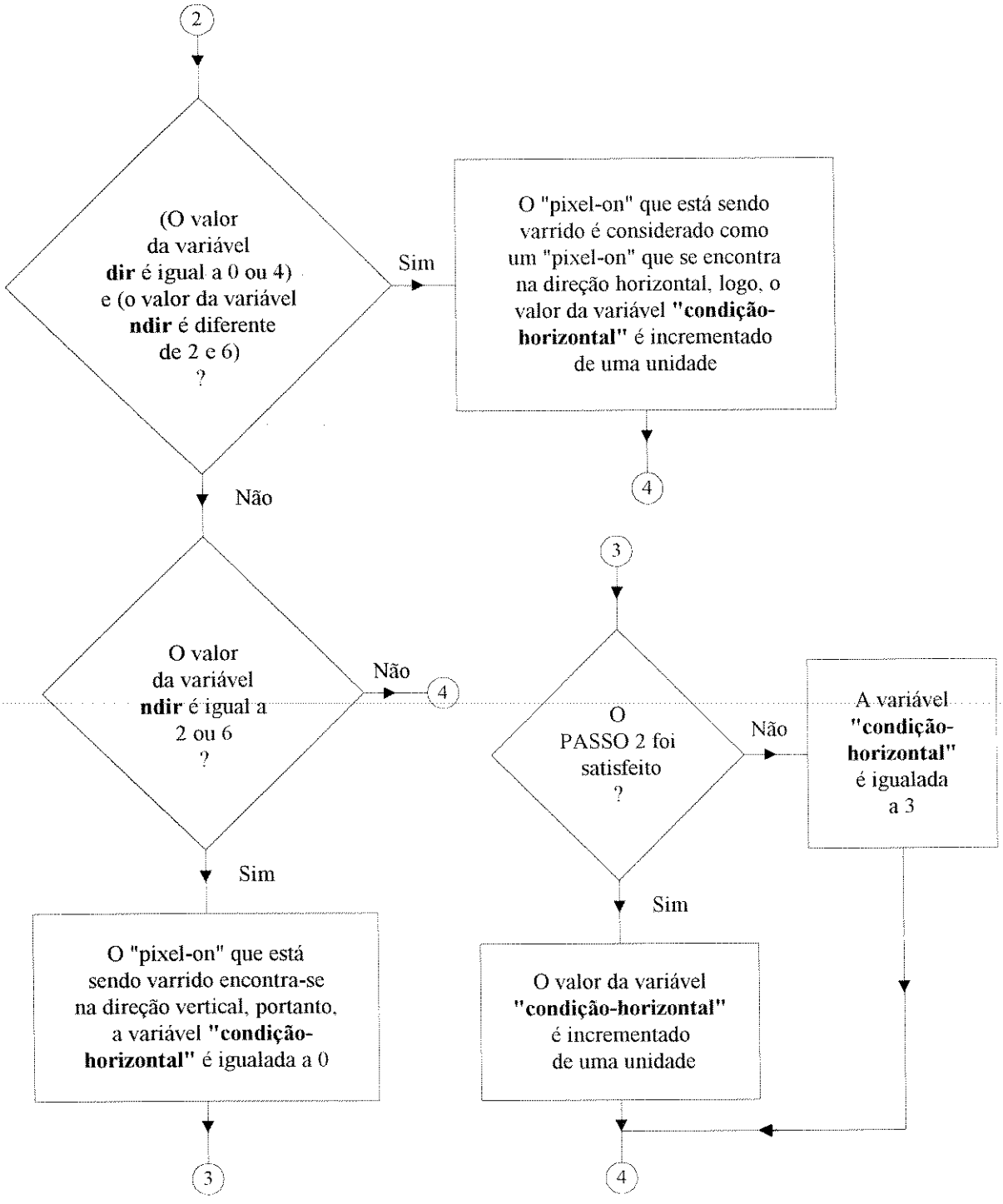


Figura 4.7(b) - Segunda parte do fluxograma que realiza a delimitação das coordenadas inicial e final da linha vertical esquerda.

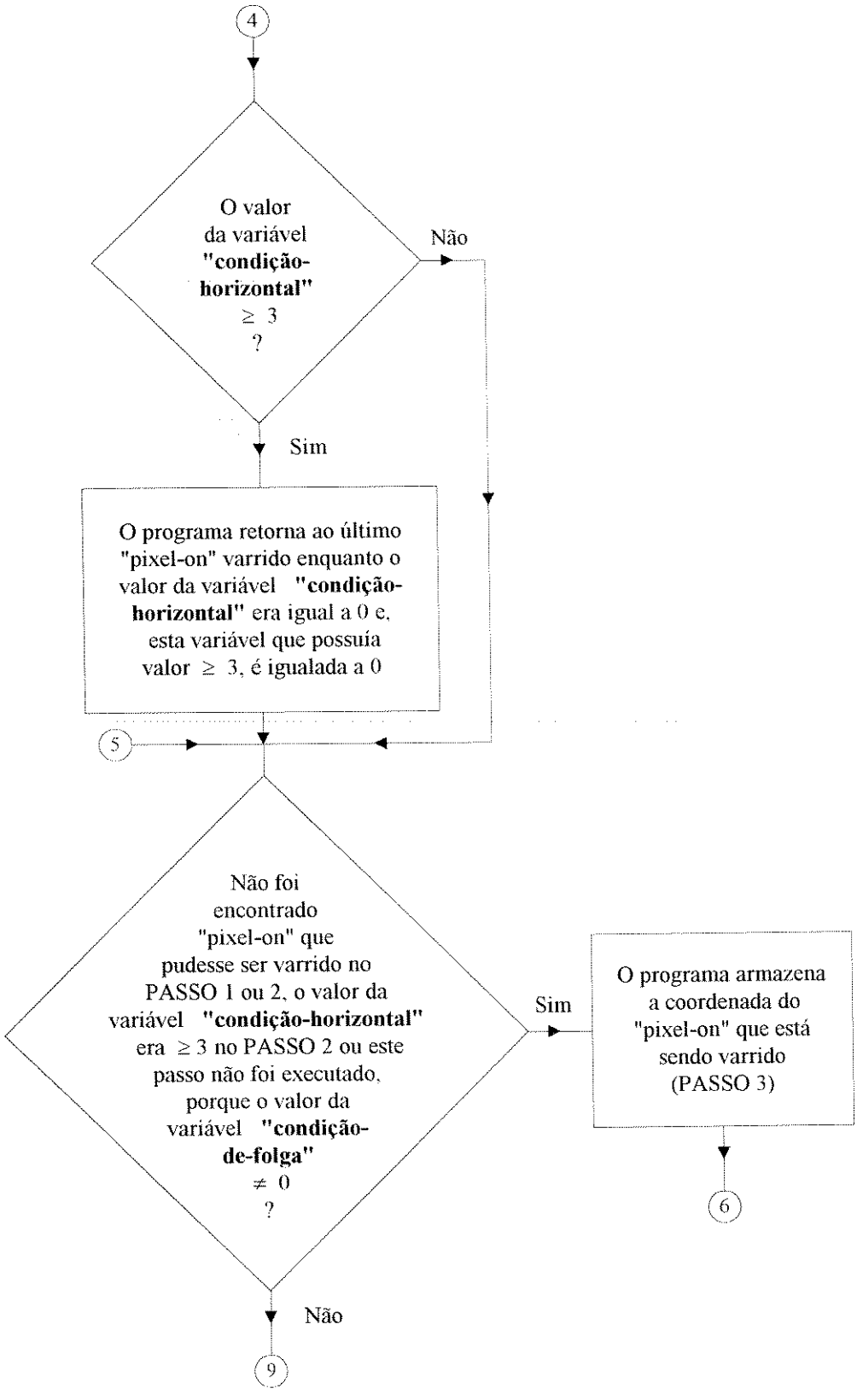


Figura 4.7(c) - Terceira parte do fluxograma que realiza a delimitação das coordenadas inicial e final da linha vertical esquerda.

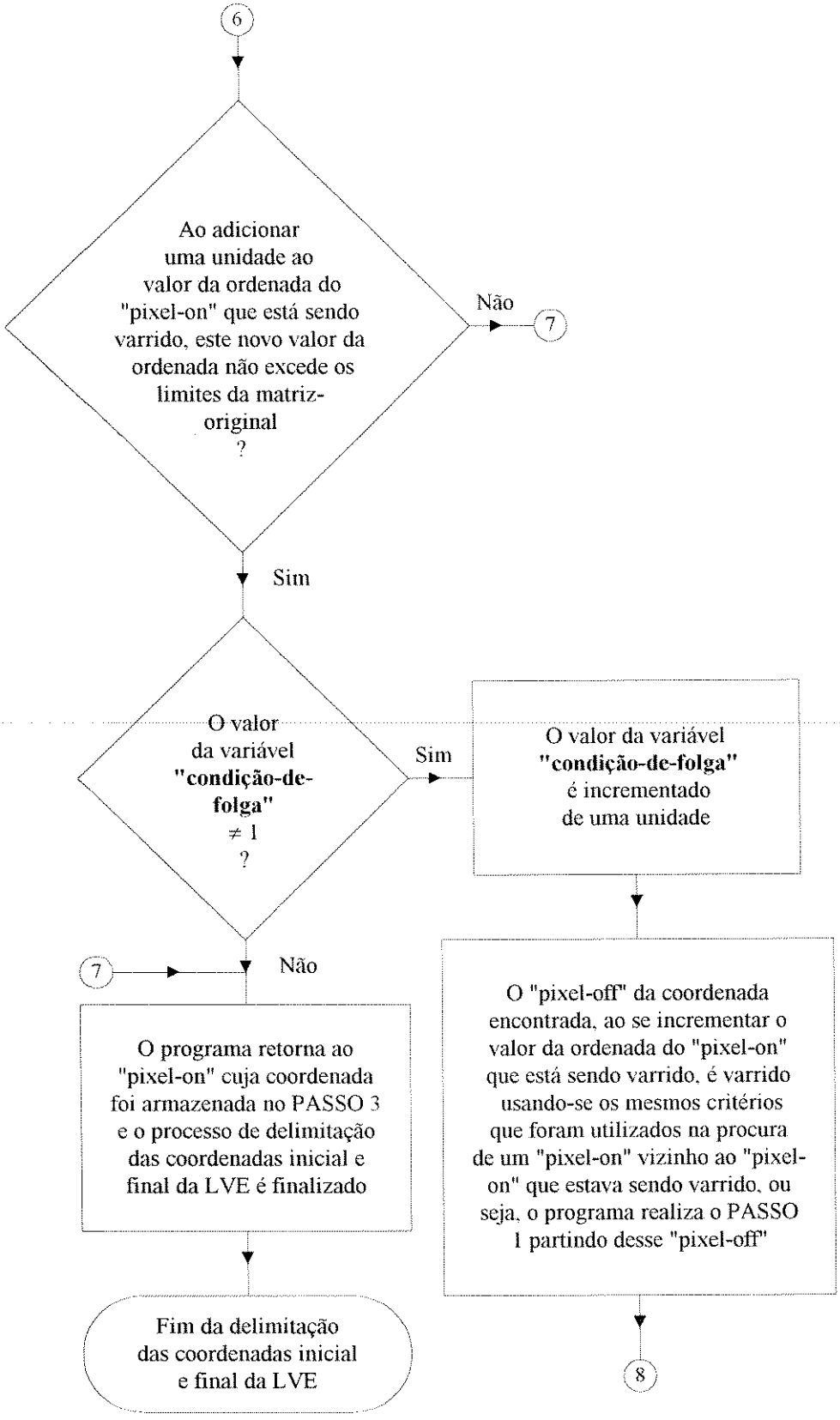


Figura 4.7(d) - Quarta parte do fluxograma que realiza a delimitação das coordenadas inicial e final da linha vertical esquerda.

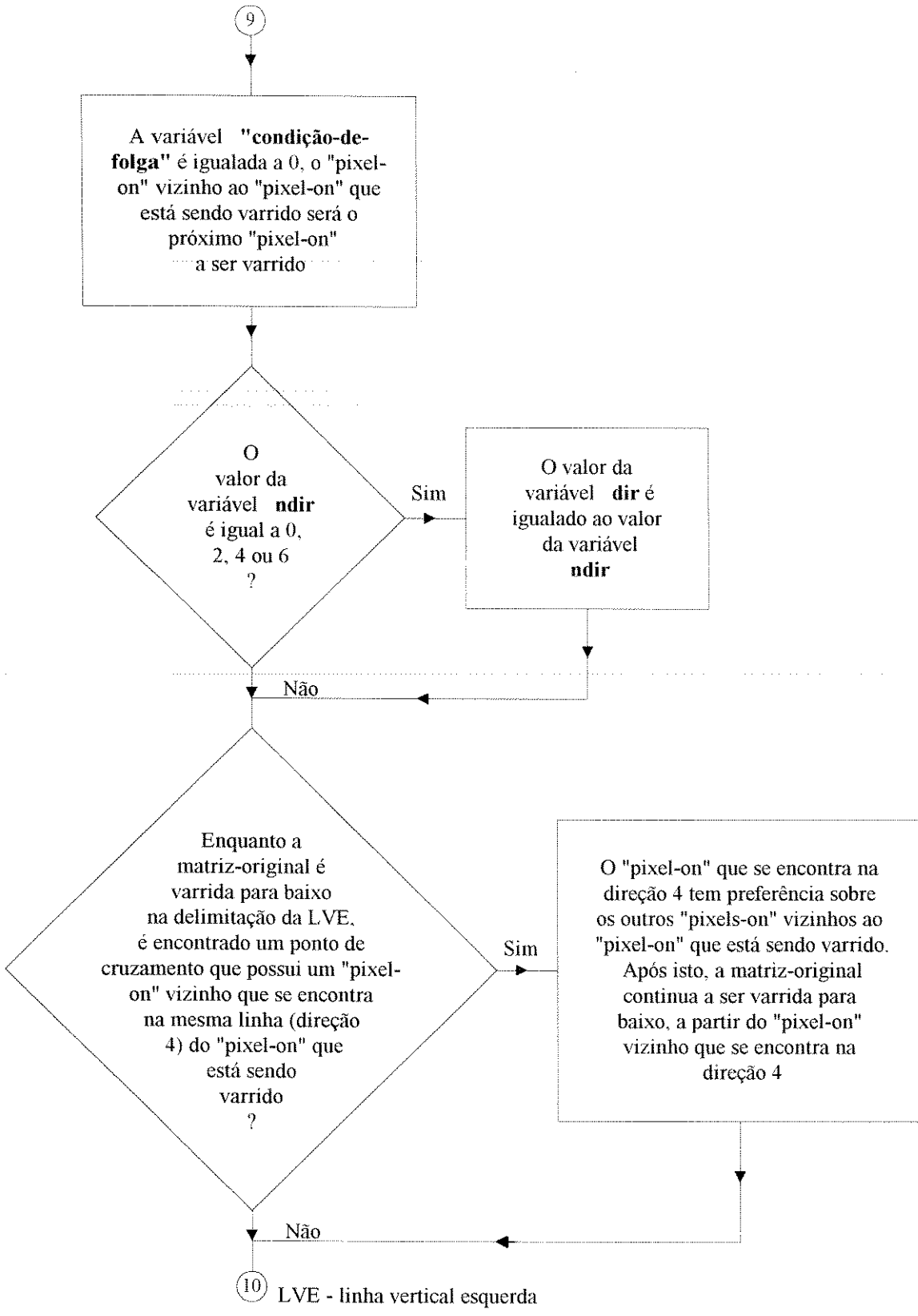
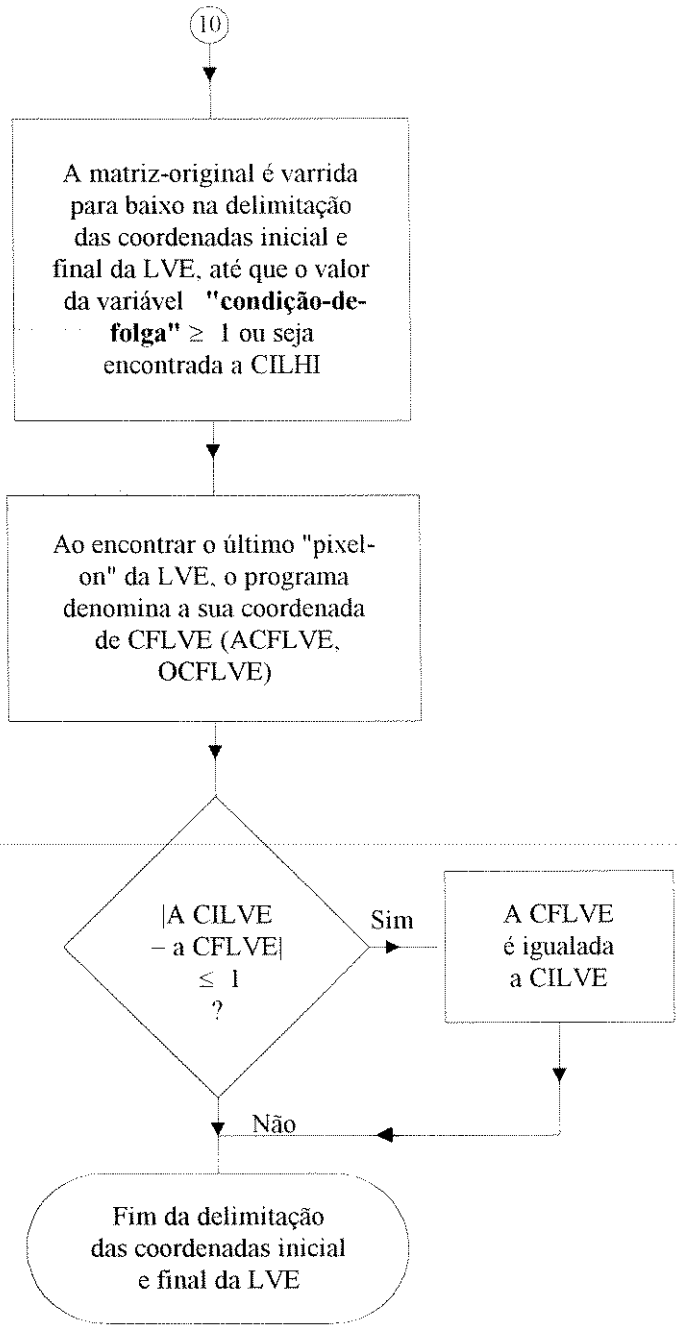


Figura 4.7(e) - Quinta parte do fluxograma que realiza a delimitação das coordenadas inicial e final da linha vertical esquerda.



LVE - linha vertical esquerda
CILVE - coordenada inicial da LVE
CFLVE - coordenada final da LVE
ACFLVE - abscissa da coordenada final da LVE
OCFLVE - ordenada da coordenada final da LVE
LHI - linha horizontal inferior
CILHI - coordenada inicial da LHI

Figura 4.7(f) - Sexta parte do fluxograma que realiza a delimitação das coordenadas inicial e final da linha vertical esquerda.

Fluxograma da função vertical

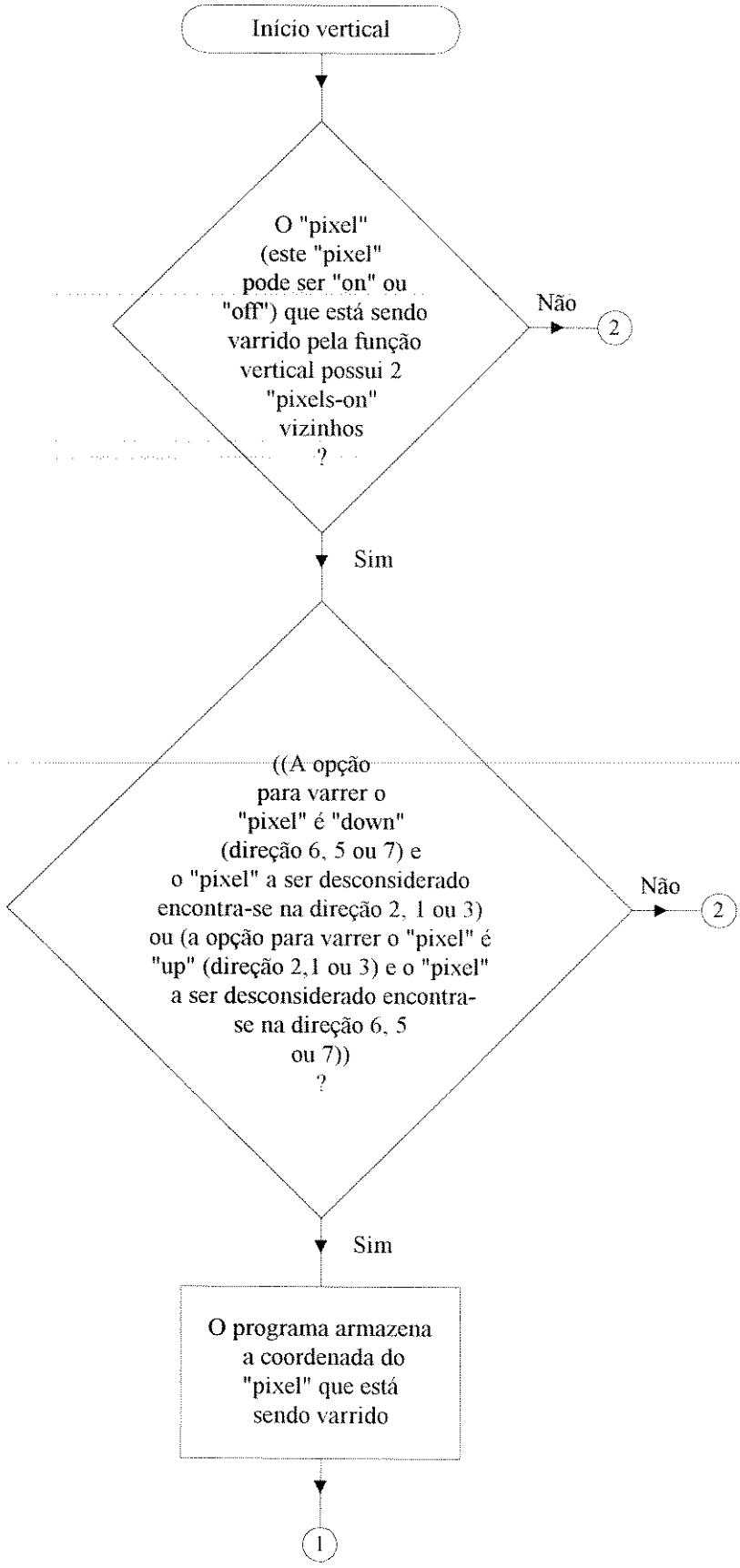


Figura 4.12(a) - Primeira parte do fluxograma da função vertical.

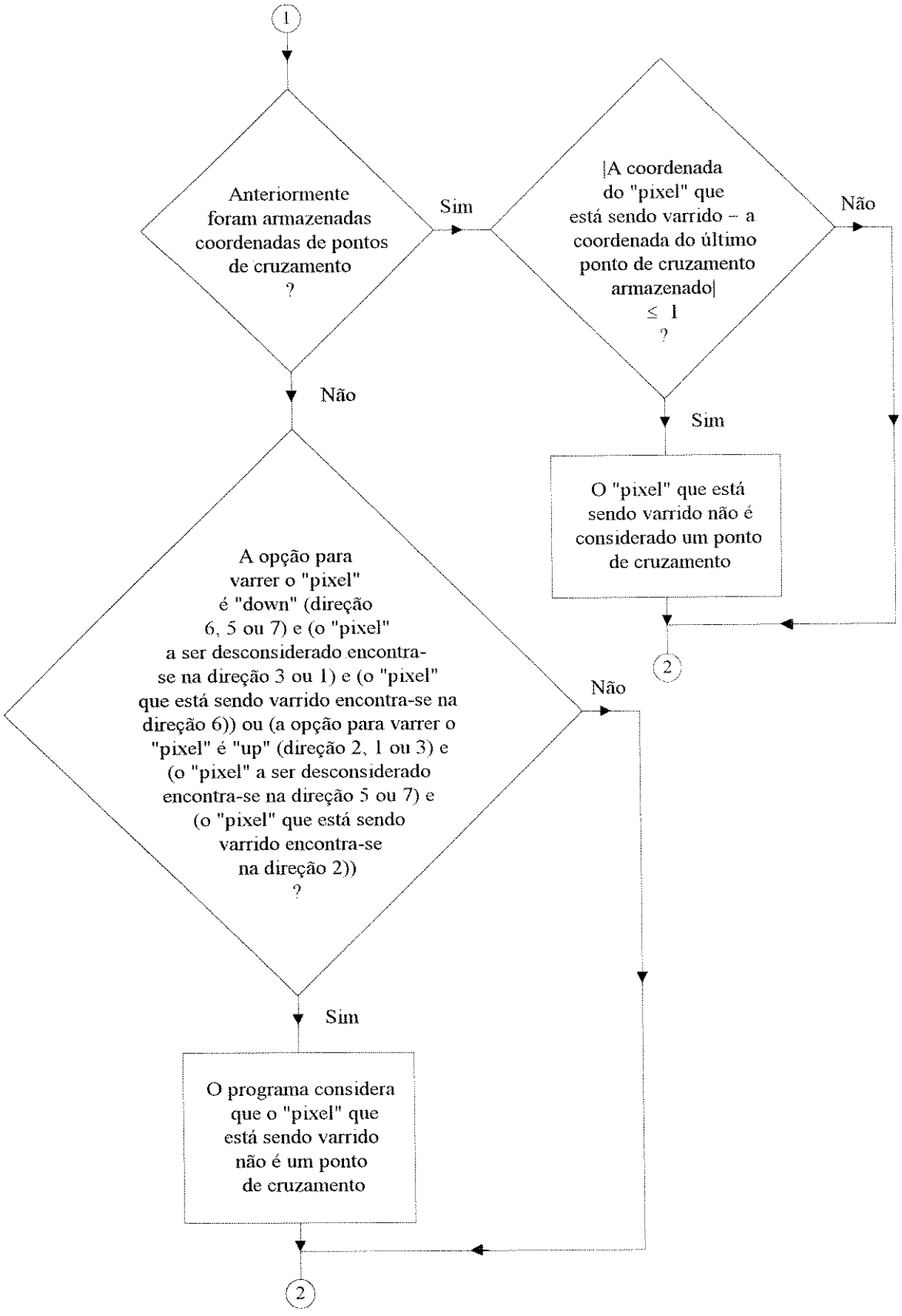


Figura 4.12(b) - Segunda parte do fluxograma da função vertical.

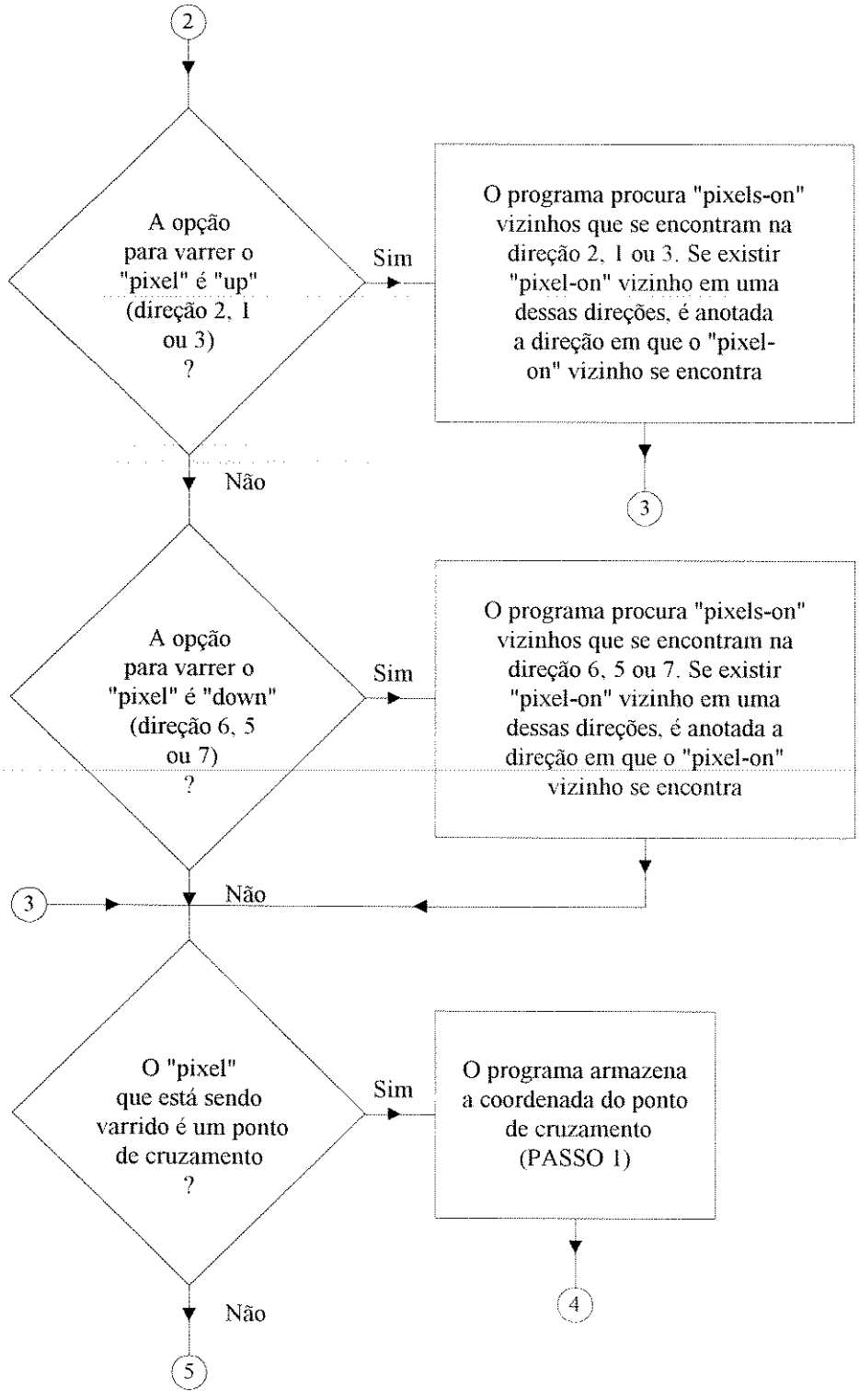


Figura 4.12(c) - Terceira parte do fluxograma da função vertical.

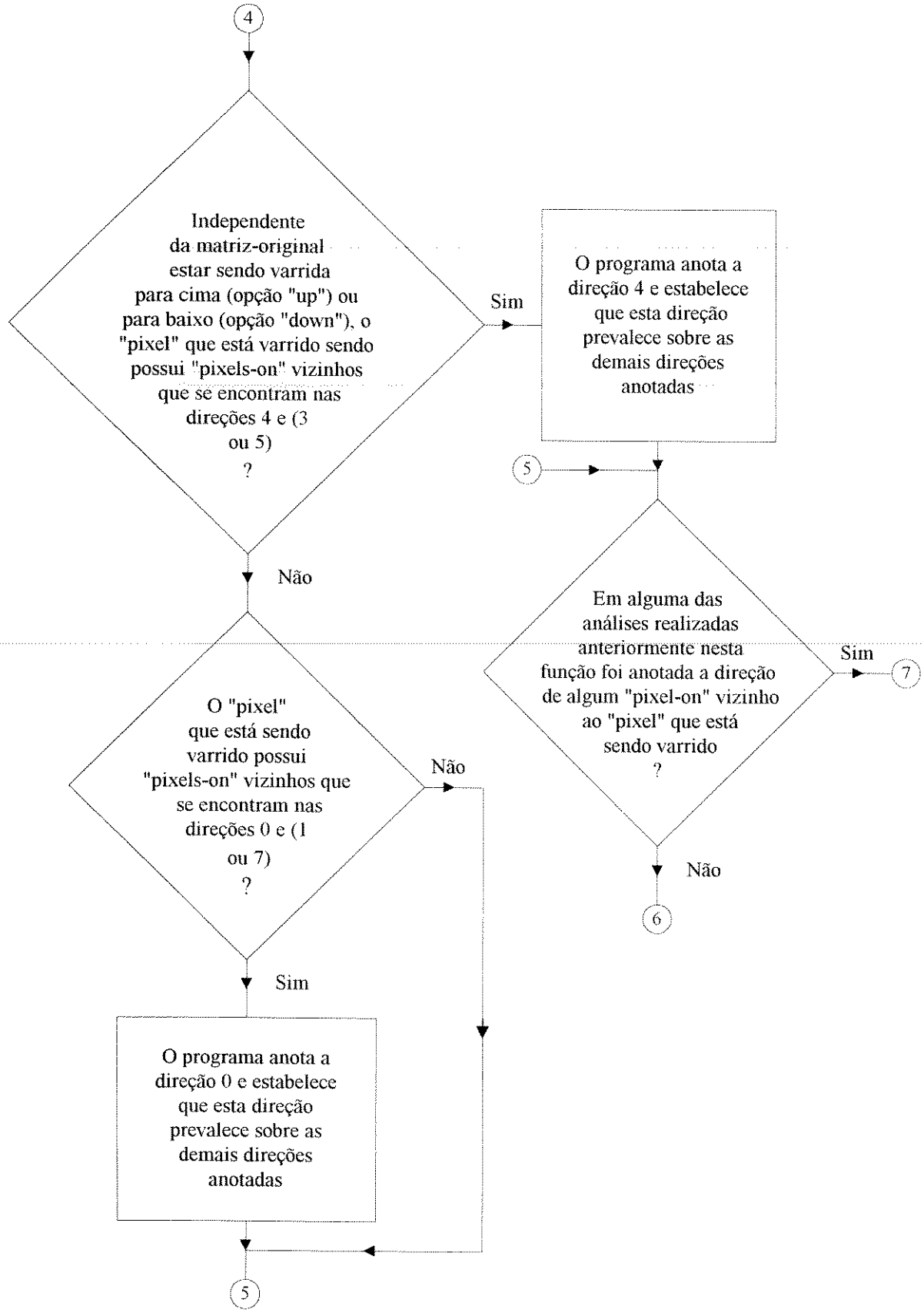


Figura 4.12(d) - Quarta parte do fluxograma da função vertical.

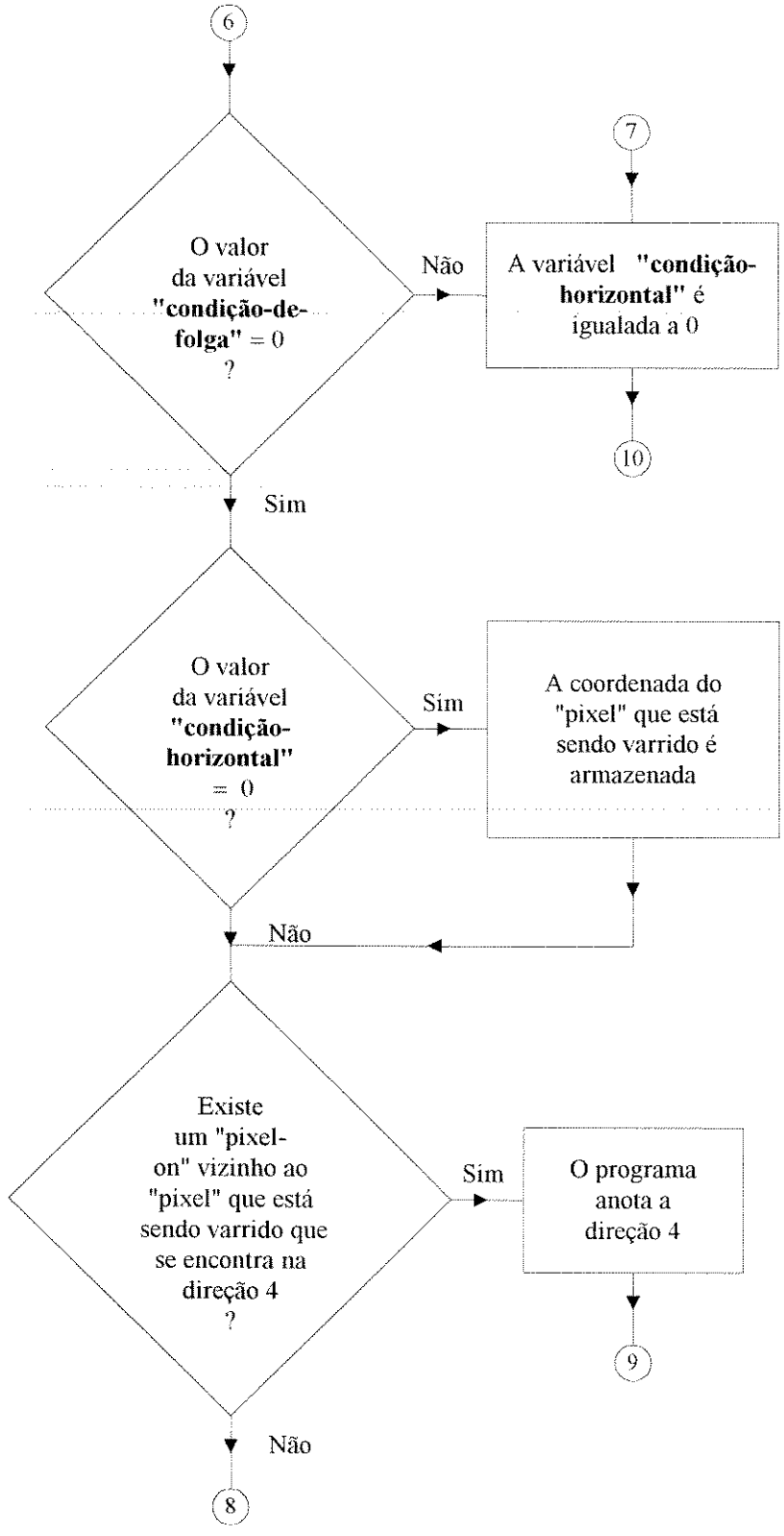


Figura 4.12(e) - Quinta parte do fluxograma da função vertical.

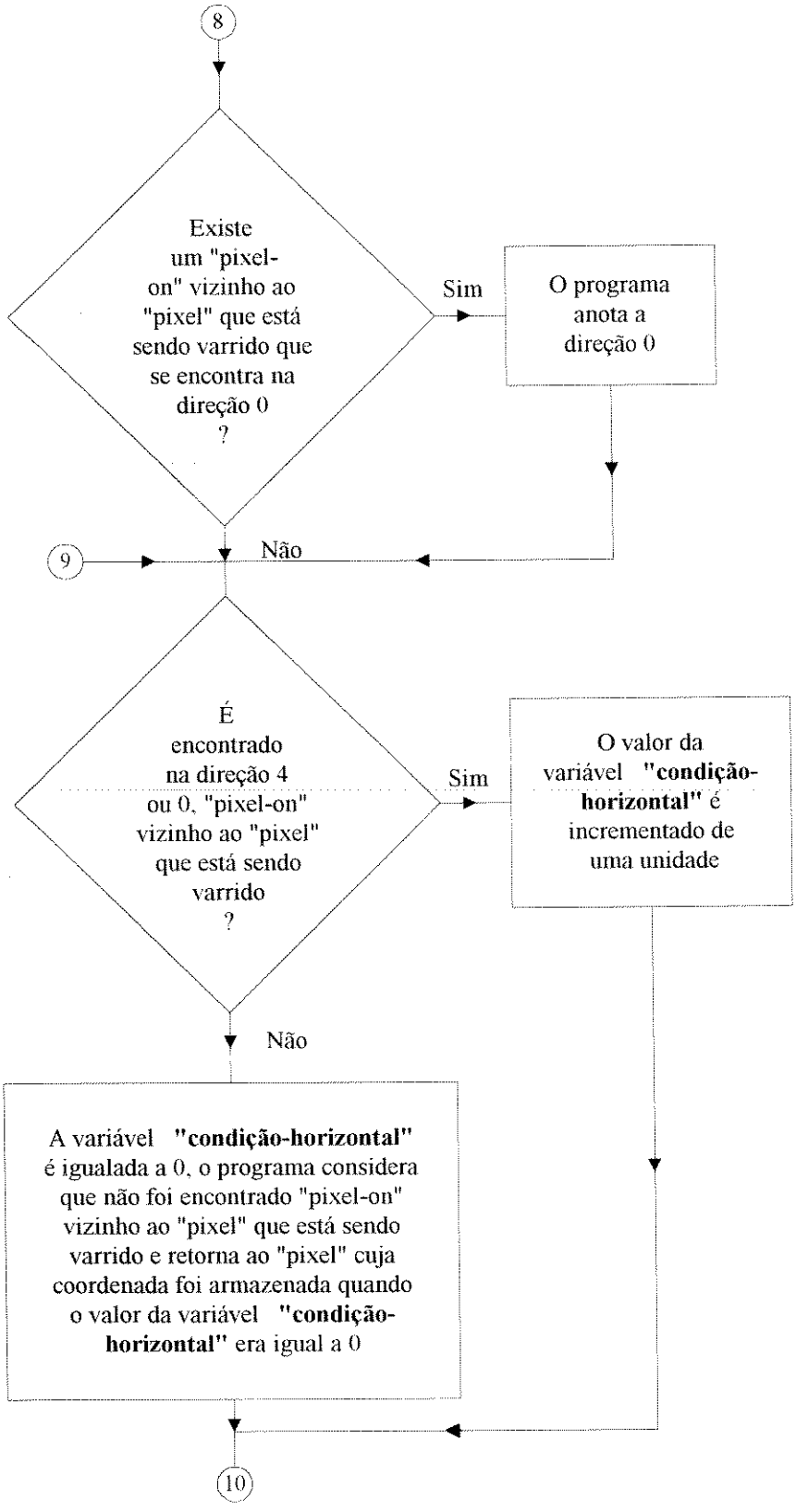


Figura 4.12(f) - Sexta parte do fluxograma da função vertical.

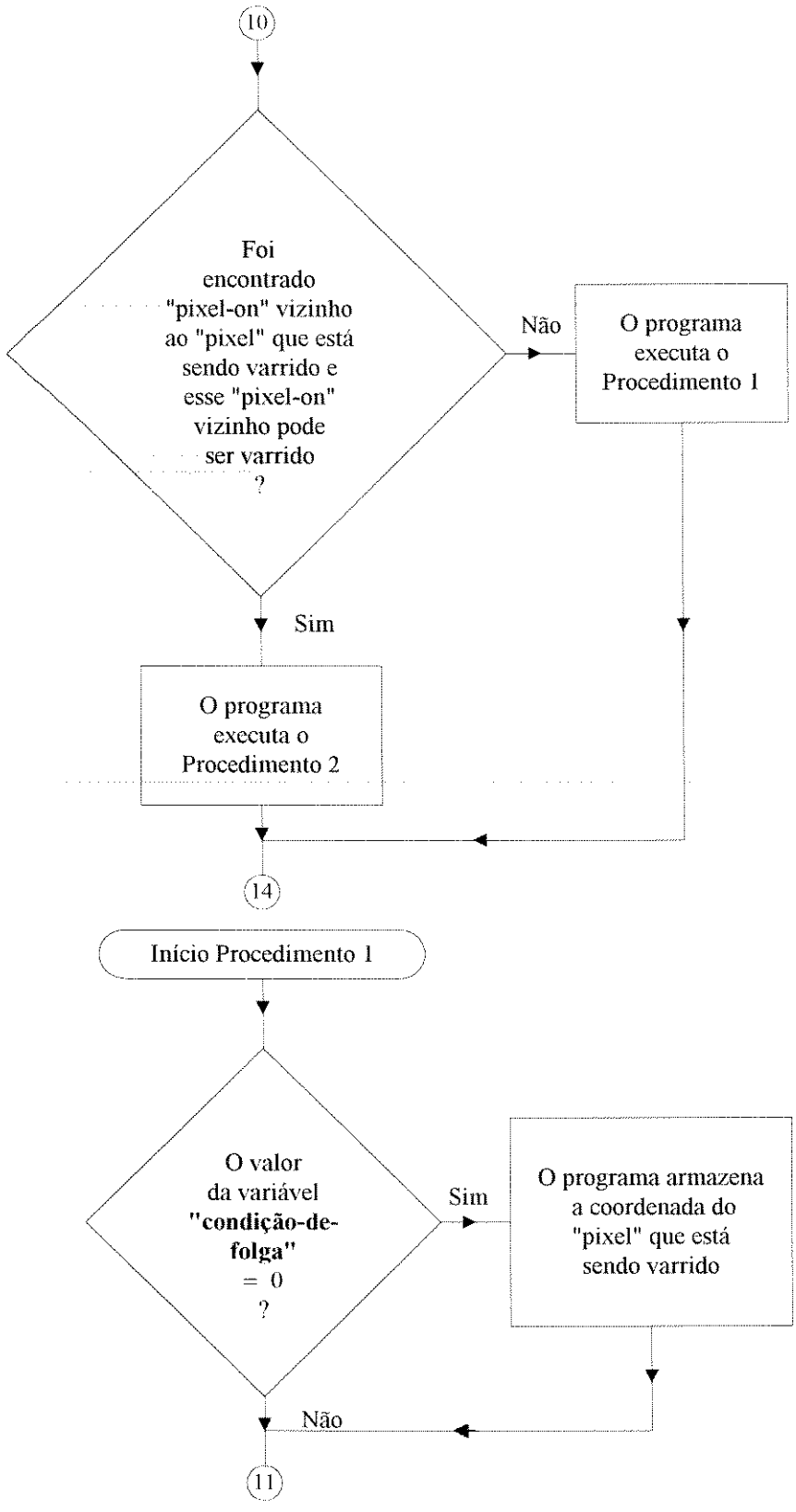


Figura 4.12(g) - Sétima parte do fluxograma da função vertical.

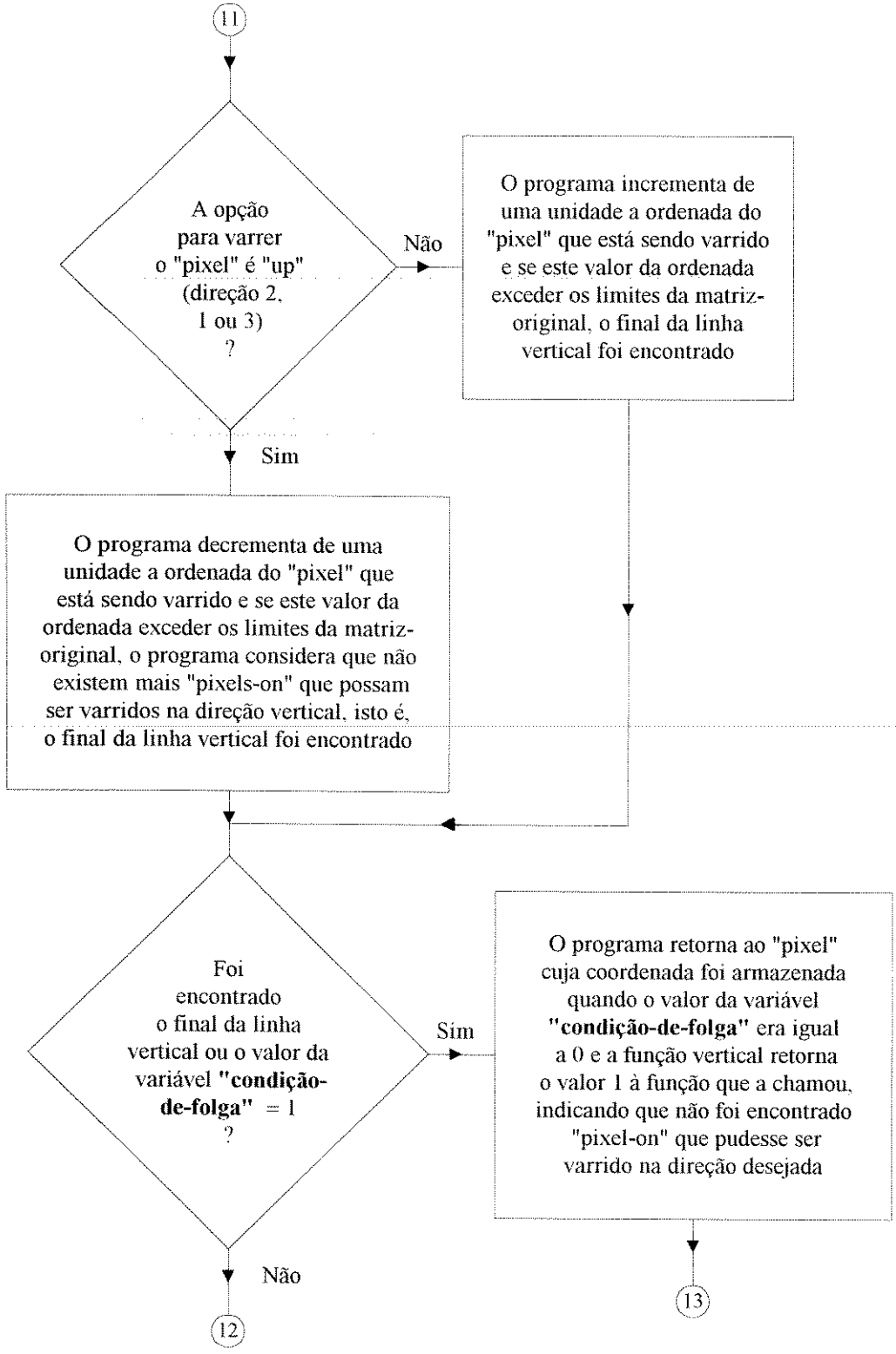


Figura 4.12(h) - Oitava parte do fluxograma da função vertical.

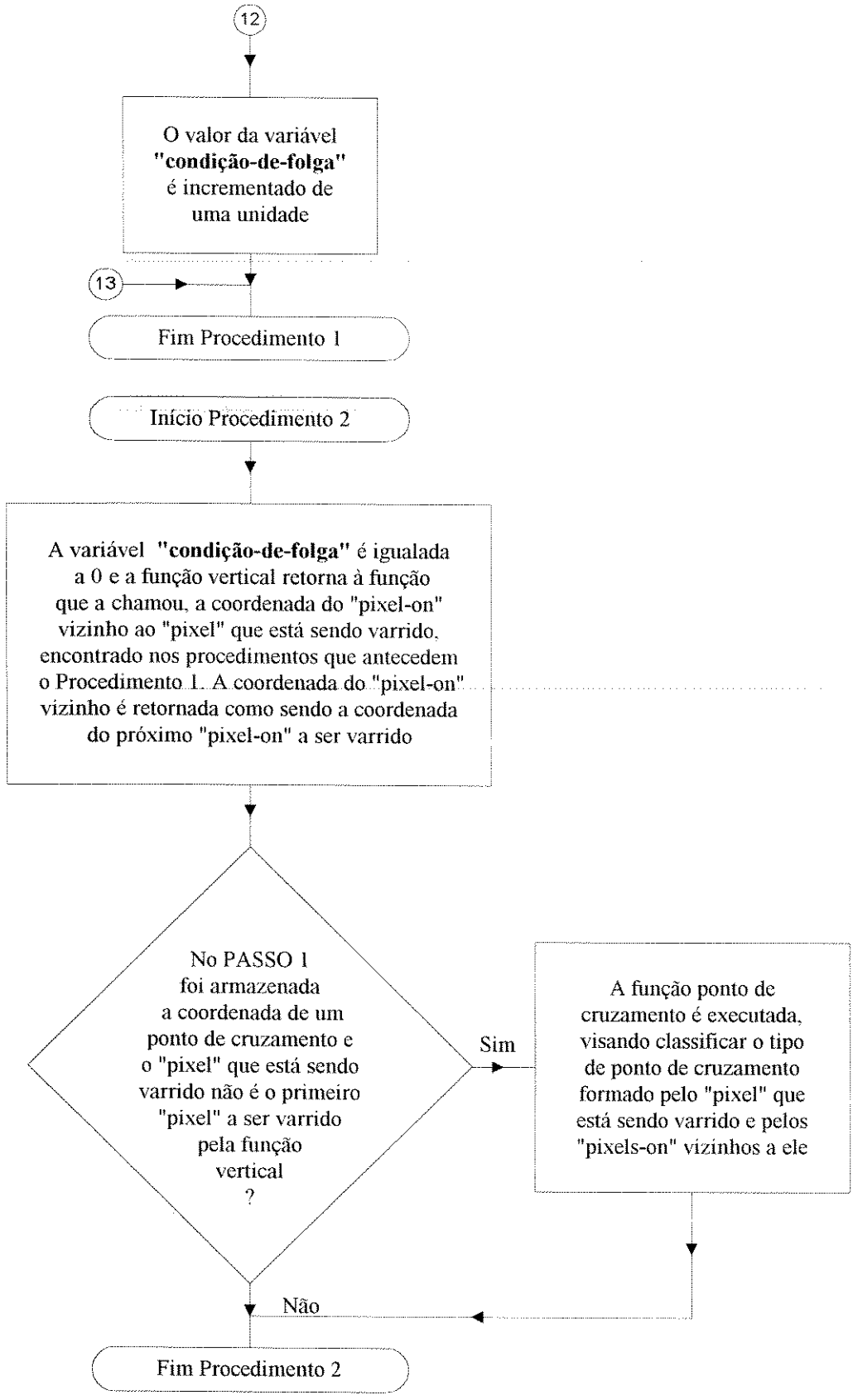


Figura 4.12(i) - Nona parte do fluxograma da função vertical.

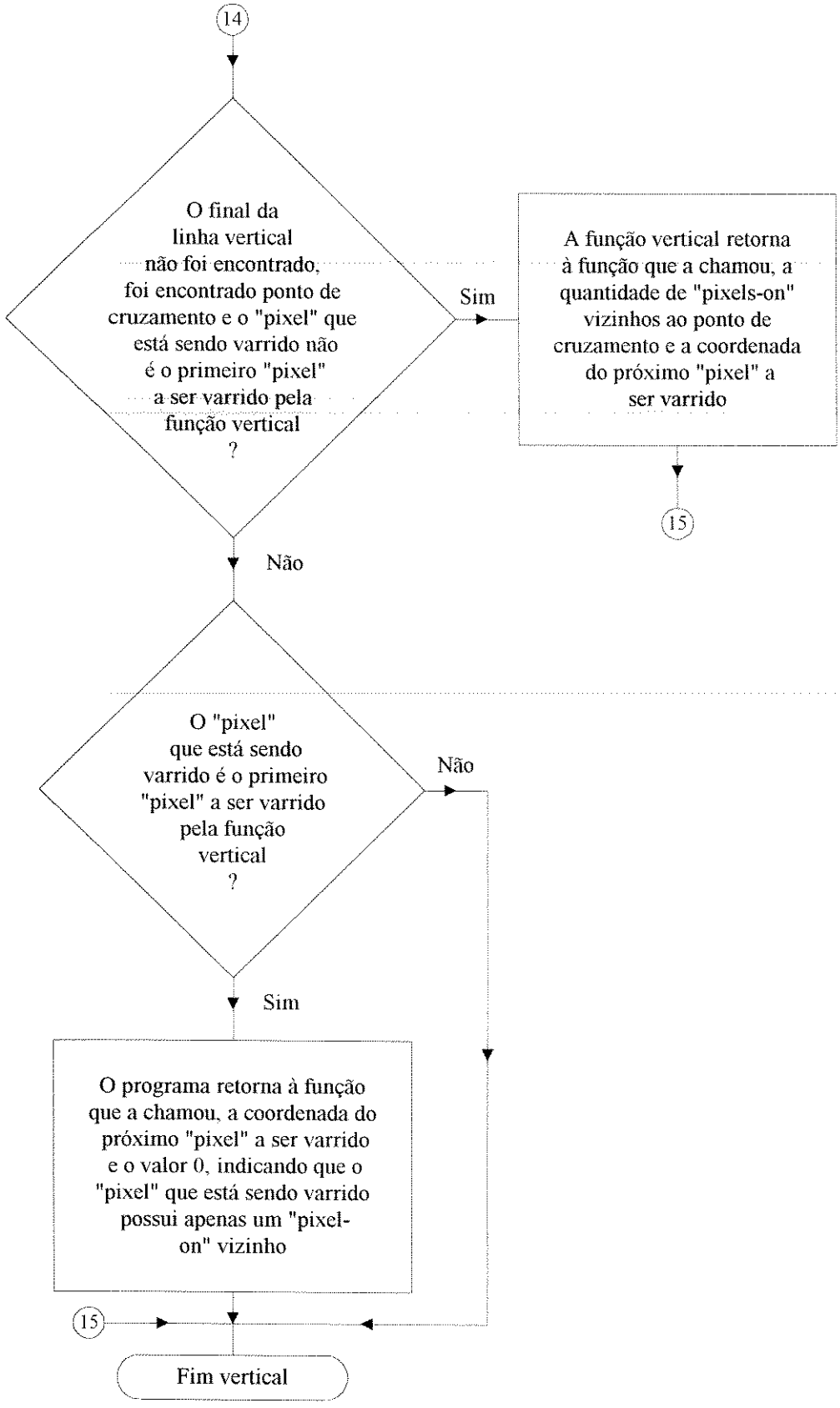


Figura 4.12(j) - Décima parte do fluxograma da função vertical.

Fluxograma da função horizontal

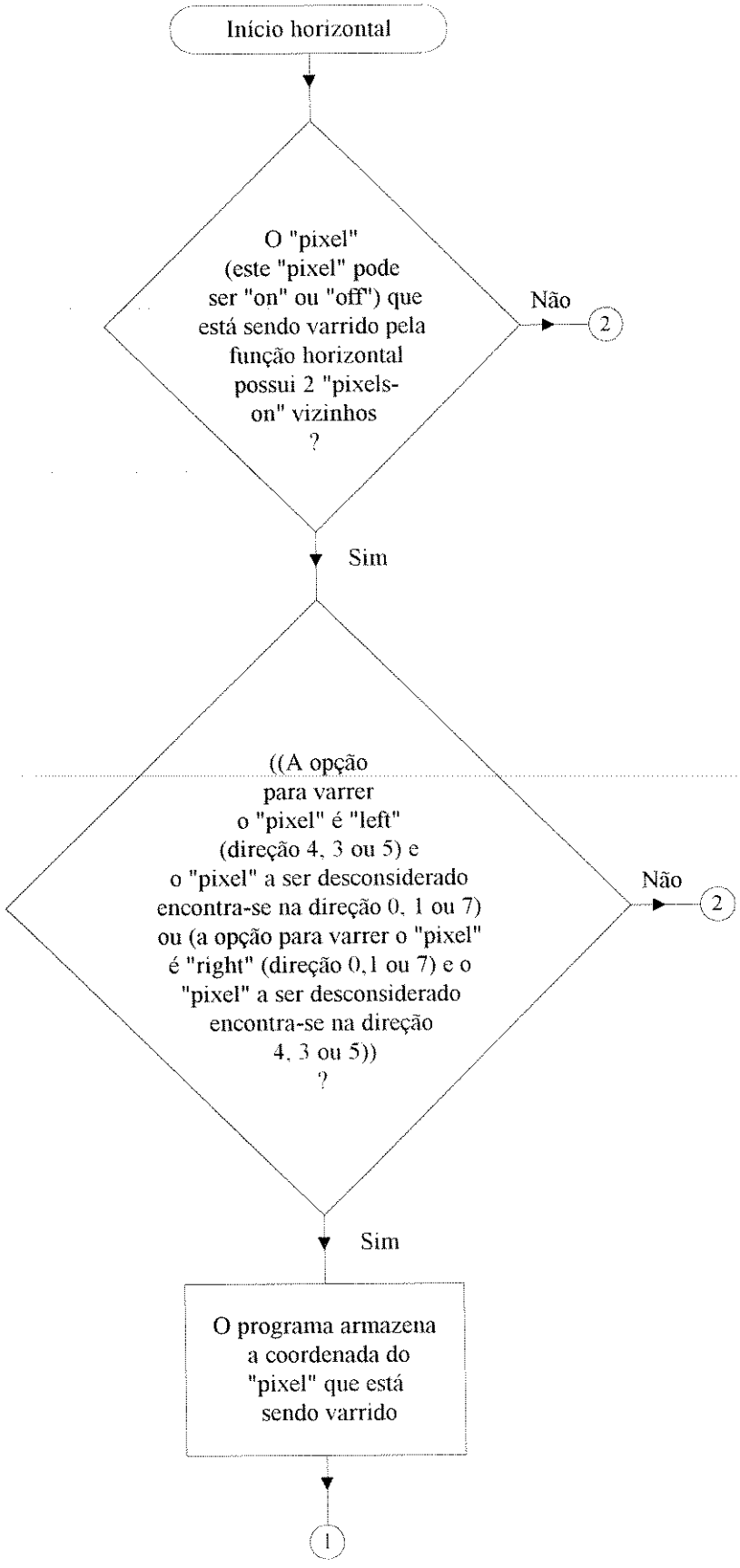


Figura 4.14(a) - Primeira parte do fluxograma da função horizontal.

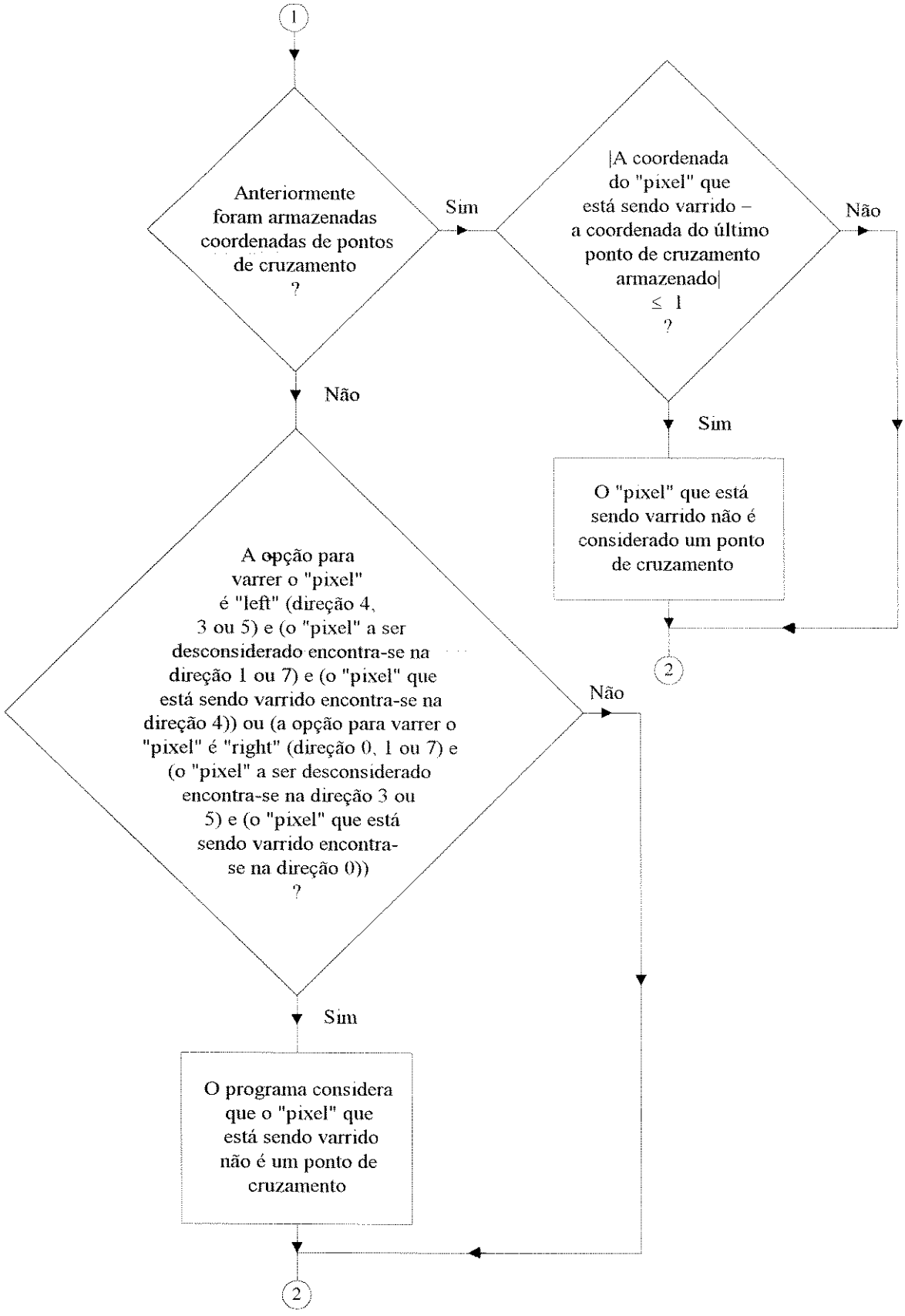


Figura 4.14(b) - Segunda parte do fluxograma da função horizontal.

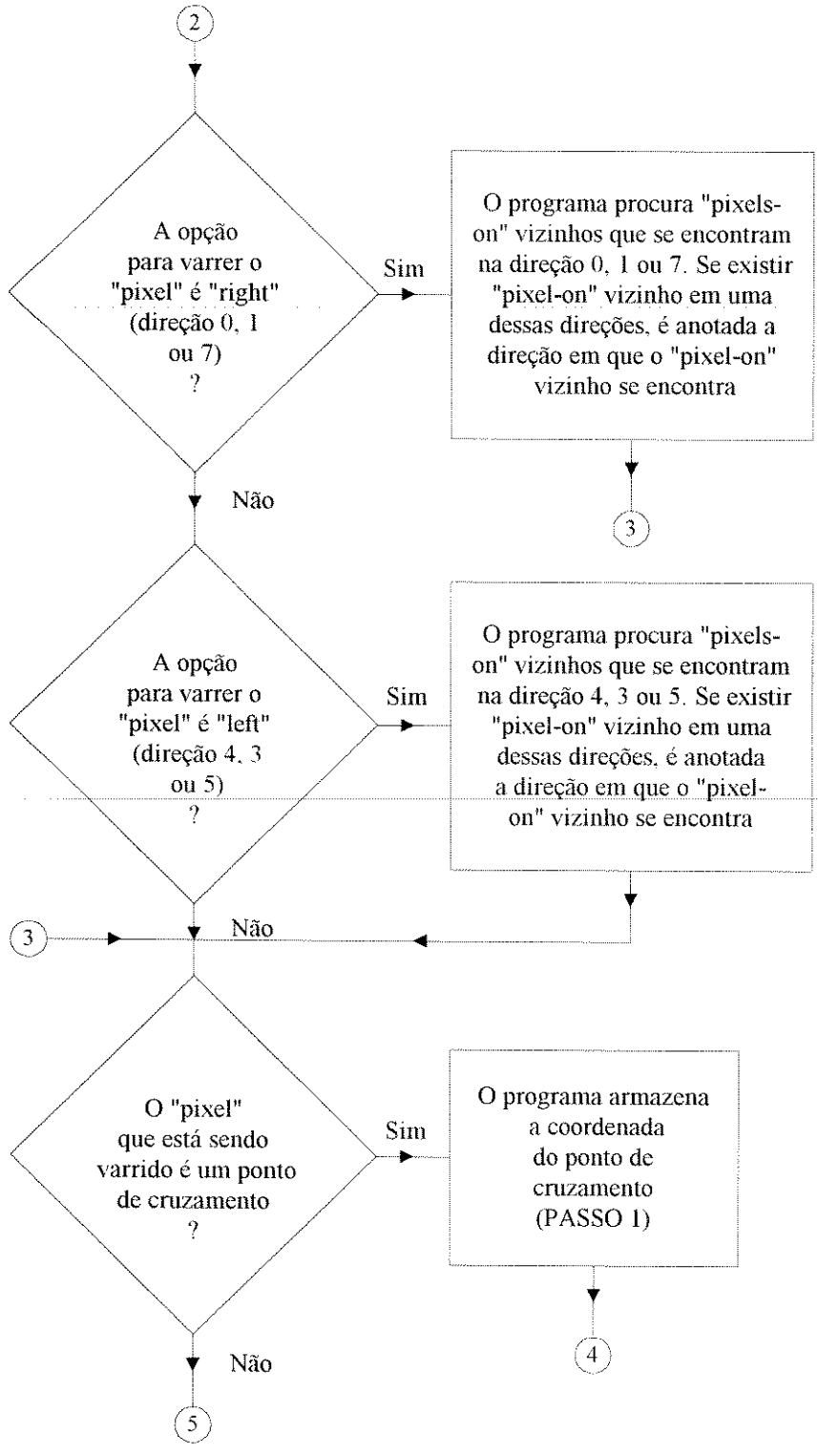


Figura 4.14(c) - Terceira parte do fluxograma da função horizontal.

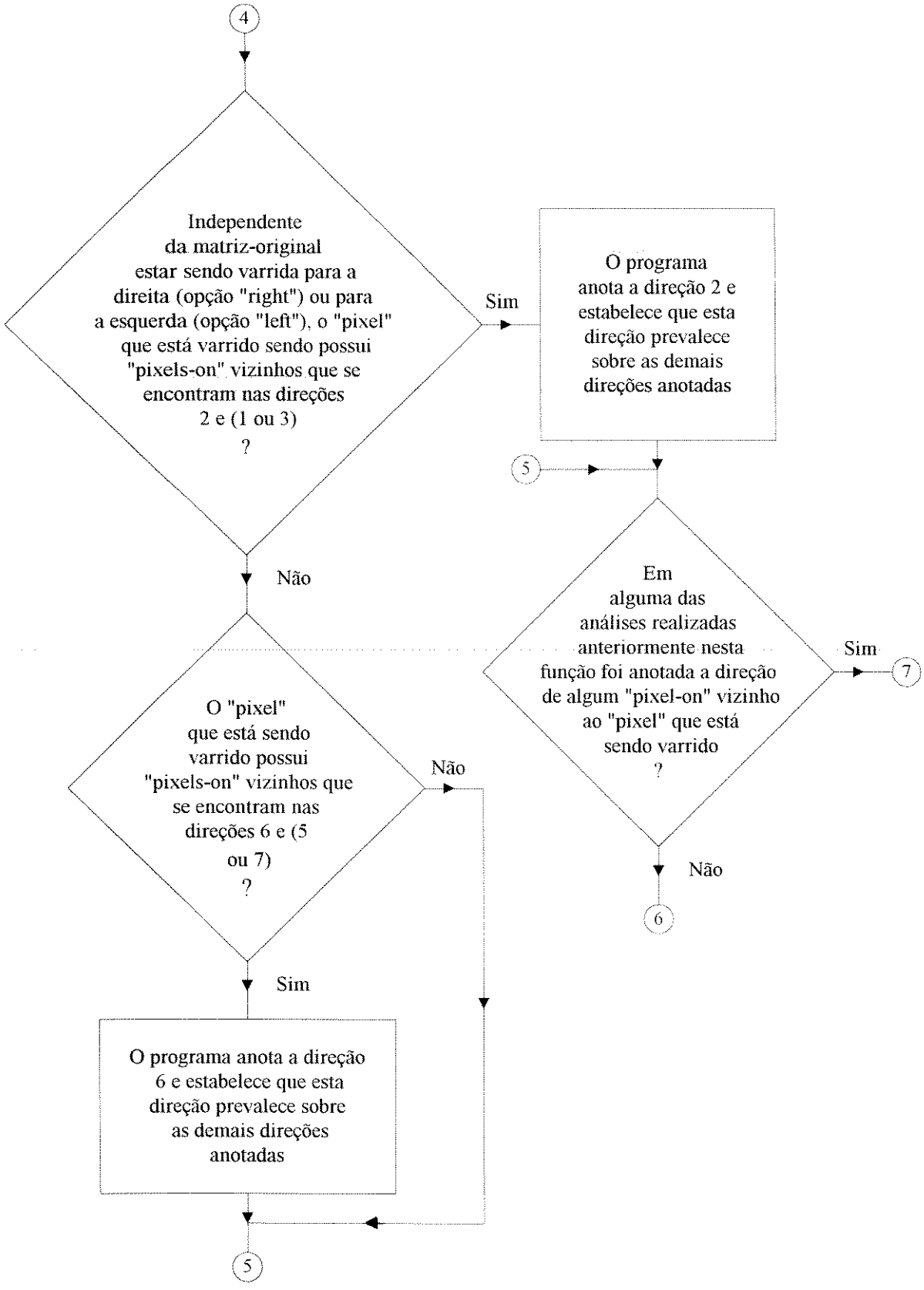


Figura 4.14(d) - Quarta parte do fluxograma da função horizontal.

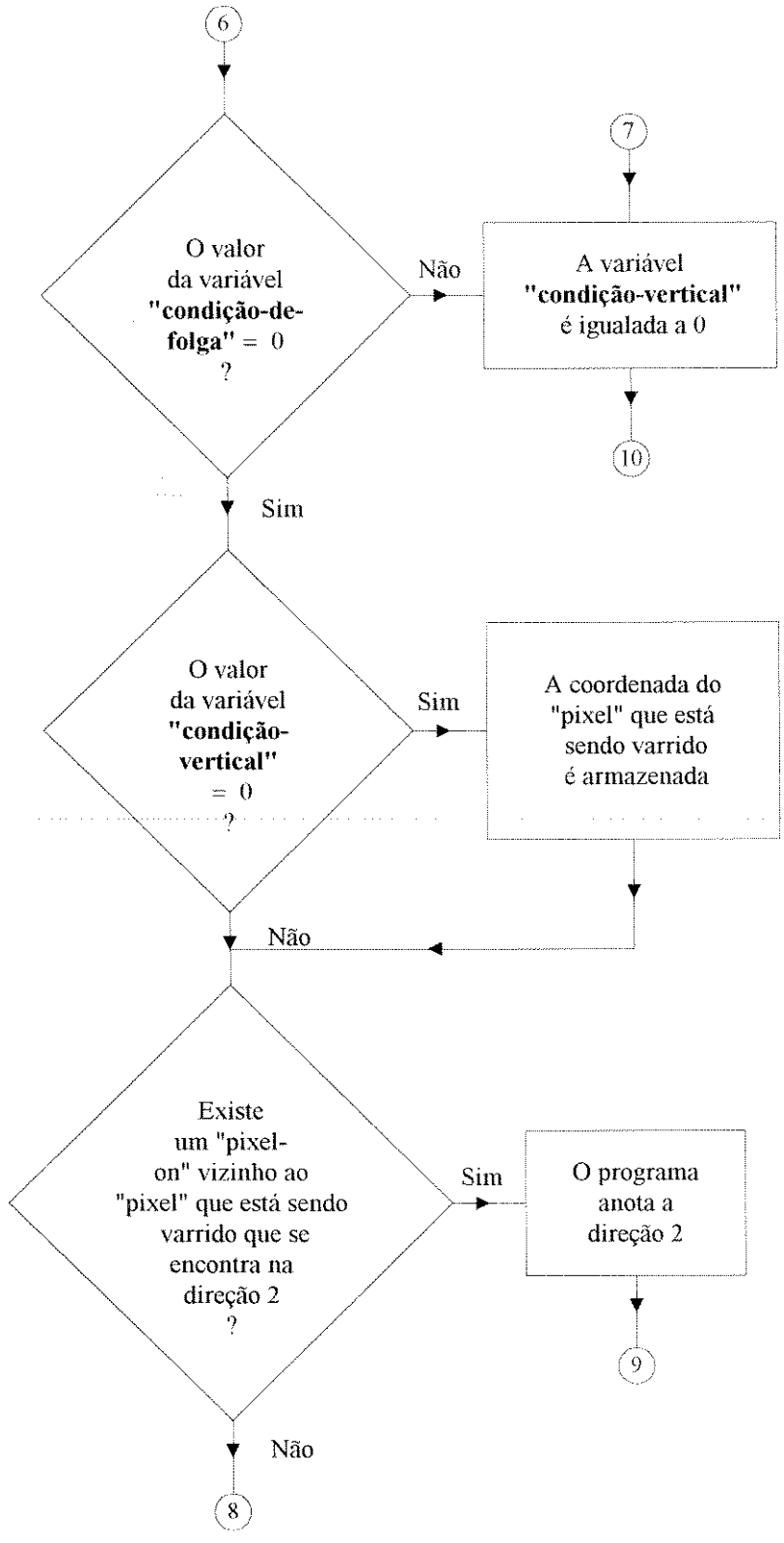


Figura 4.14(e) - Quinta parte do fluxograma da função horizontal.

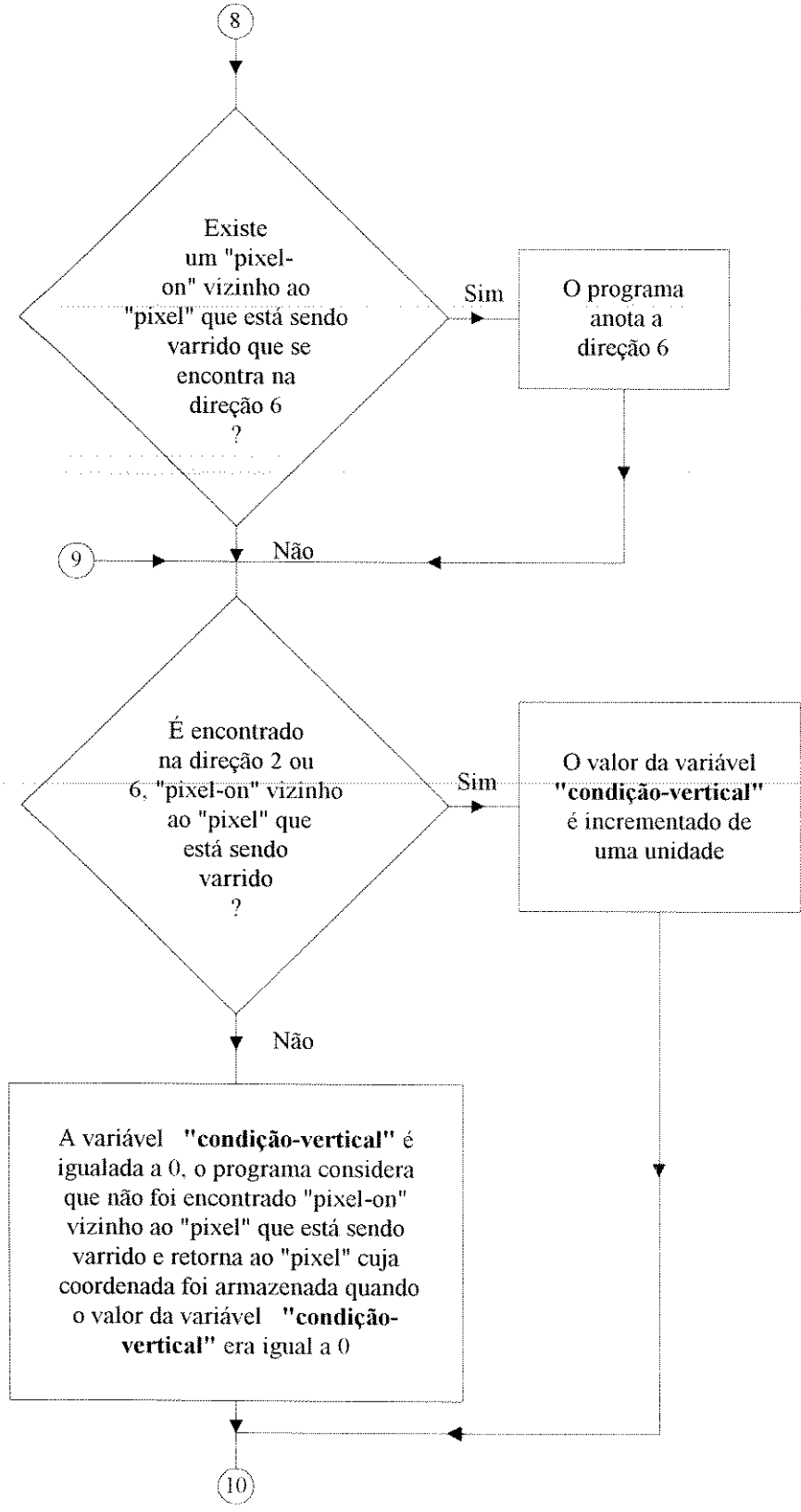


Figura 4.14(f) - Sexta parte do fluxograma da função horizontal.

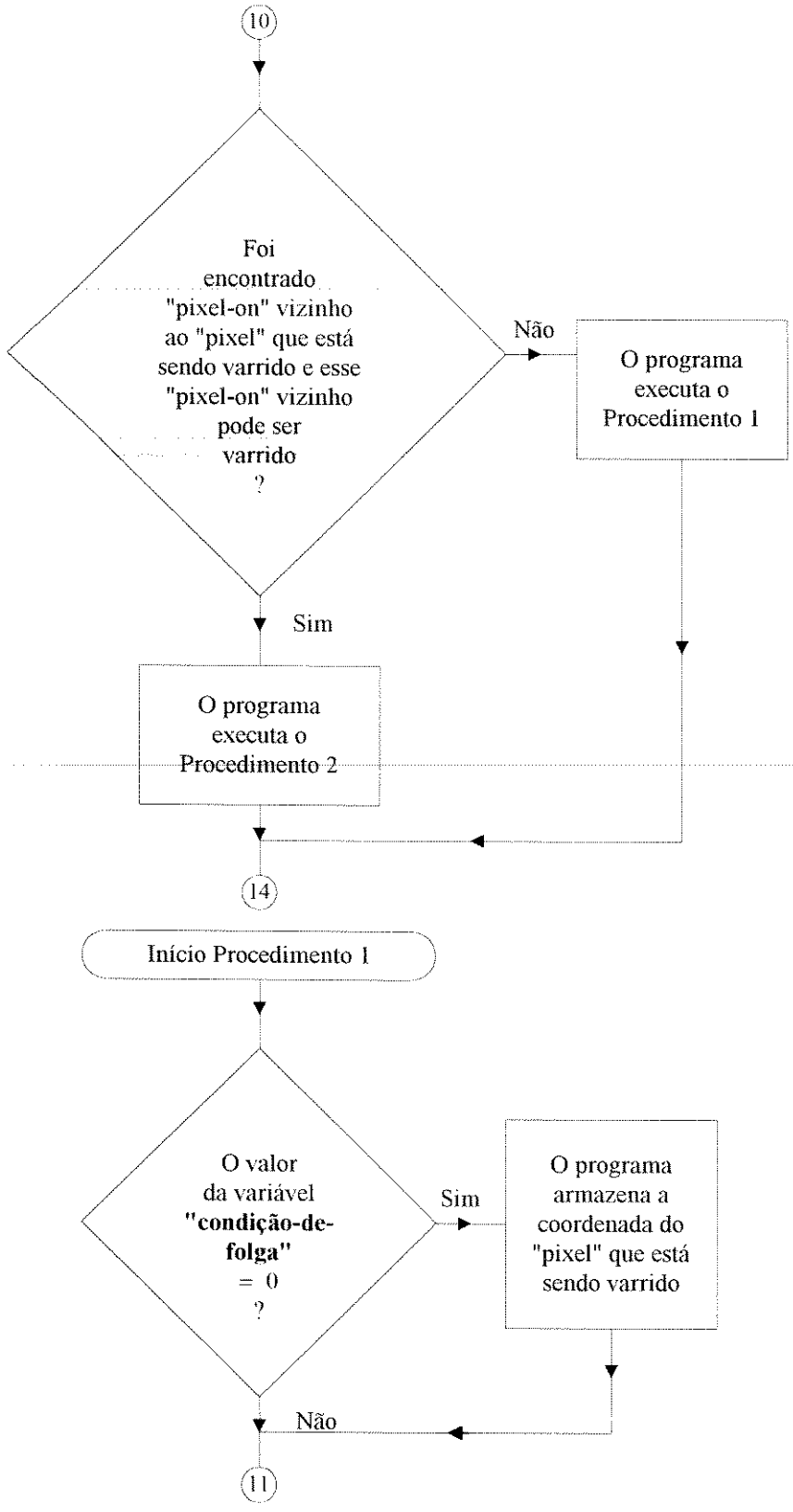


Figura 4.14(g) - Sétima parte do fluxograma da função horizontal.

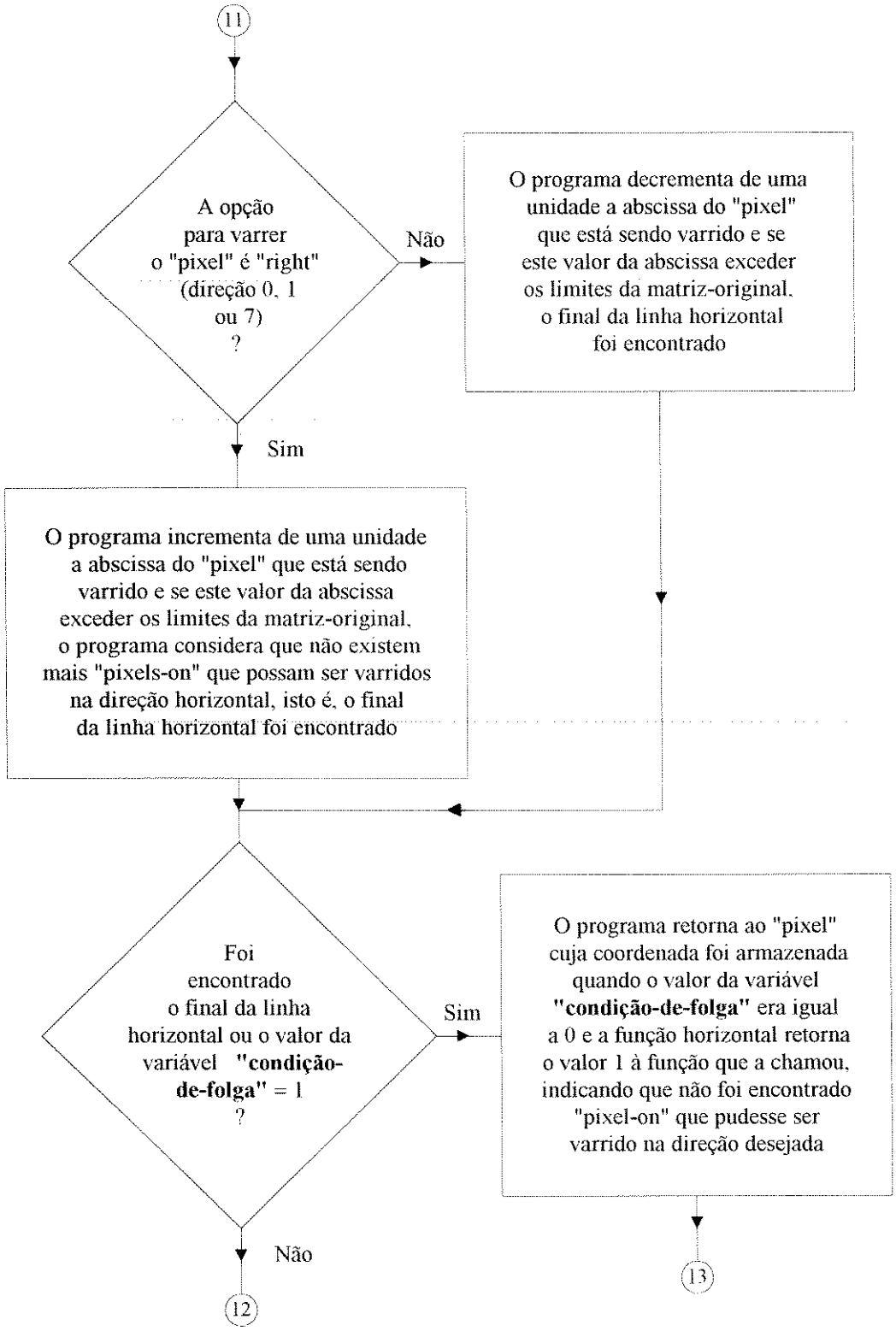


Figura 4.14(h) - Oitava parte do fluxograma da função horizontal.

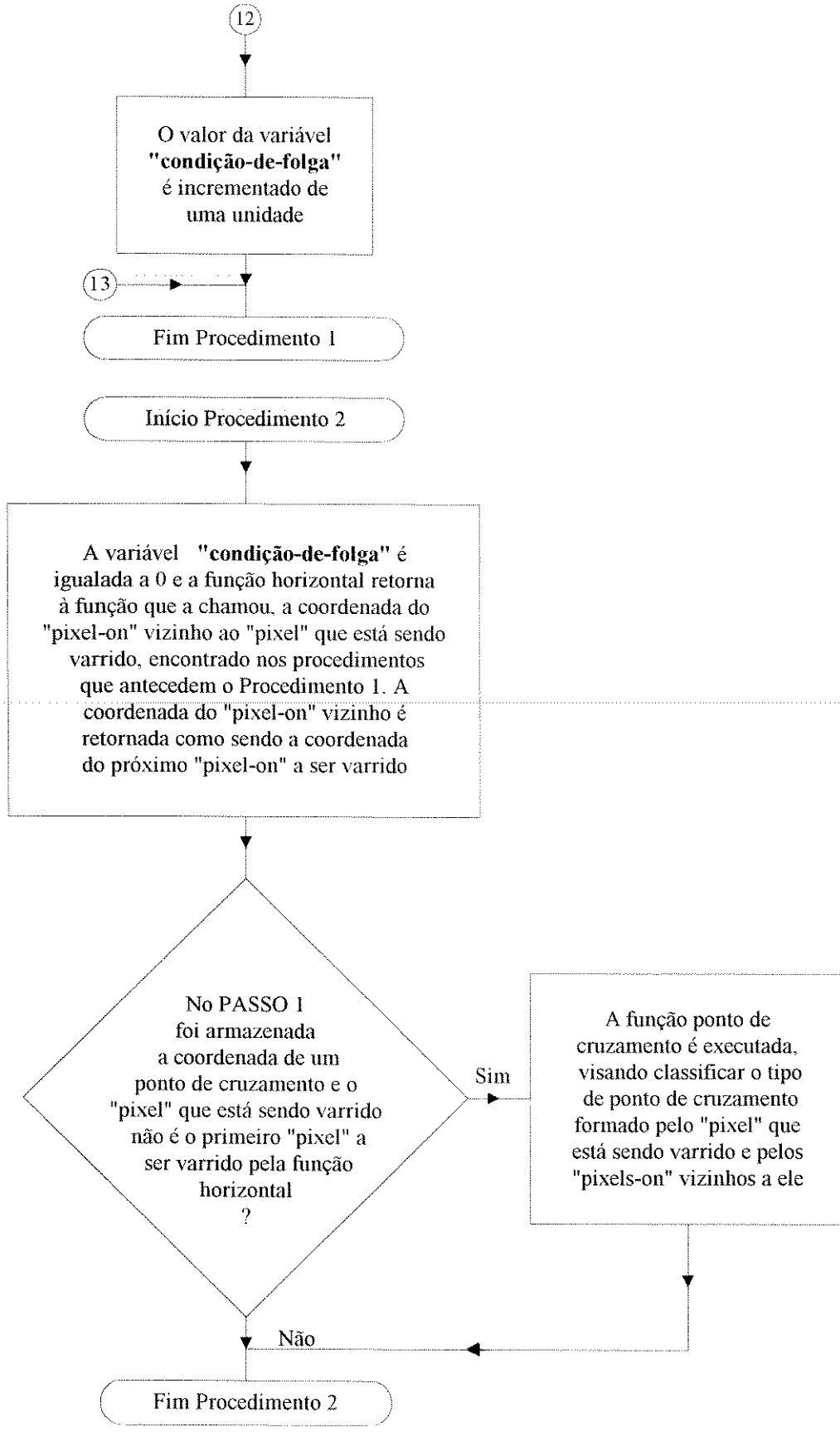


Figura 4.14(i) - Nona parte do fluxograma da função horizontal.

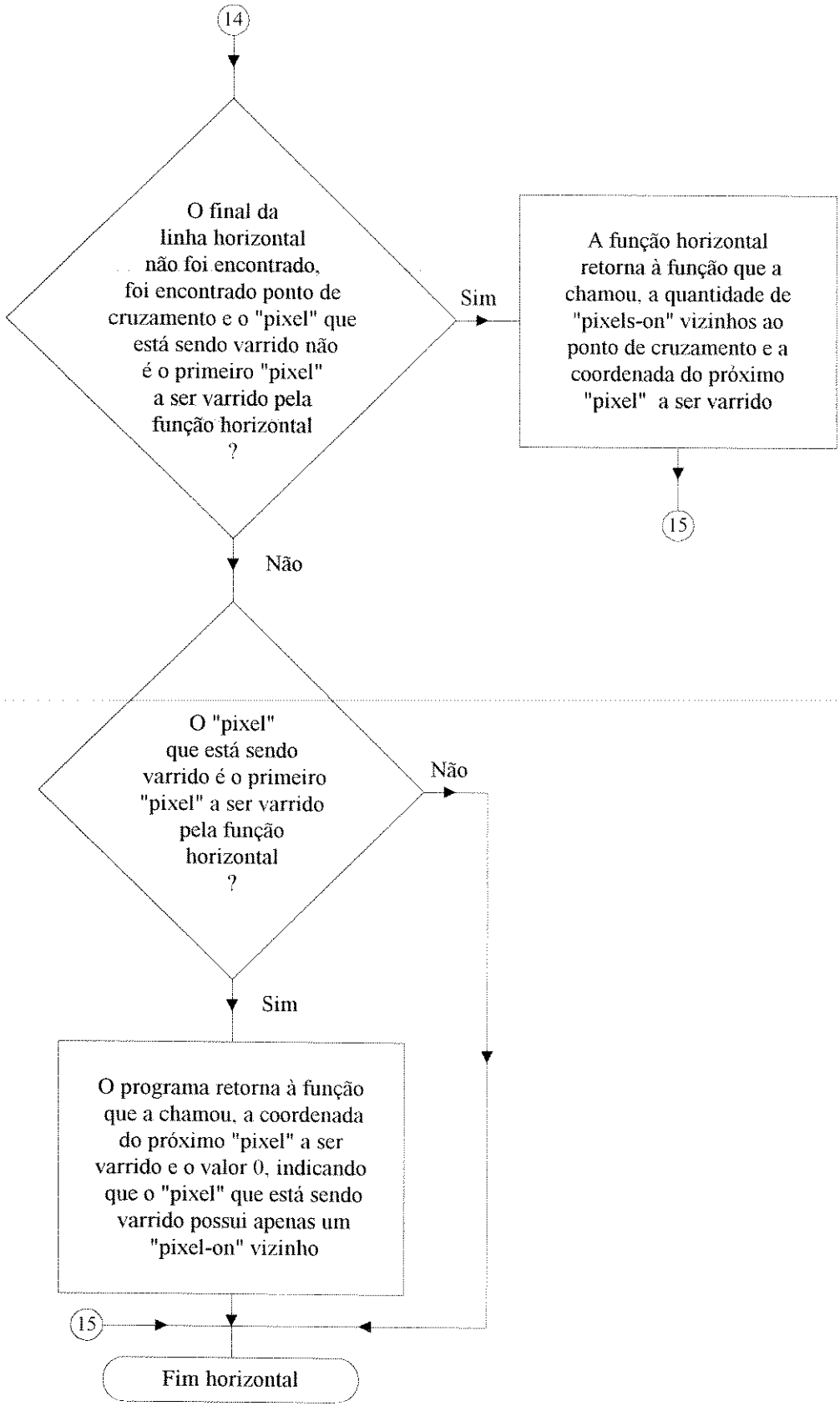


Figura 4.14(j) - Décima parte do fluxograma da função horizontal.

Fluxograma de todo o processo utilizado no reconhecimento de caracteres numéricos manuscritos isolados

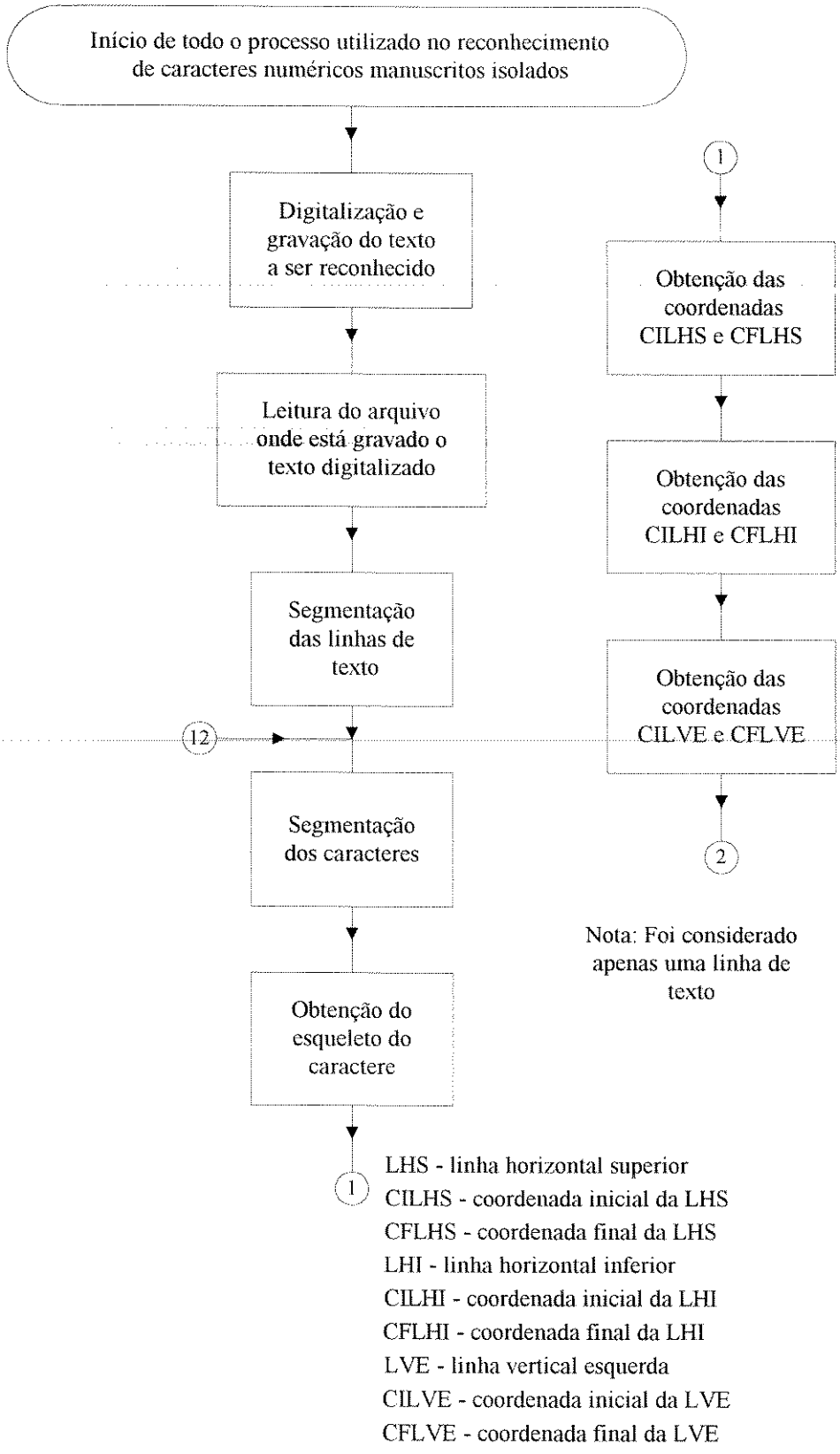
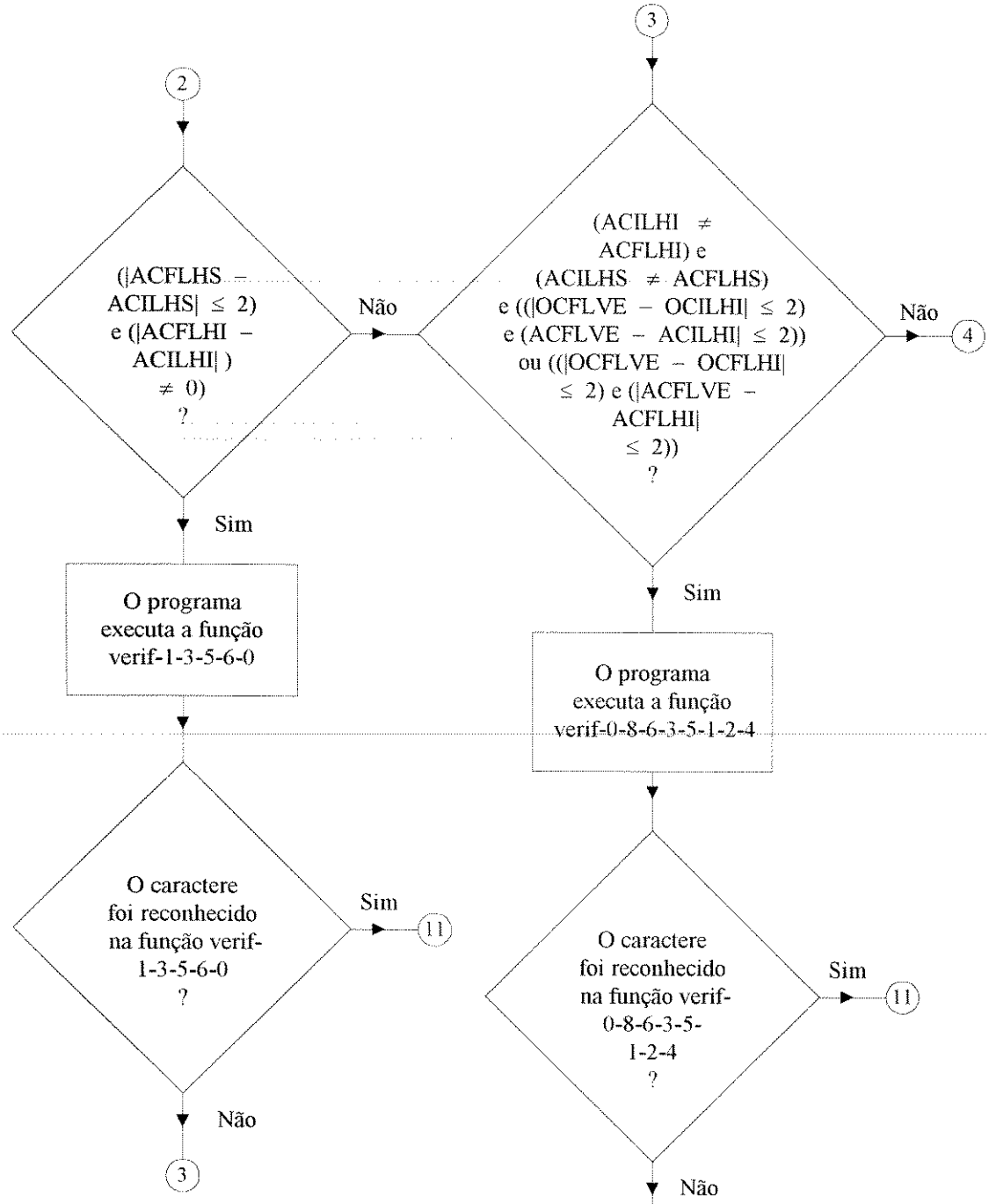


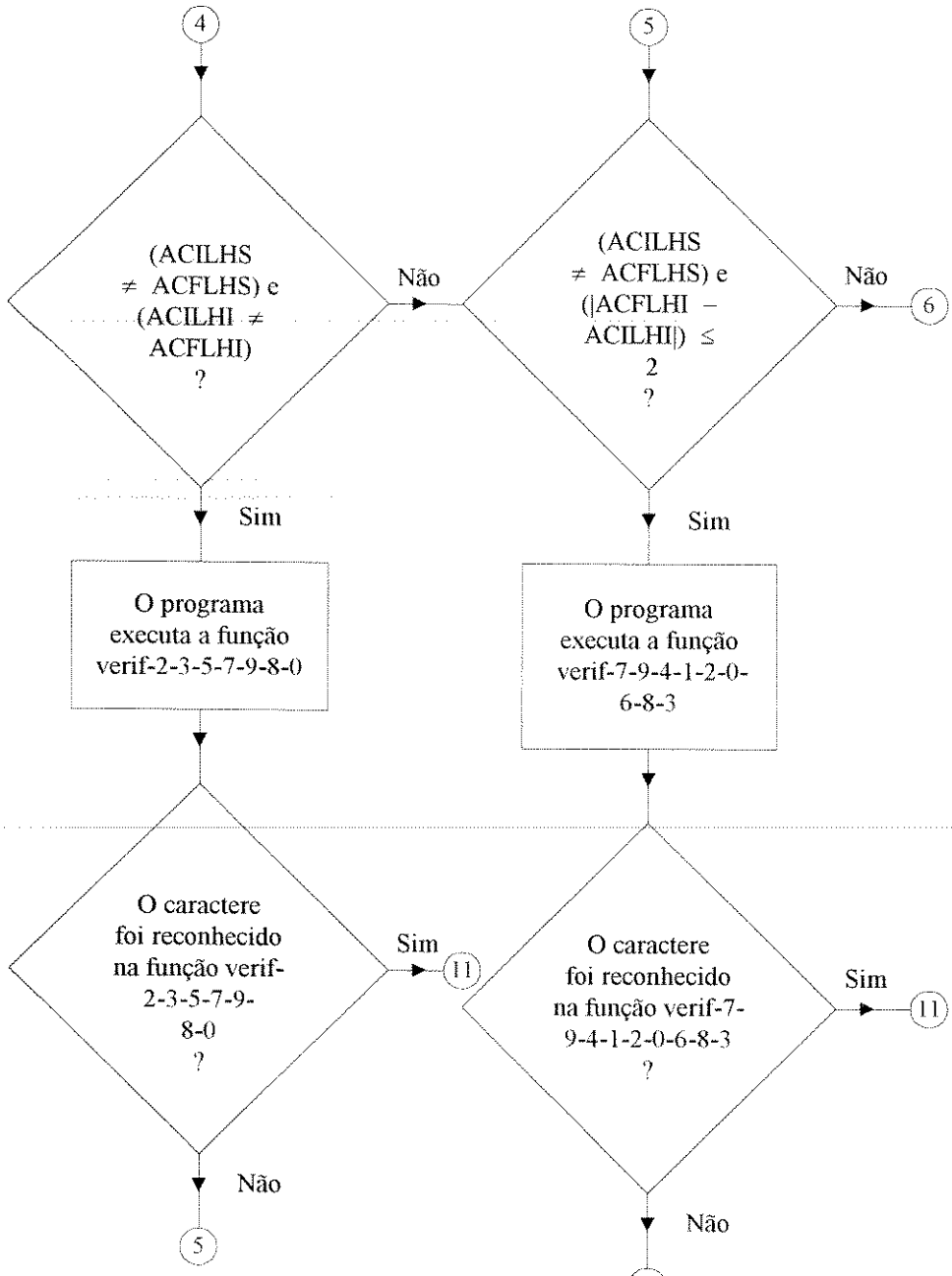
Figura 416(a) Primeira parte do fluxograma de todo o processo utilizado no reconhecimento de caracteres numéricos manuscritos isolados.



LHS - linha horizontal superior
ACILHS - abscissa da coordenada inicial da LHS
ACFLHS - abscissa da coordenada final da LHS
LHI - linha horizontal inferior
ACILHI - abscissa da coordenada inicial da LHI
OCILHI - ordenada da coordenada inicial da LHI
ACFLHI - abscissa da coordenada final da LHI

LVE - linha vertical esquerda
OCFLHI - ordenada da coordenada final da LHI
ACFLVE - abscissa da coordenada final da LVE
OCFLVE - ordenada da coordenada final da LVE

Figura 4.16(b) Segunda parte do fluxograma de todo o processo utilizado no reconhecimento de caracteres numéricos manuscritos isolados.



LHS - linha horizontal superior
ACILHS - abscissa da coordenada inicial da LHS
ACFLHS - abscissa da coordenada final da LHS
LHI - linha horizontal inferior
ACILHI - abscissa da coordenada inicial da LHI
ACFLHI - abscissa da coordenada final da LHI

Figura 4.16(c) Terceira parte do fluxograma de todo o processo utilizado no reconhecimento de caracteres numéricos manuscritos isolados.

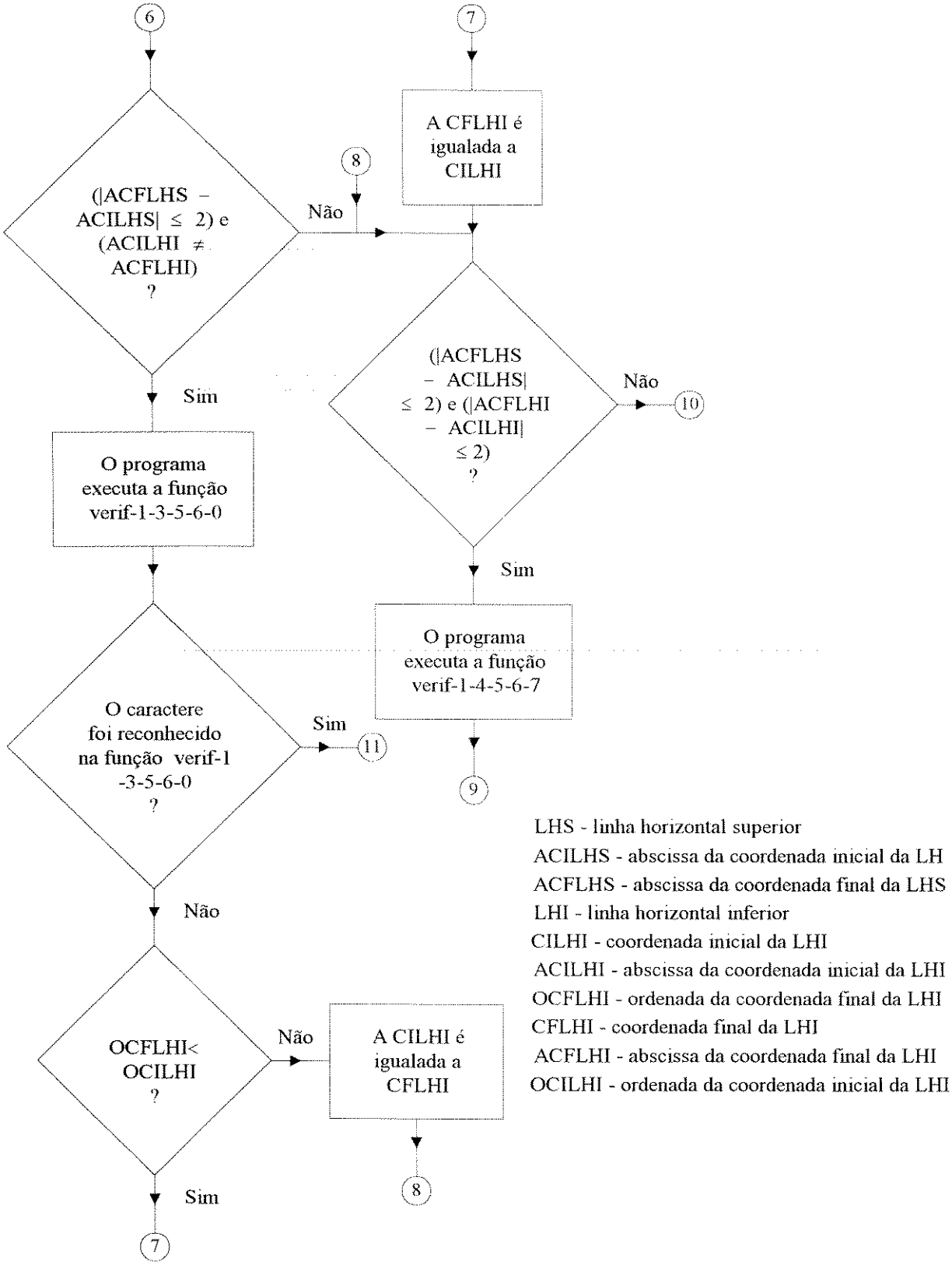


Figura 4.16(d) Quarta parte do fluxograma de todo o processo utilizado no reconhecimento de caracteres numéricos manuscritos isolados.

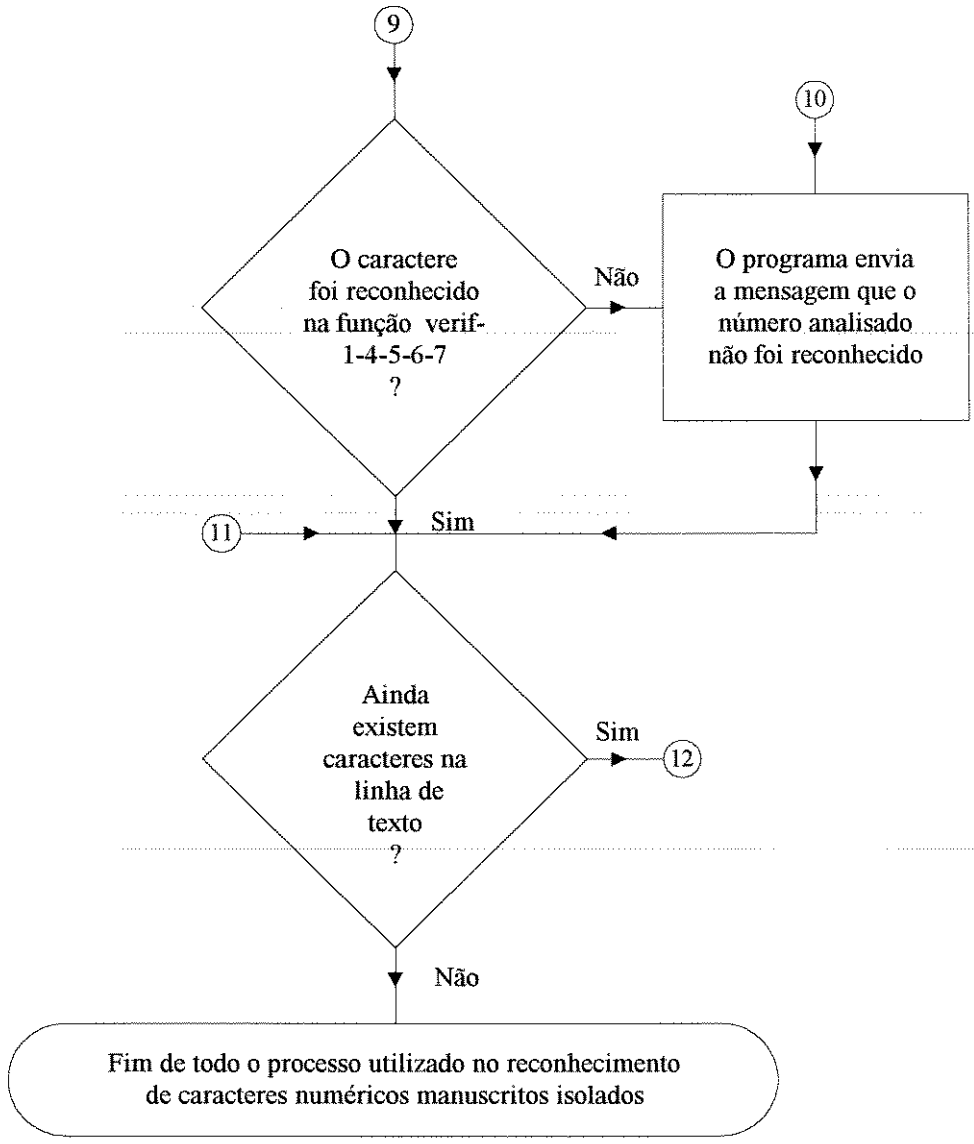
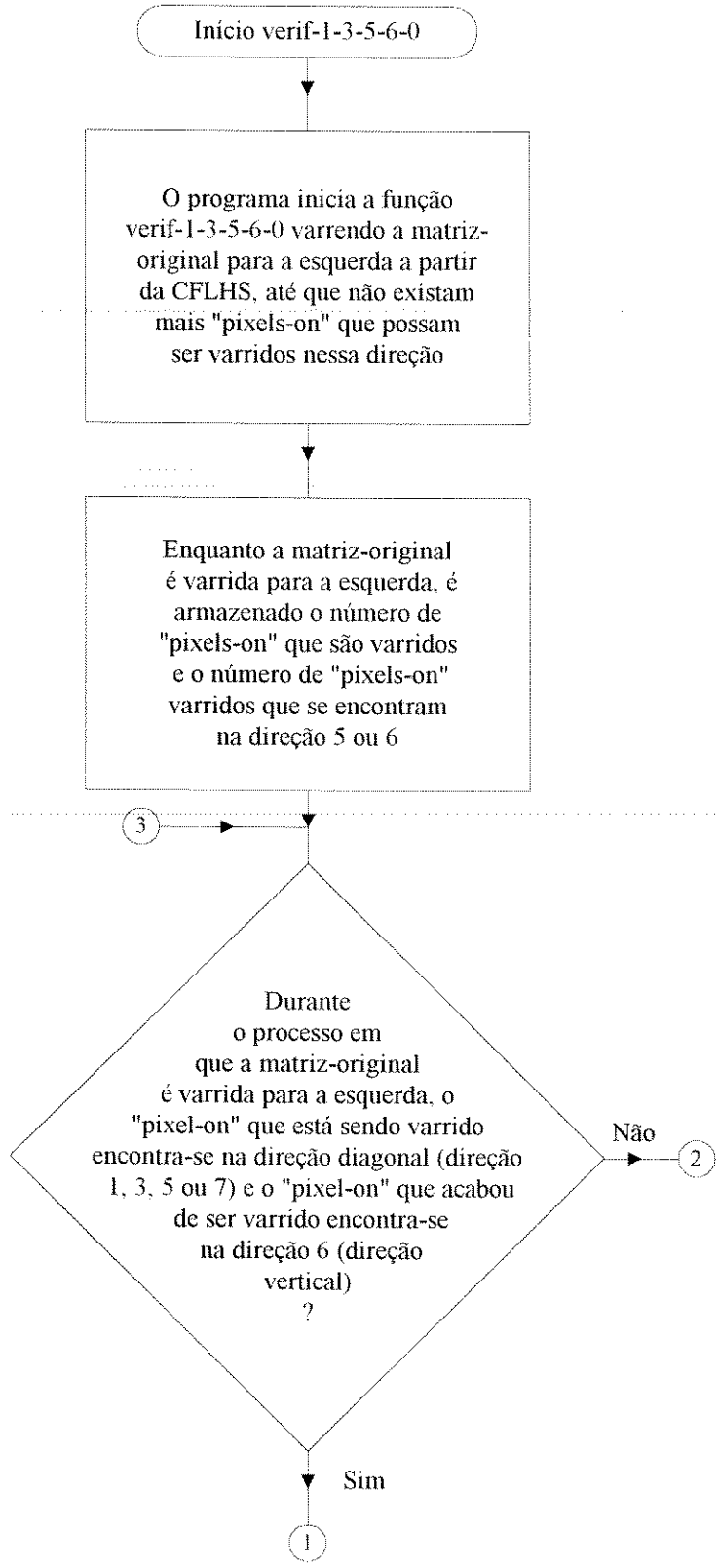


Figura 4.16(e) Quinta parte do fluxograma de todo o processo utilizado no reconhecimento de caracteres numéricos manuscritos isolados.

Fluxograma da função verif-1-3-5-6-0



LHS - linha horizontal superior
CFLHS - coordenada final da LHS

Figura 4.17(a) - Primeira parte do fluxograma da função verif-1-3-5-6-0.

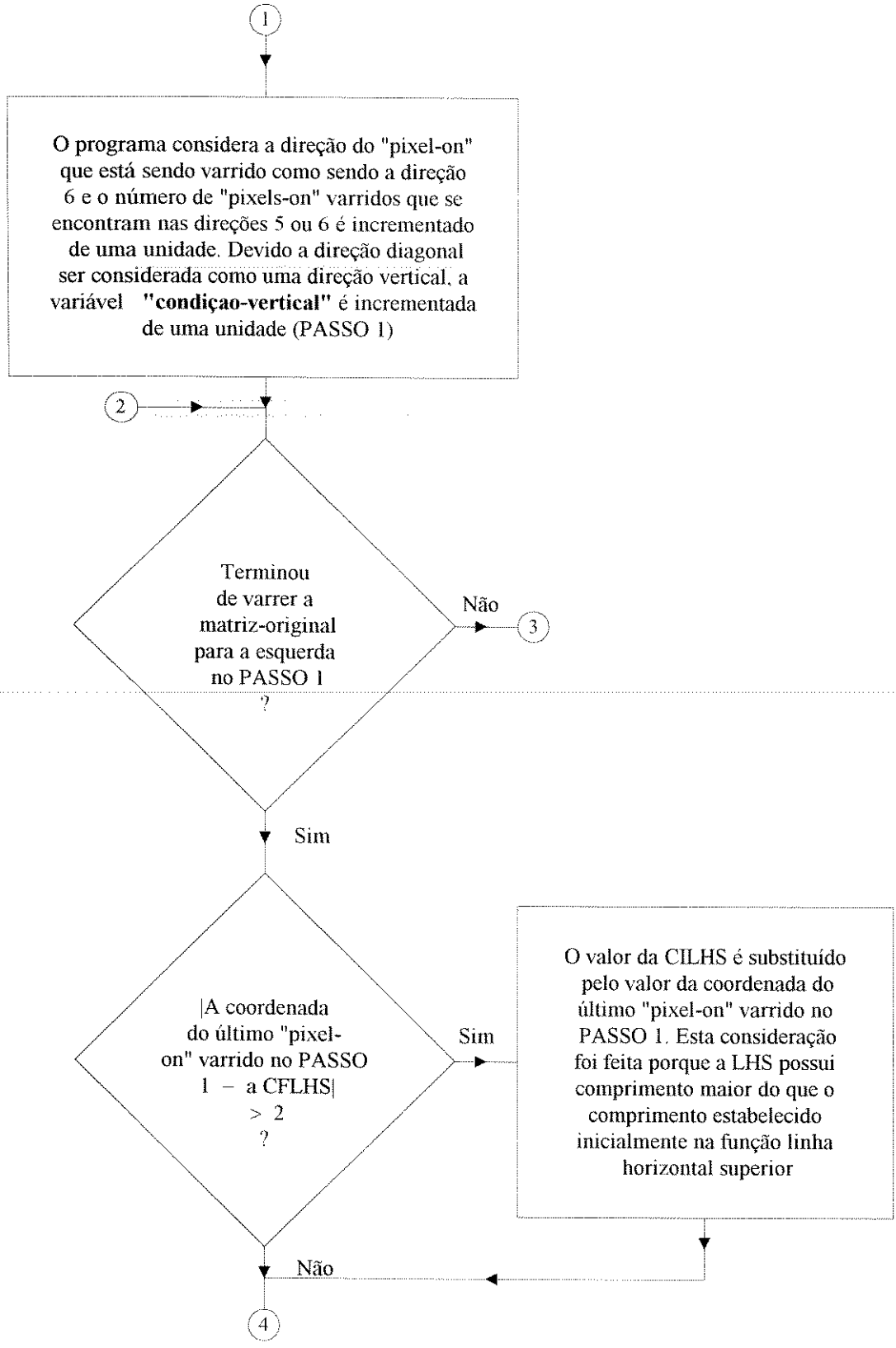


Figura 4.17(b) - Segunda parte do fluxograma da função verif-1-3-5-6-0.

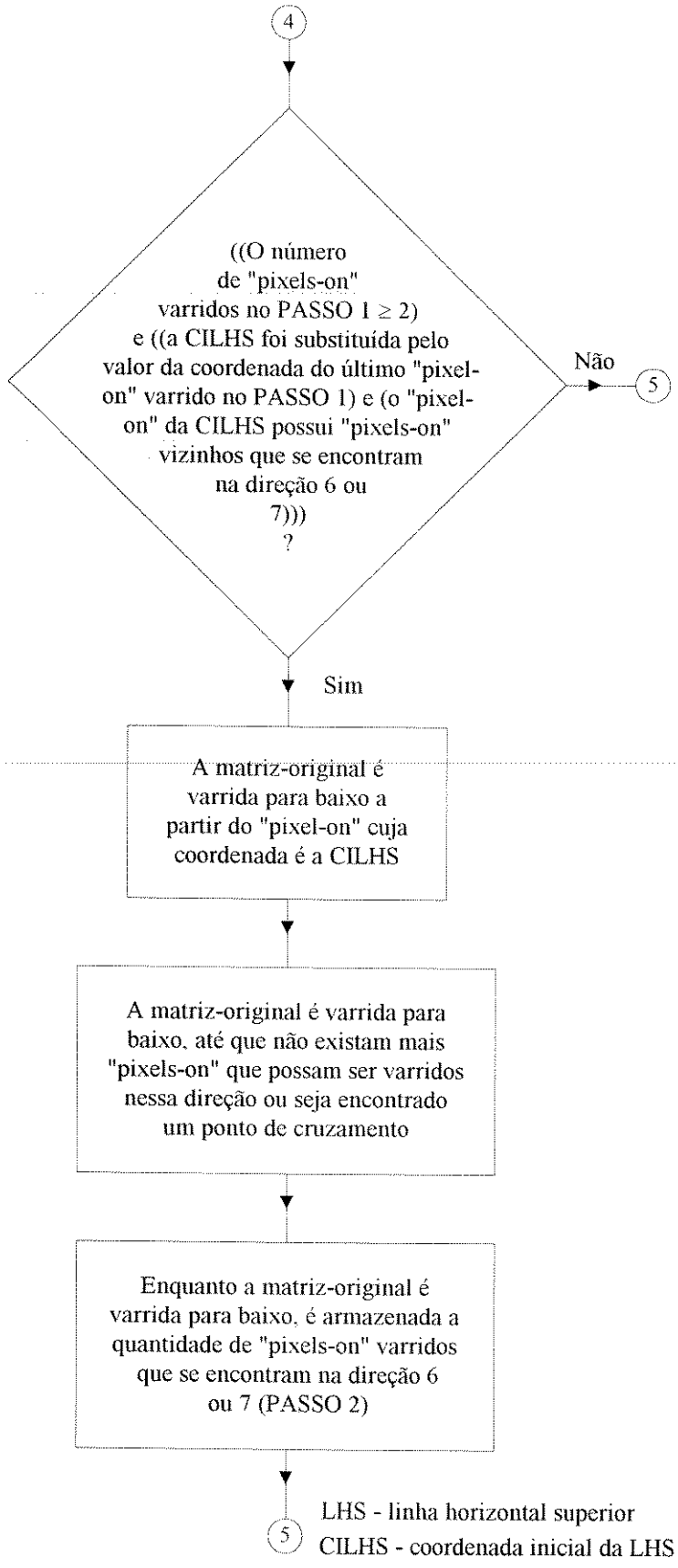
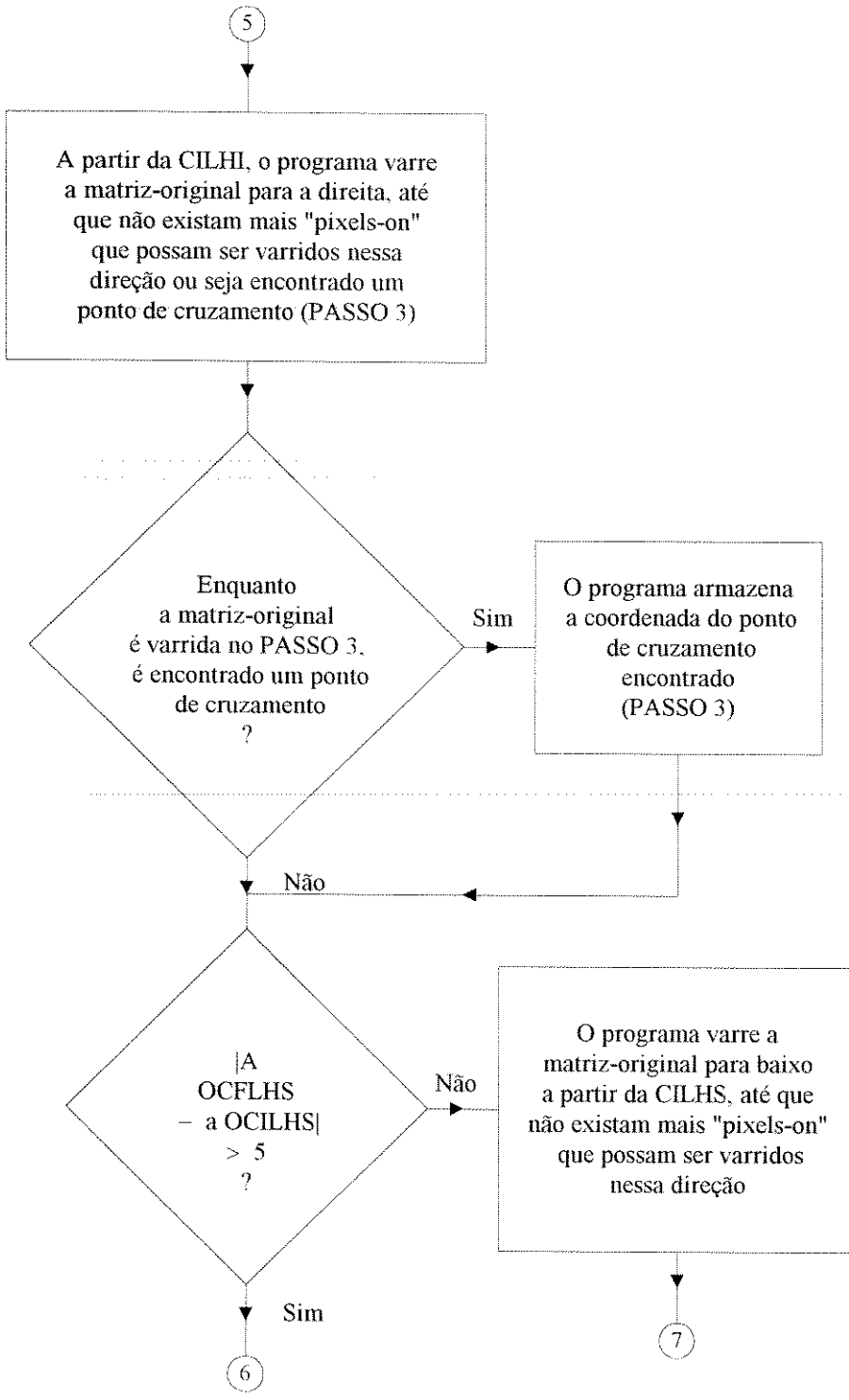


Figura 4.17(c) - Terceira parte do fluxograma da função verif-1-3-5-6-0.



LHS - linha horizontal superior
CILHS - coordenada inicial da LHS
OCILHS - ordenada da coordenada inicial da LHS
OCFLHS - ordenada da coordenada final da LHS
CILHI - coordenada inicial da LHI
LHI - linha horizontal inferior

Figura 4.17(d) - Quarta parte do fluxograma da função verif-1-3-5-6-0.

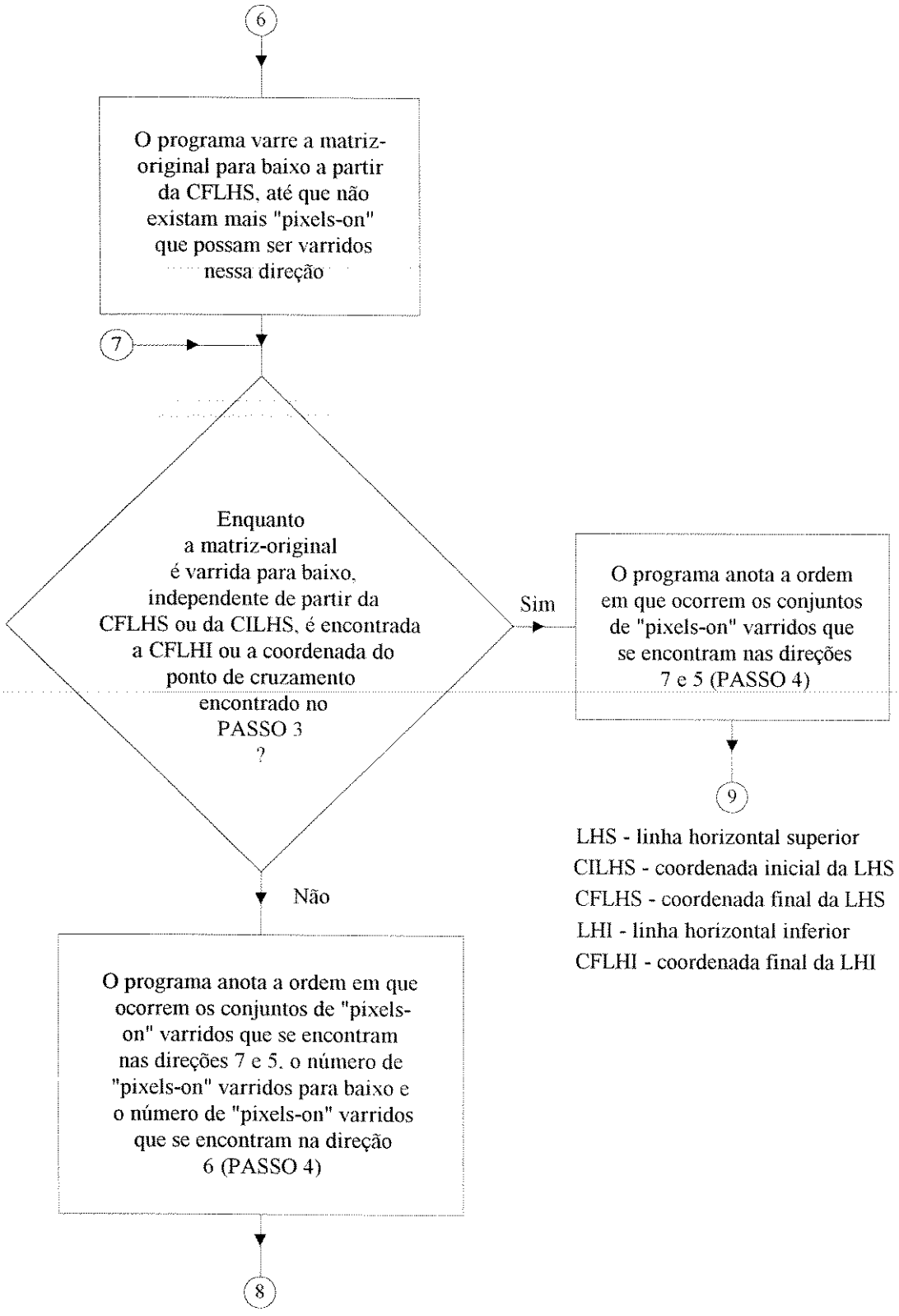


Figura 4.17(e) - Quinta parte do fluxograma da função verif-1-3-5-6-0.

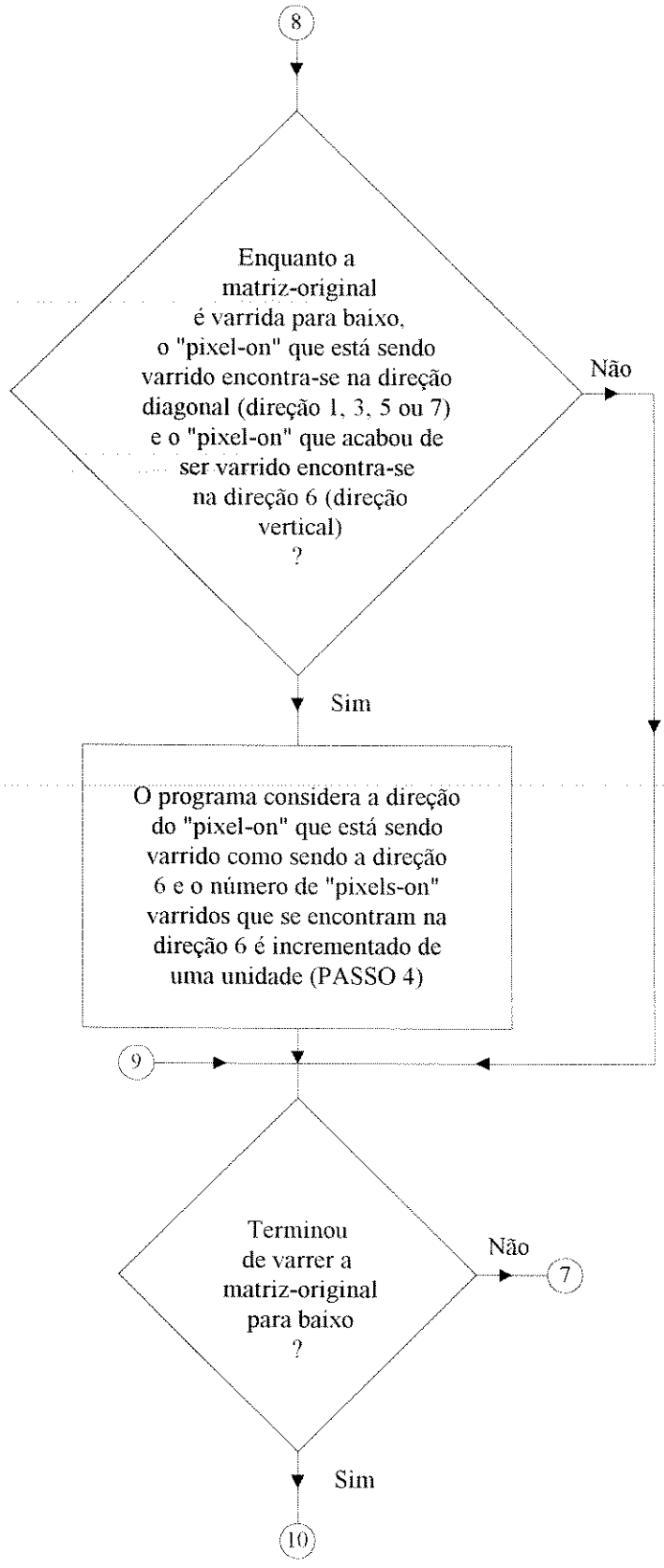


Figura 4.17(f) - Sexta parte do fluxograma da função verif-1-3-5-6-0.

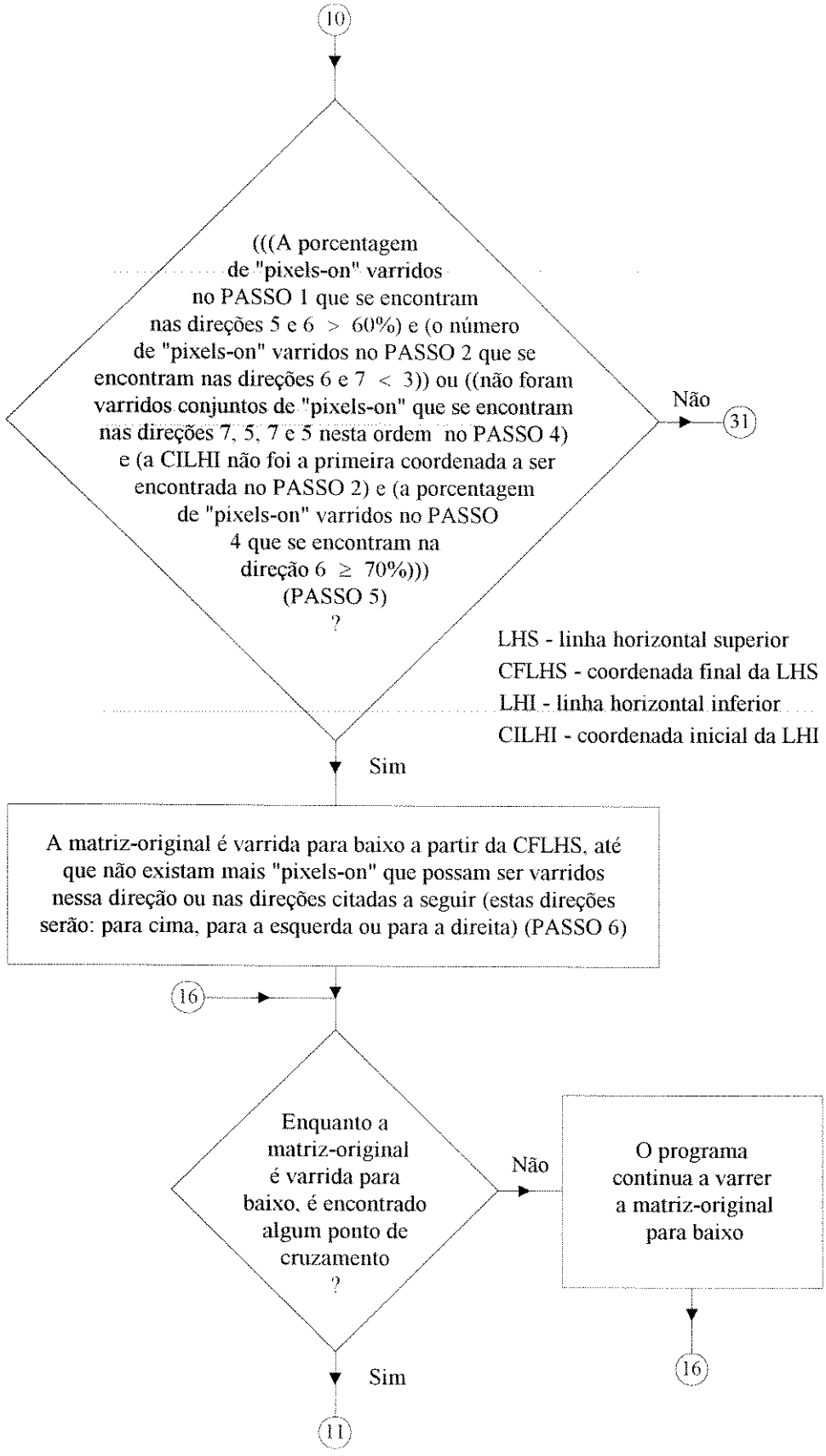


Figura 4.17(g) - Sétima parte do fluxograma da função verif-1-3-5-6-0.

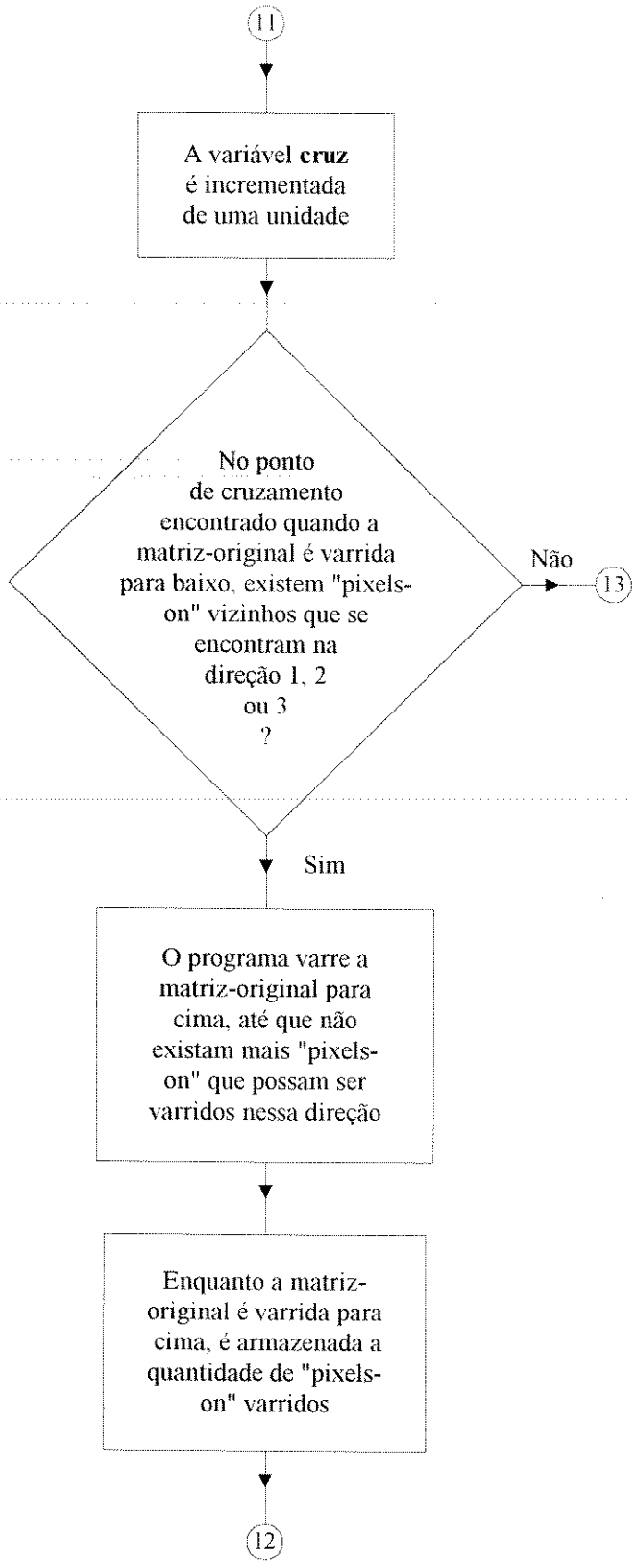


Figura 4.17(h) - Oitava parte do fluxograma da função verif-1-3-5-6-0.

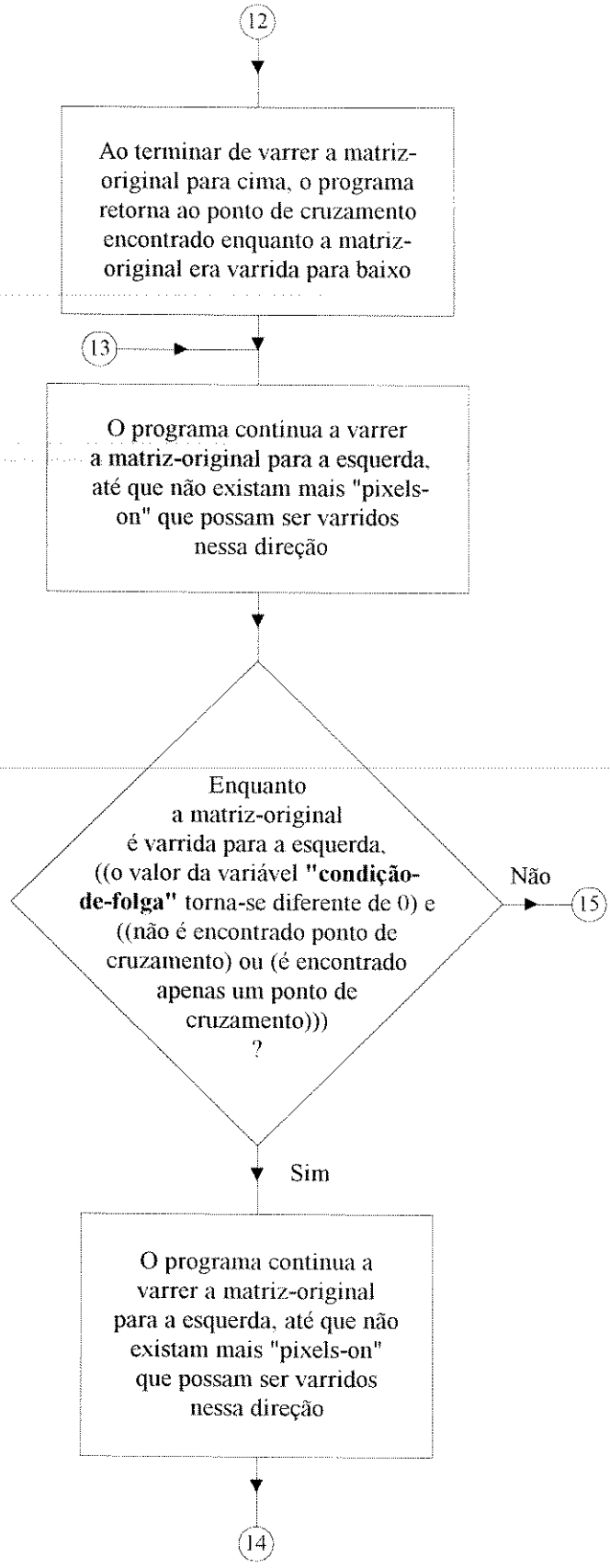
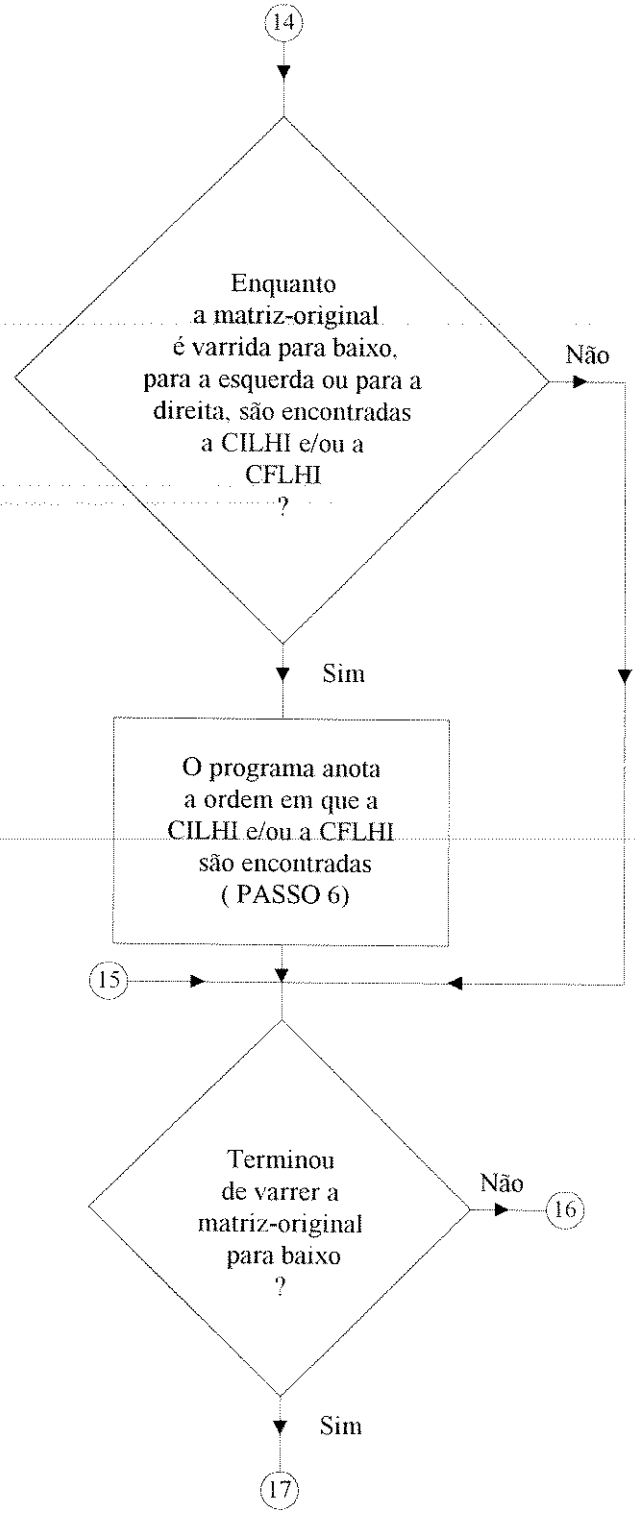


Figura 4.17(i) - Nona parte do fluxograma da função verif-1-3-5-6-0.



LHI - linha horizontal inferior
CILHI - coordenada inicial da LHI
CFLHI - coordenada final da LHI

Figura 4.17(j) - Décima parte do fluxograma da função verif-1-3-5-6-0.

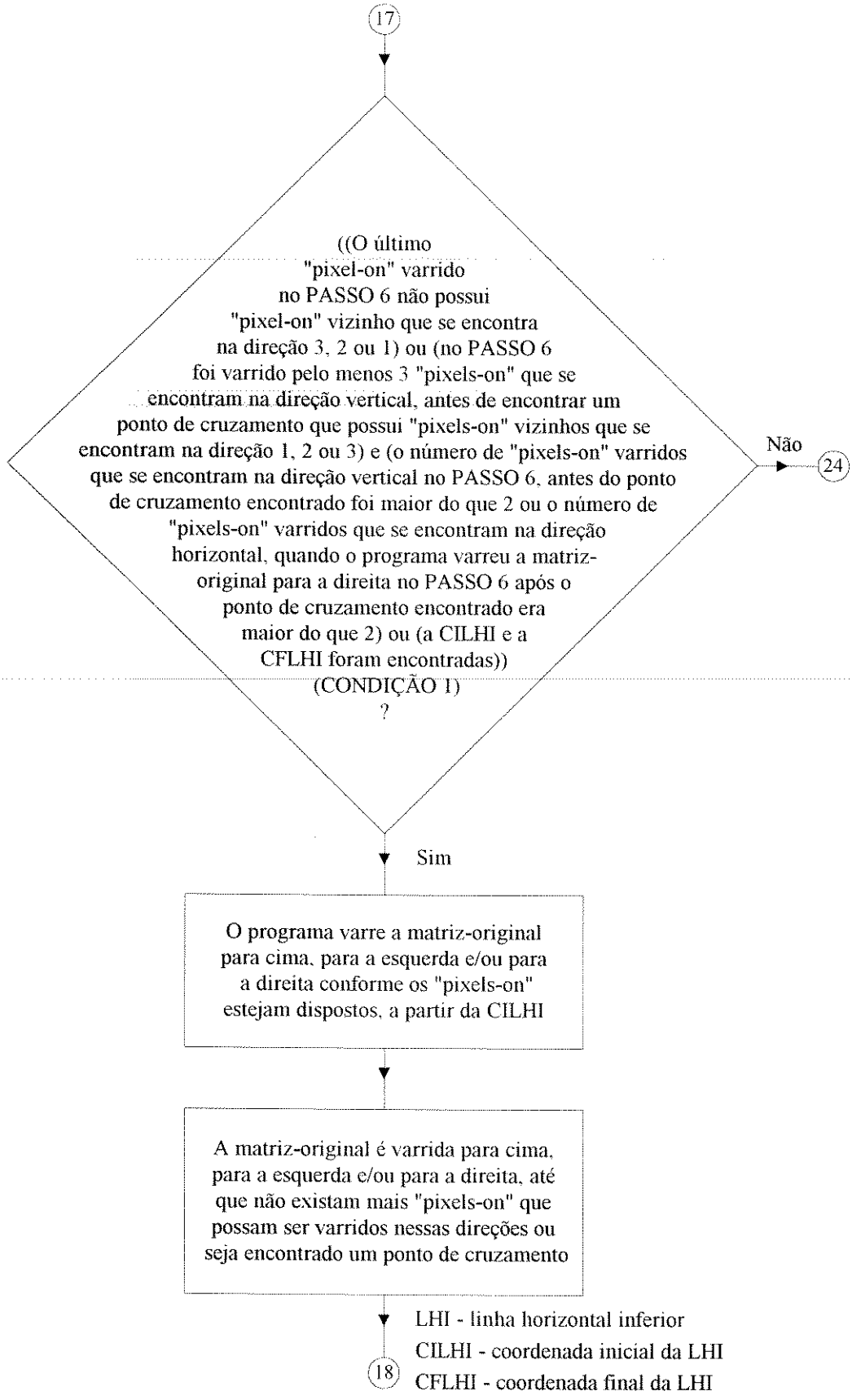


Figura 4.17(k) - Décima primeira parte do fluxograma da função verif-1-3-5-6-0.

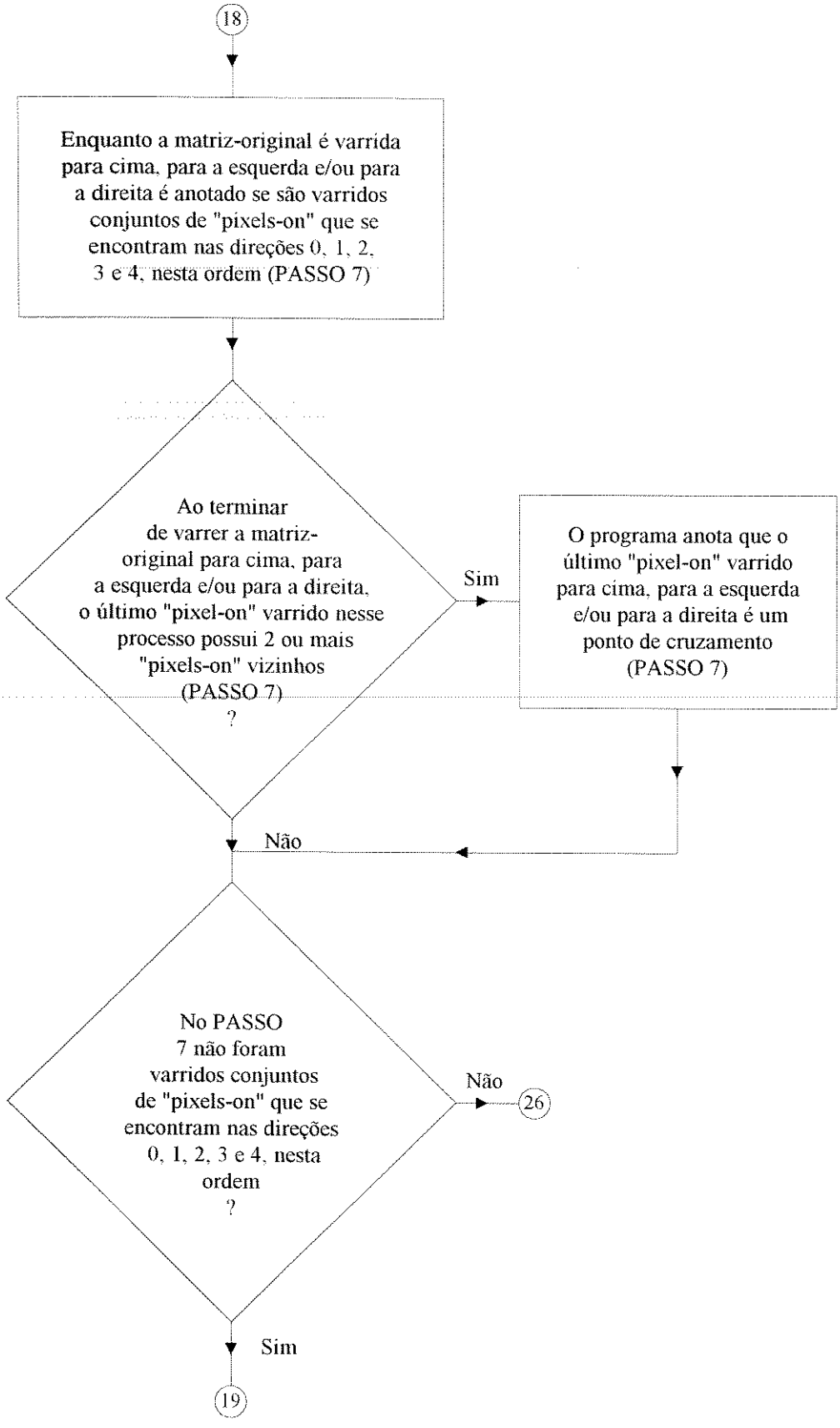


Figura 4.17(l) - Décima segunda parte do fluxograma da função verif-1-3-5-6-0.

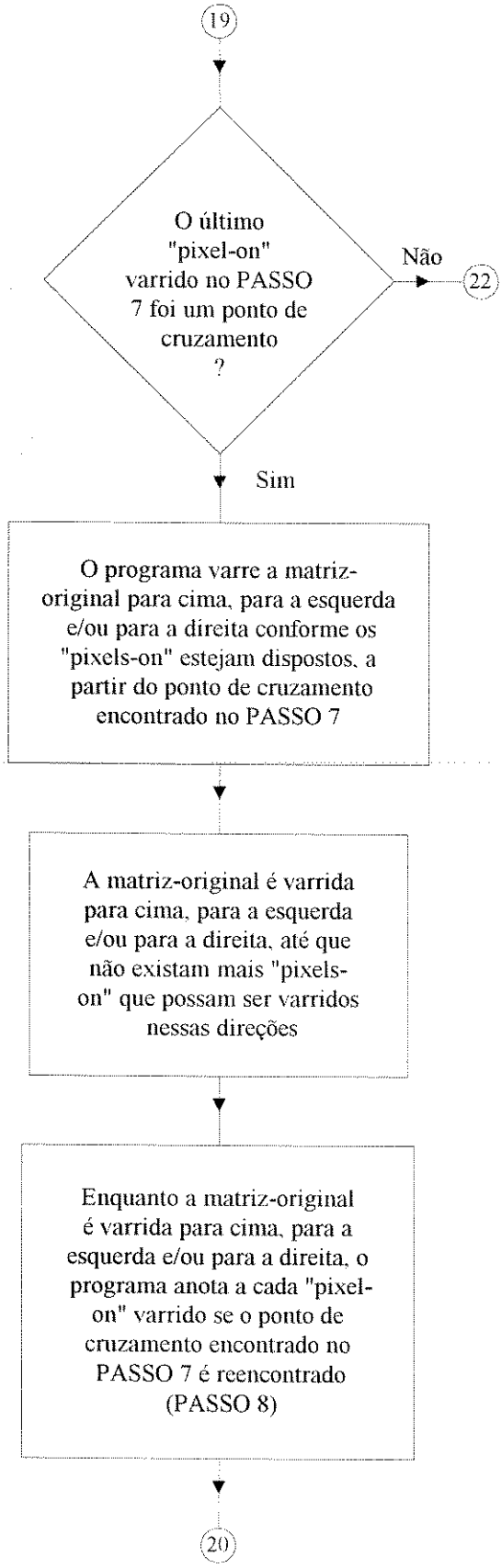


Figura 4.17(m) - Décima terceira parte do fluxograma da função verif-1-3-5-6-0.

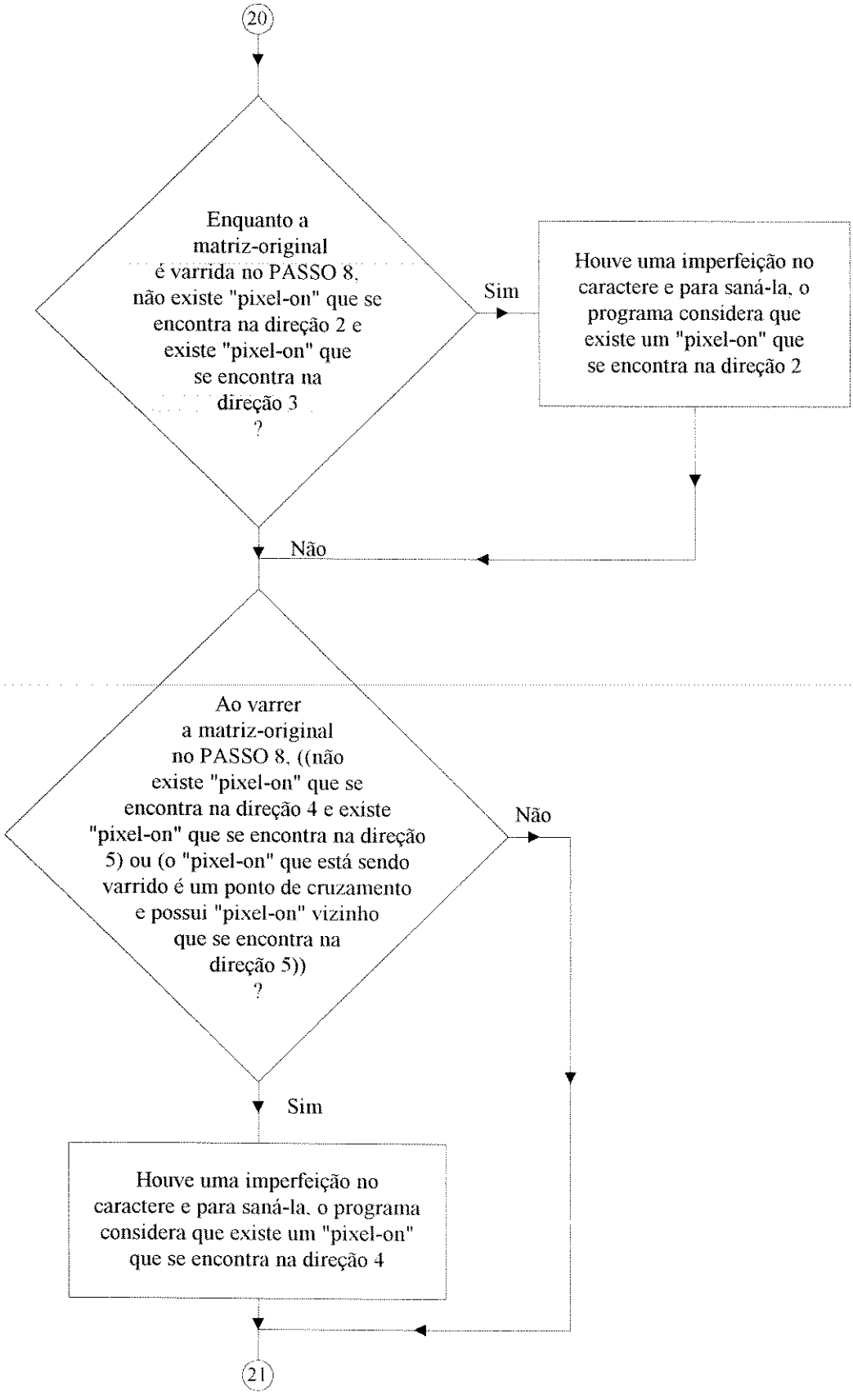


Figura 4.17(n) - Décima quarta parte do fluxograma da função verif-1-3-5-6-0.

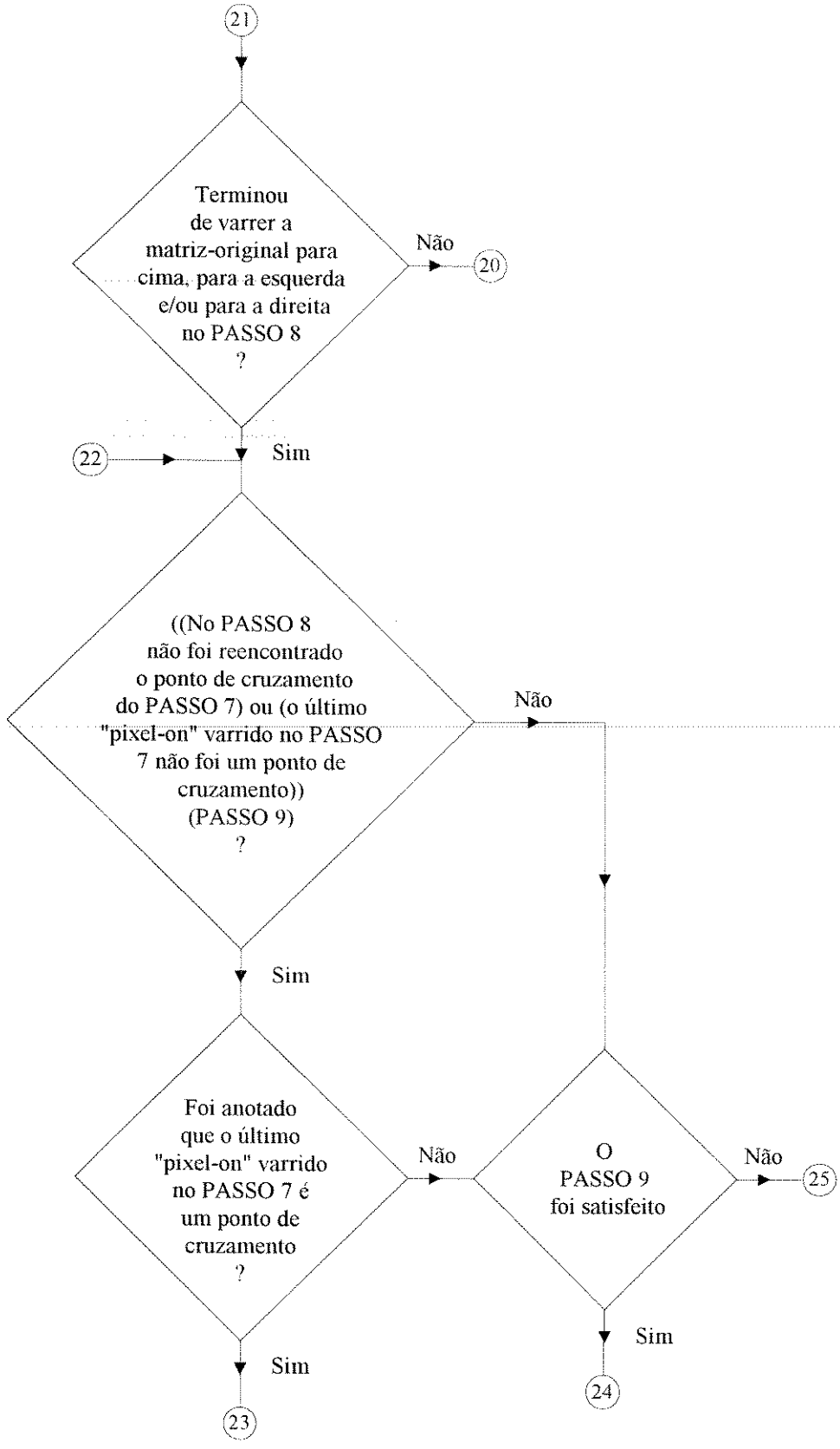


Figura 4.17(o) - Décima quinta parte do fluxograma da função verif-1-3-5-6-0.

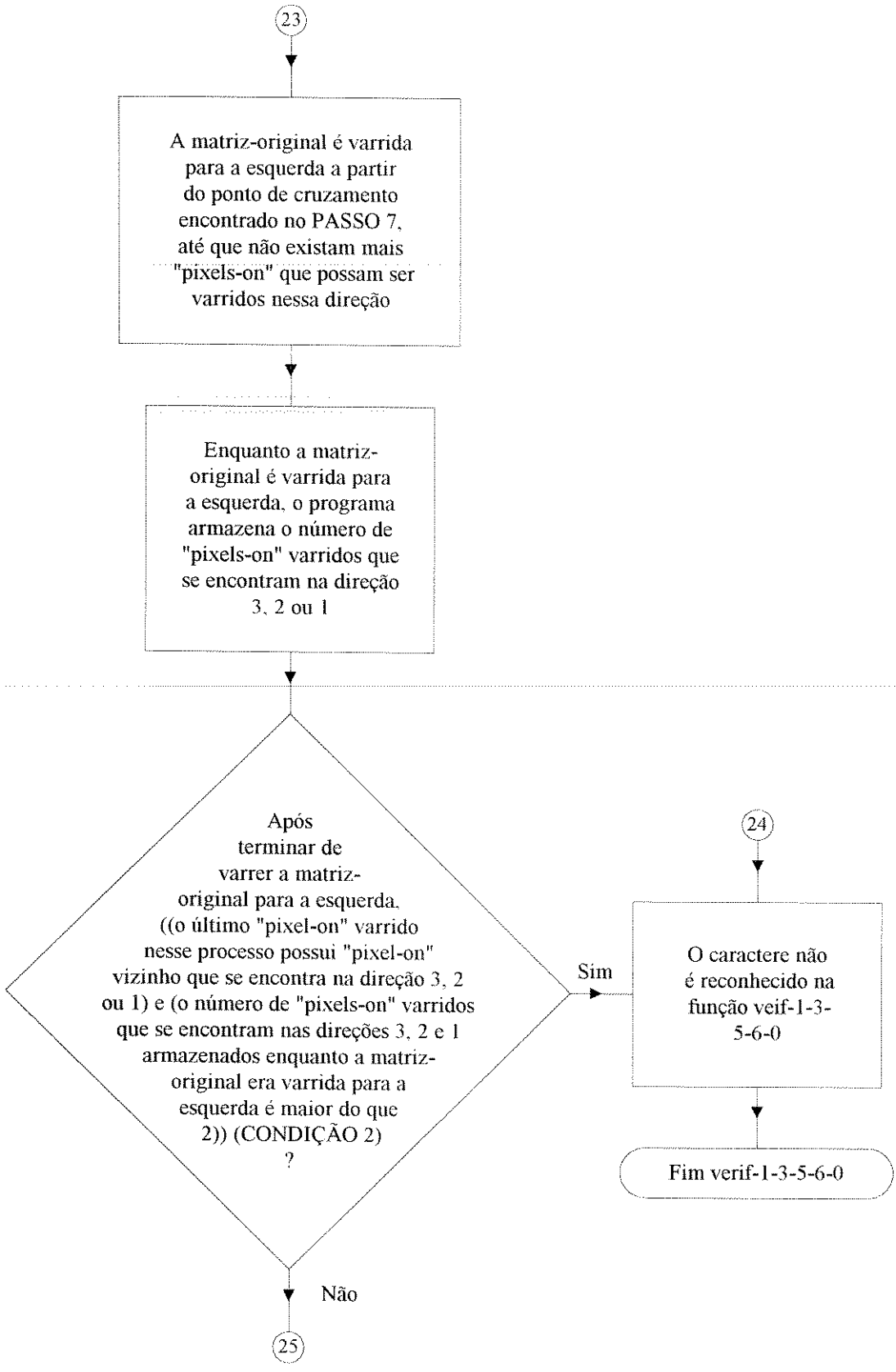


Figura 4.17(p) - Décima sexta parte do fluxograma da função verif-1-3-5-6-0.

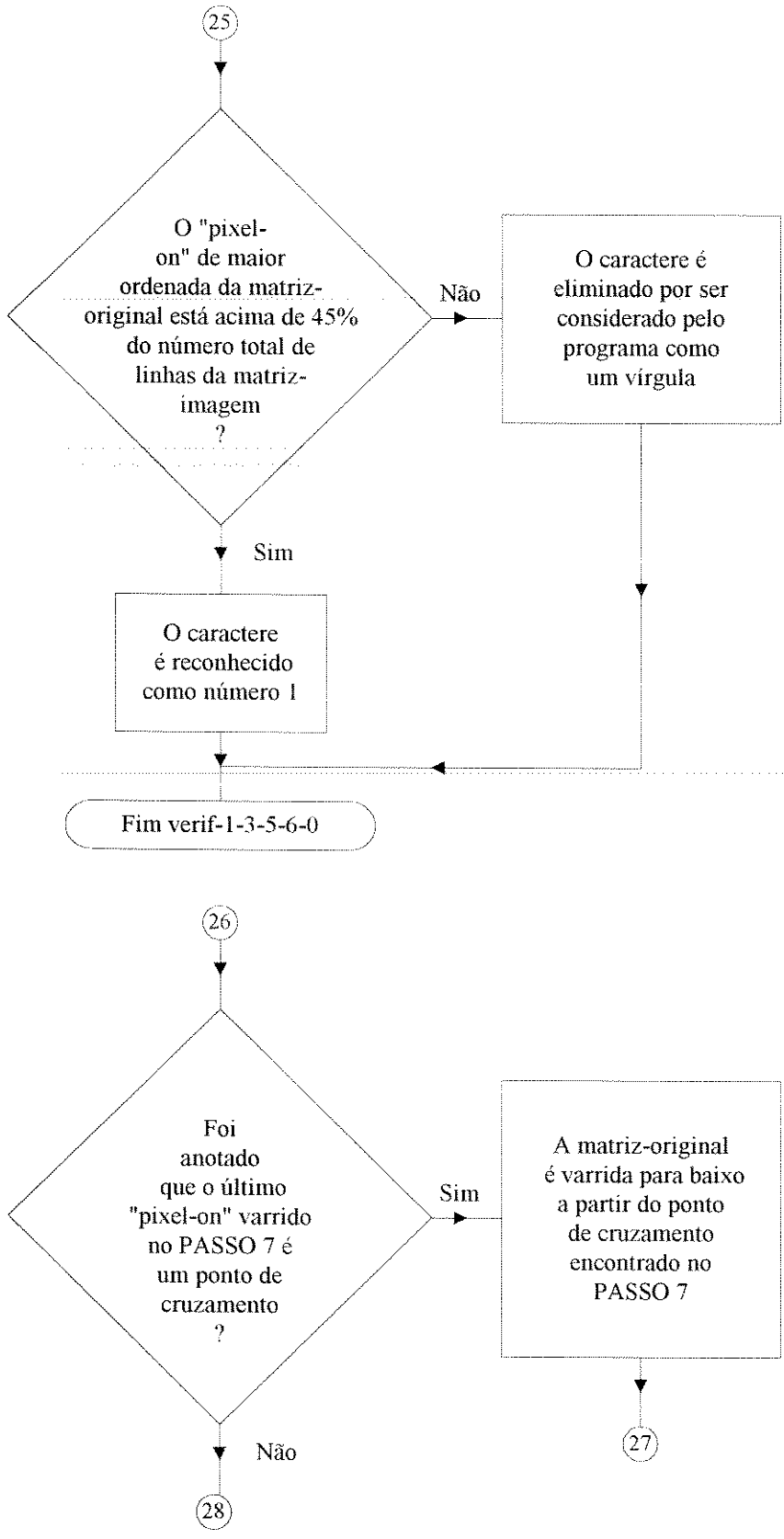


Figura 4.17(q) - Décima sétima parte do fluxograma da função verif-1-3-5-6-0.

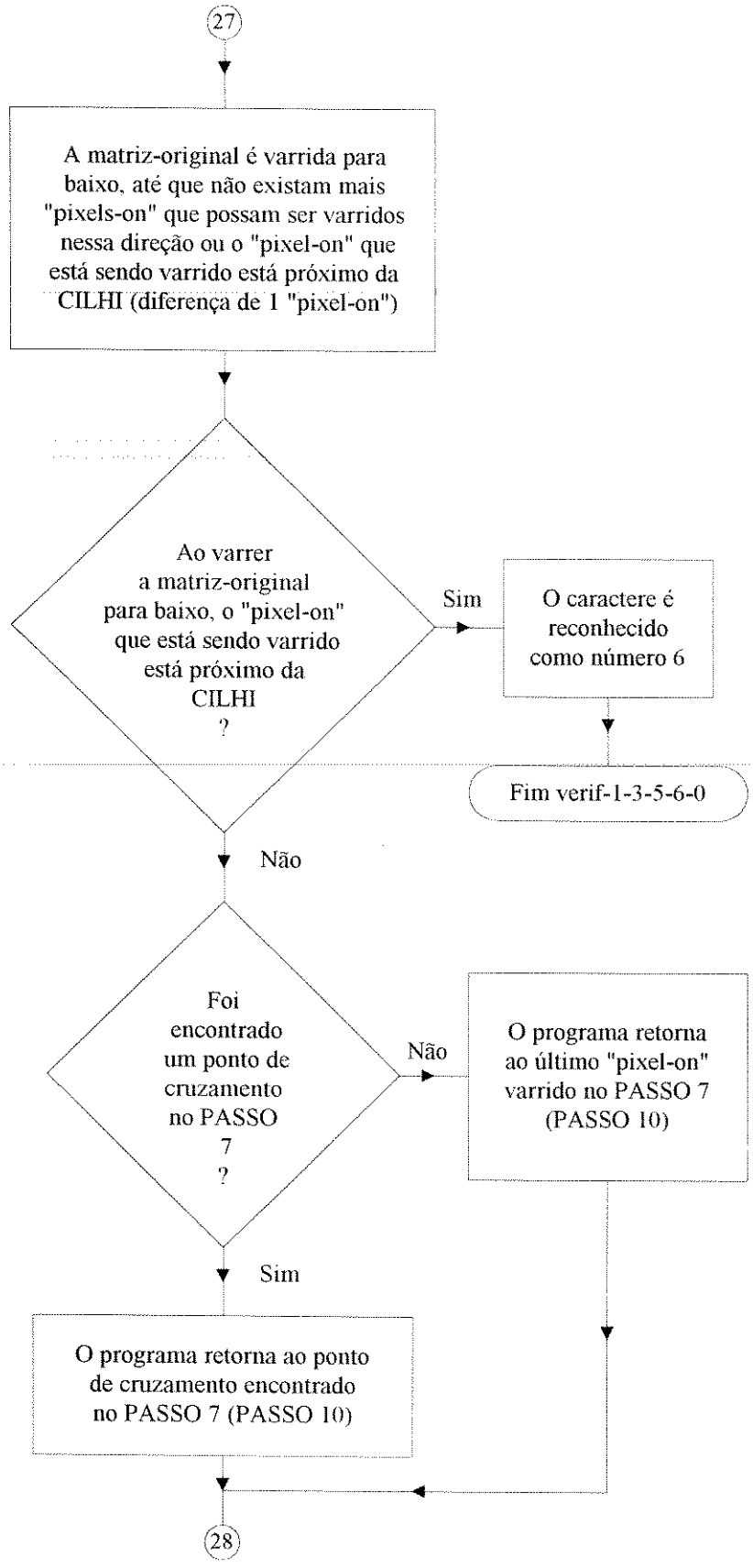


Figura 4.17(r) - Décima oitava parte do fluxograma da função verif-1-3-5-6-0.

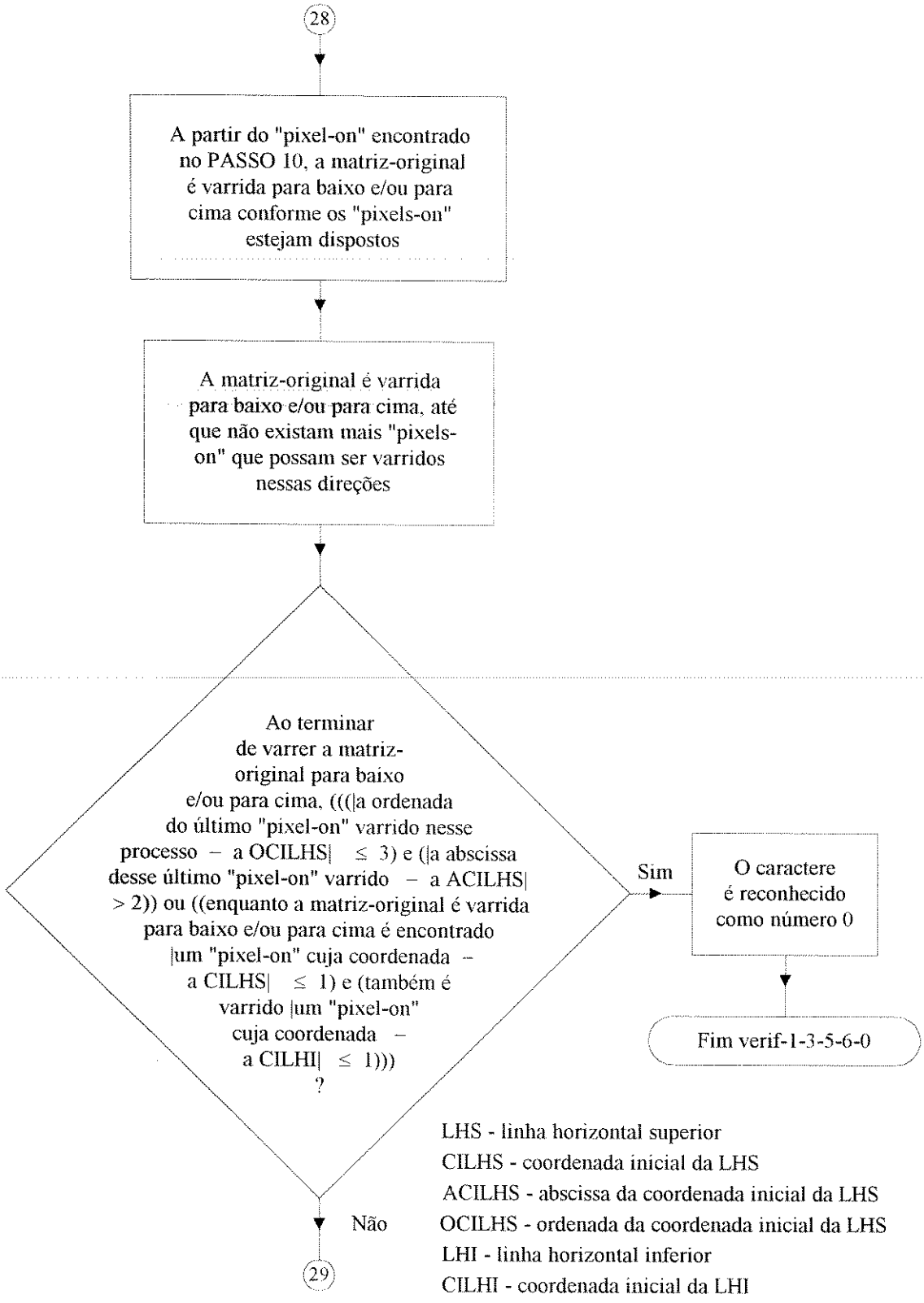


Figura 4.17(s) - Décima nona parte do fluxograma da função verif-1-3-5-6-0.

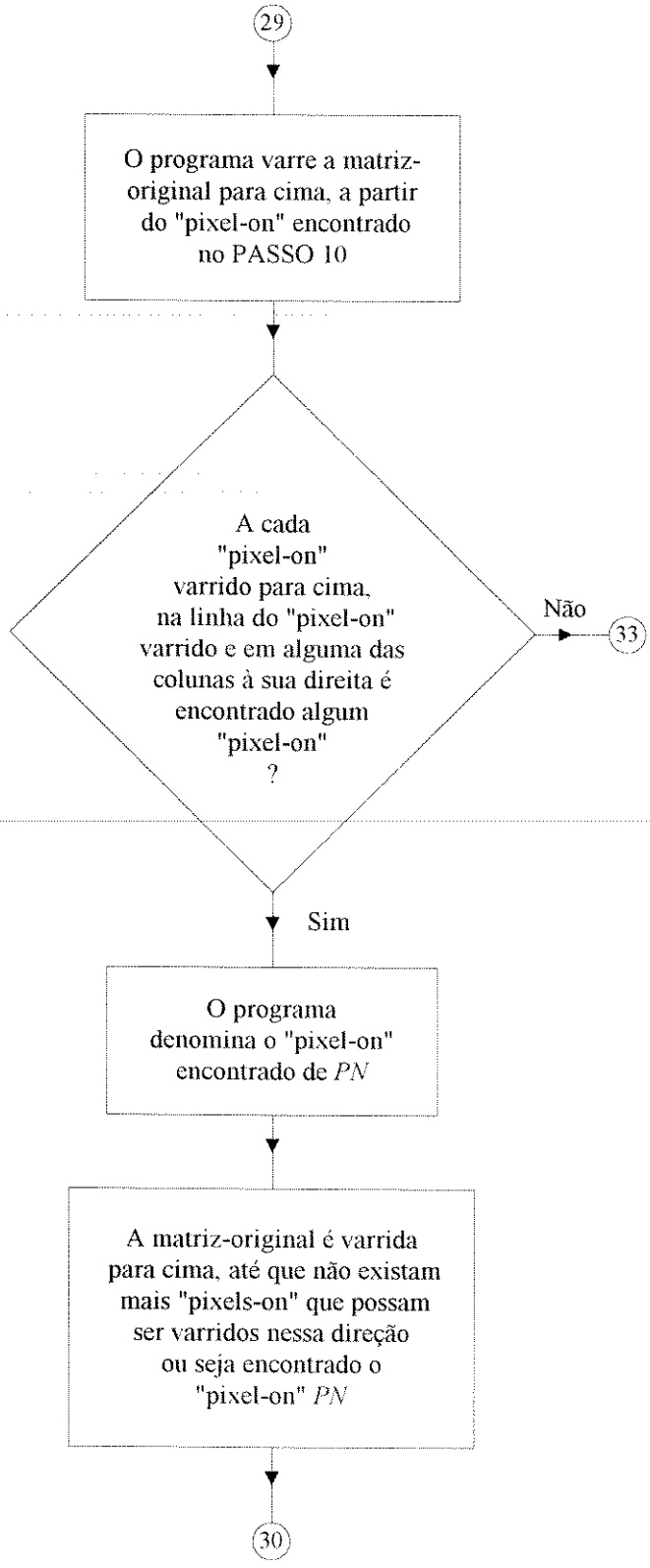
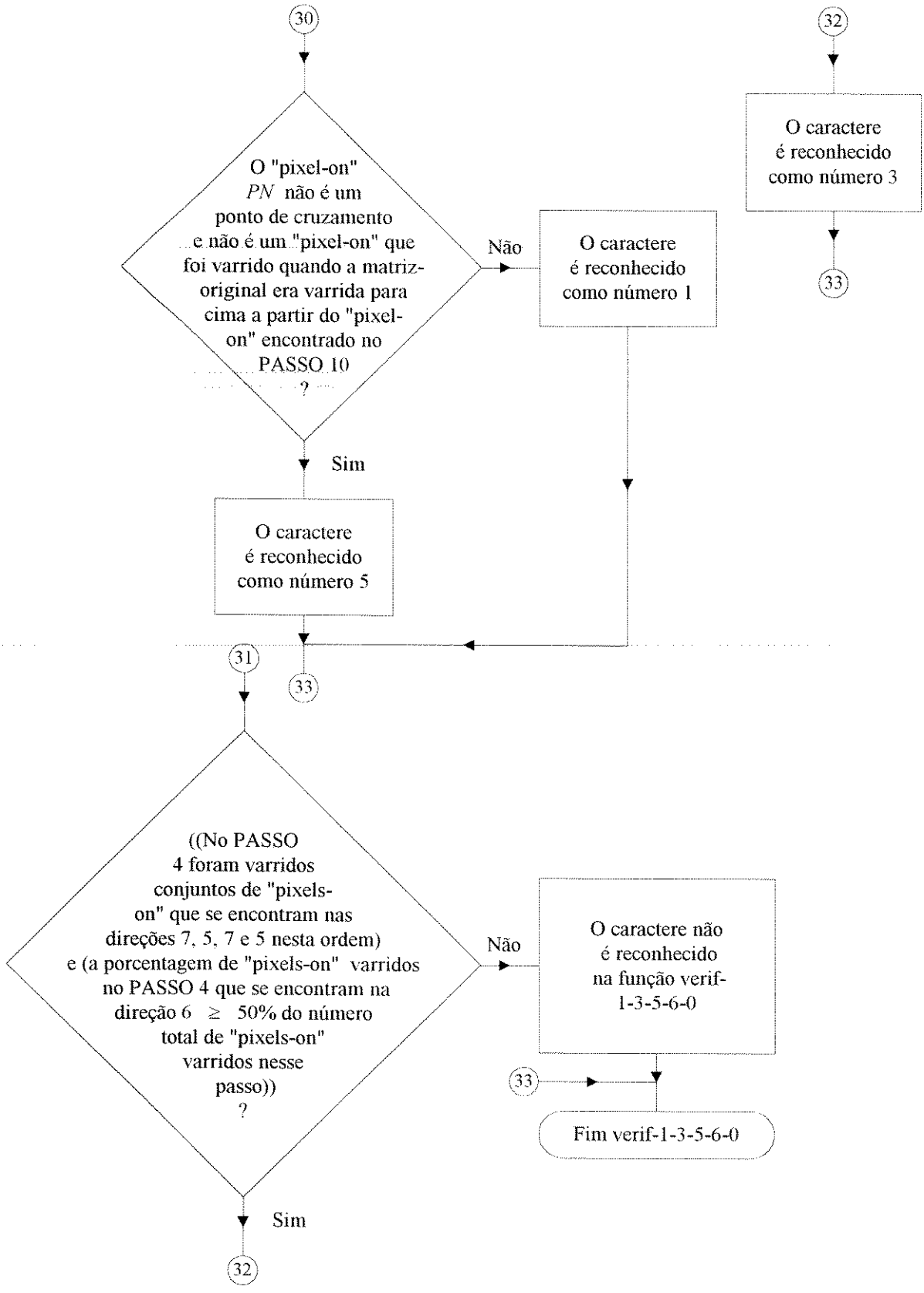
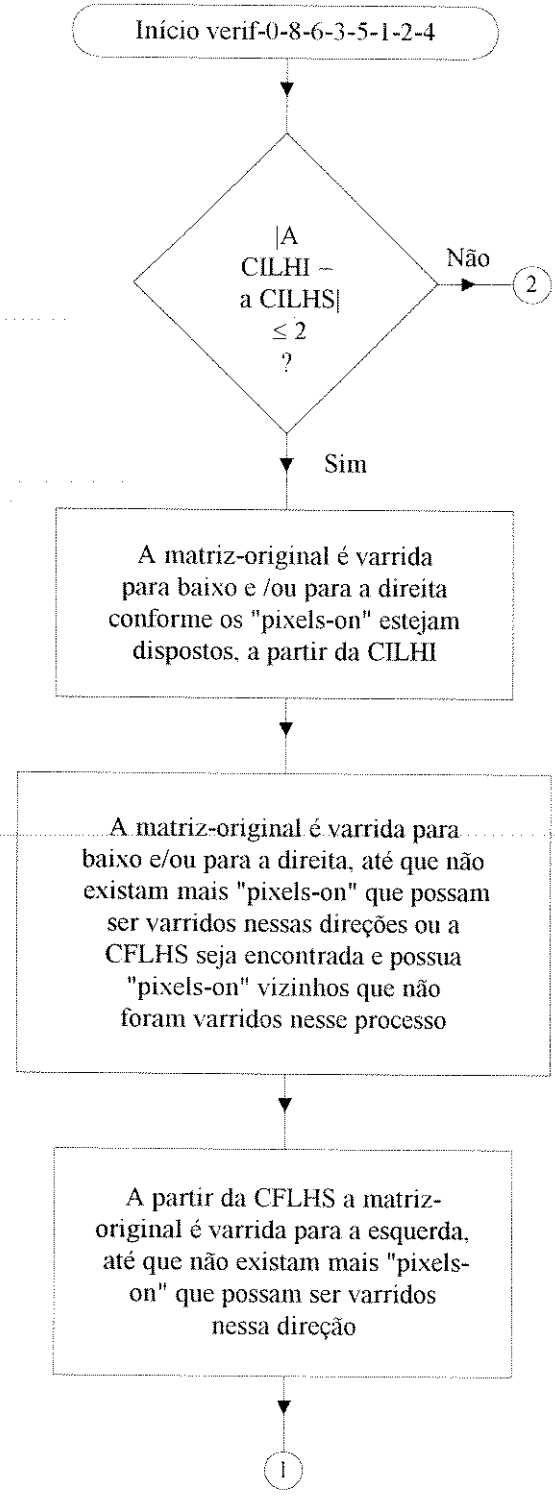


Figura 4.17(t) - Vigésima parte do fluxograma da função verif-1-3-5-6-0.



Figra 4.17(u) - Vigésima primeira parte do fluxograma da função verif-1-3-5-6-0.

Fluxograma da função verif-0-8-6-3-5-1-2-4



LHS - linha horizontal superior
CILHS - coordenada inicial da LHS
CFLHS - coordenada final da LHS
LHI - linha horizontal inferior
CILHI - coordenada inicial da LHI

Figura 4.19(a) - Primeira parte do fluxograma da função 0-8-6-3-5-1-2-4.

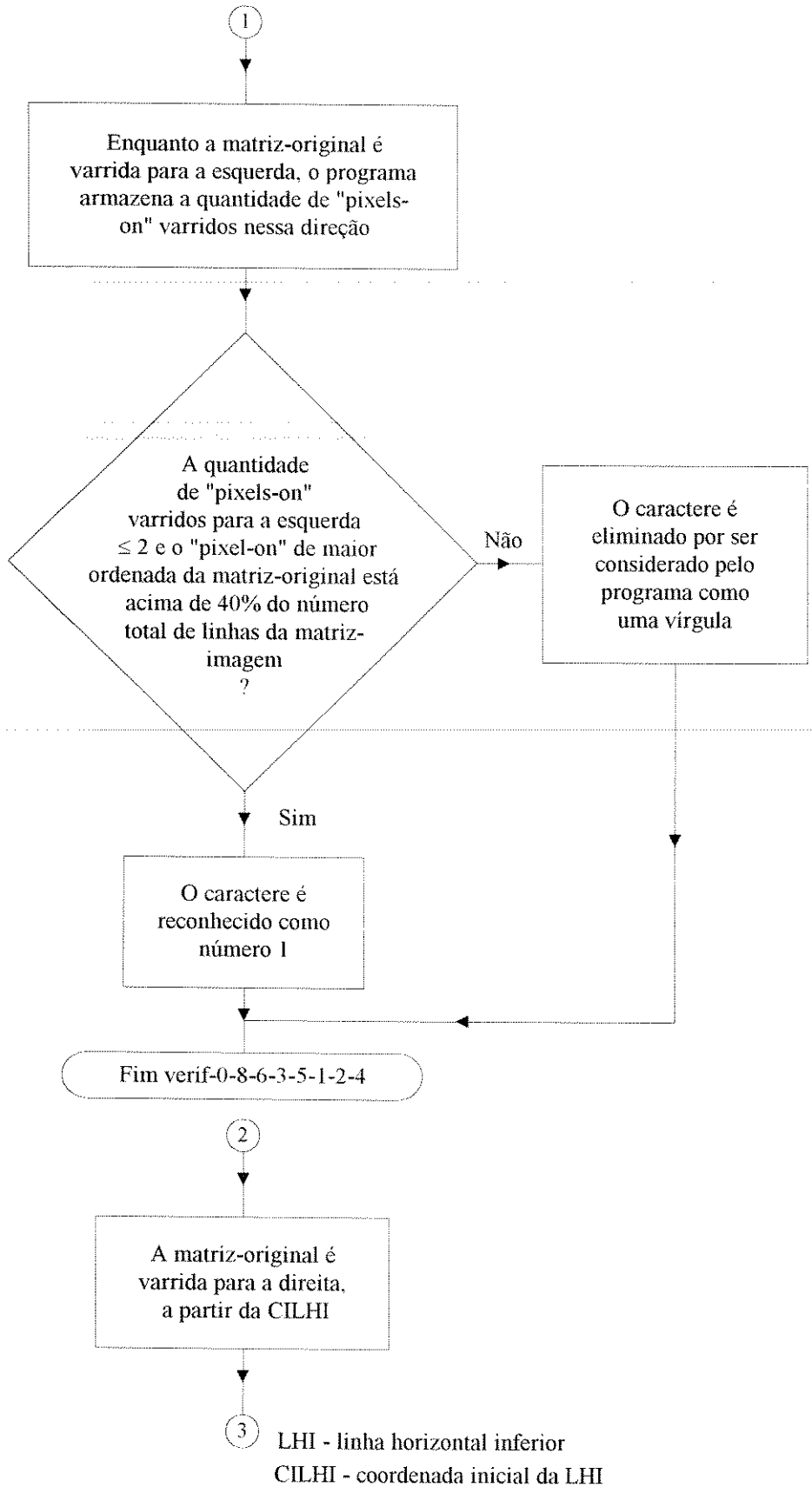
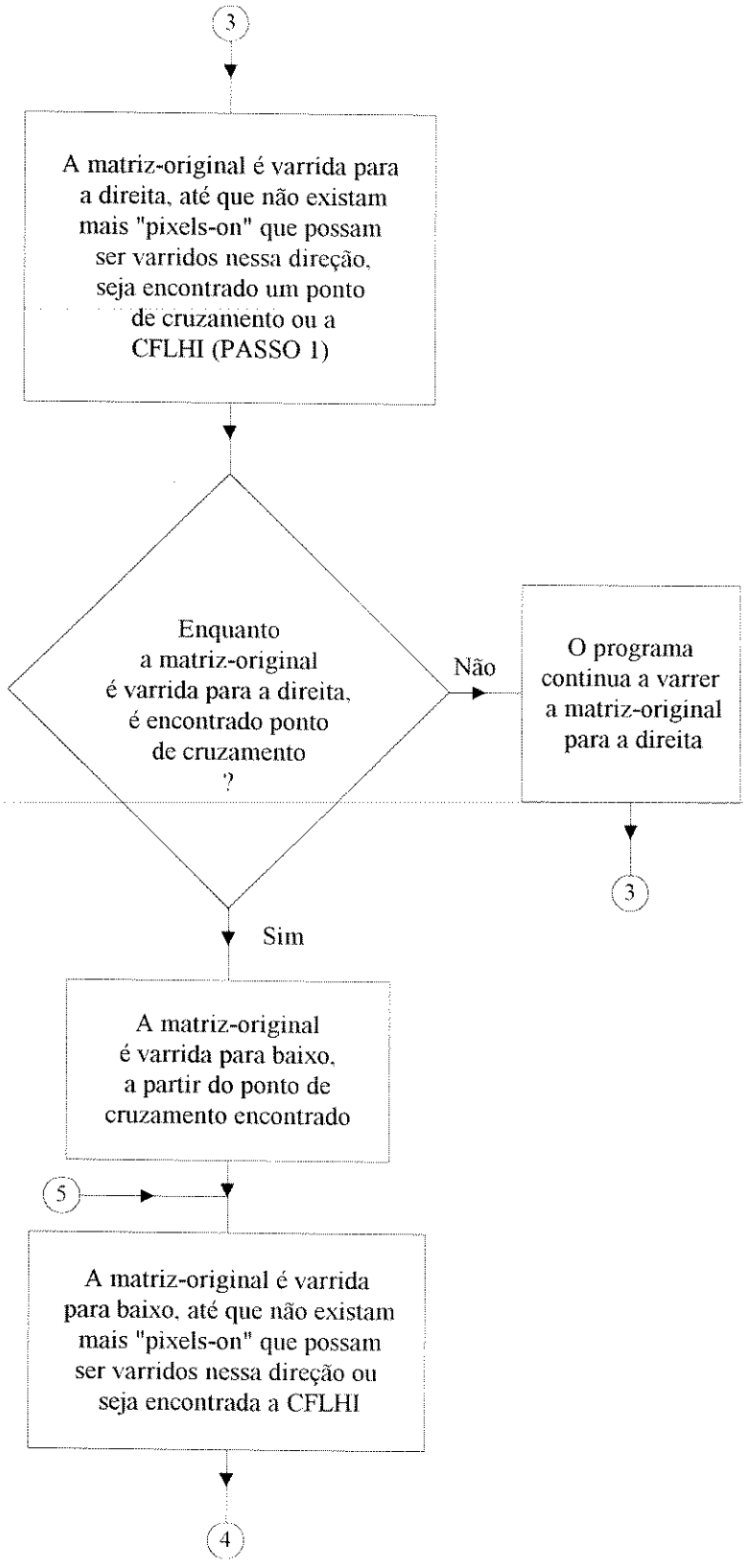


Figura 4.19(b) - Segunda parte do fluxograma da função 0-8-6-3-5-1-2-4.



LHI - linha horizontal inferior
CFLHI - coordenada final da LHI

Figura 4.19(c) - Terceira parte do fluxograma da função 0-8-6-3-5-1-2-4.

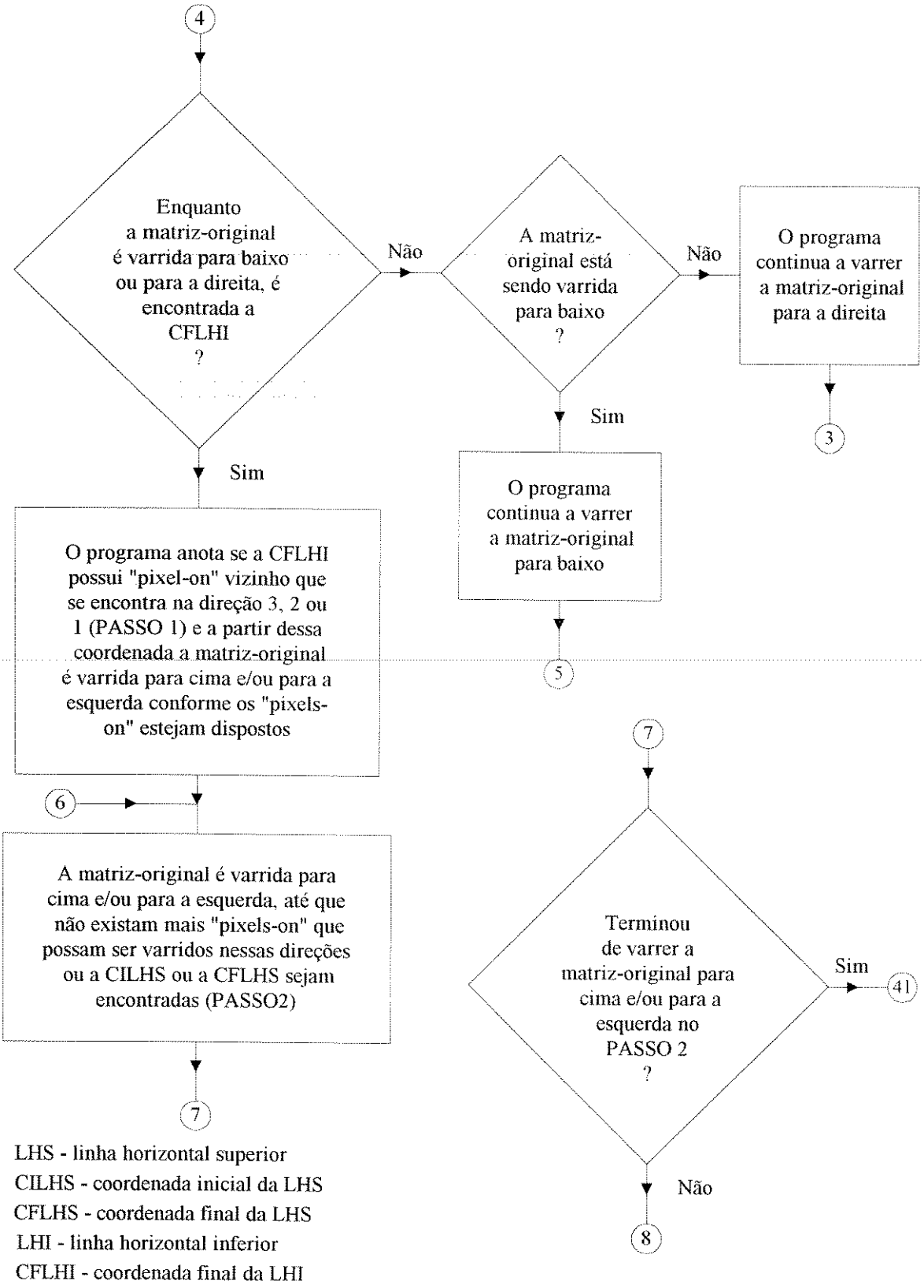


Figura 4.19(d) - Quarta parte do fluxograma da função 0-8-6-3-5-1-2-4.

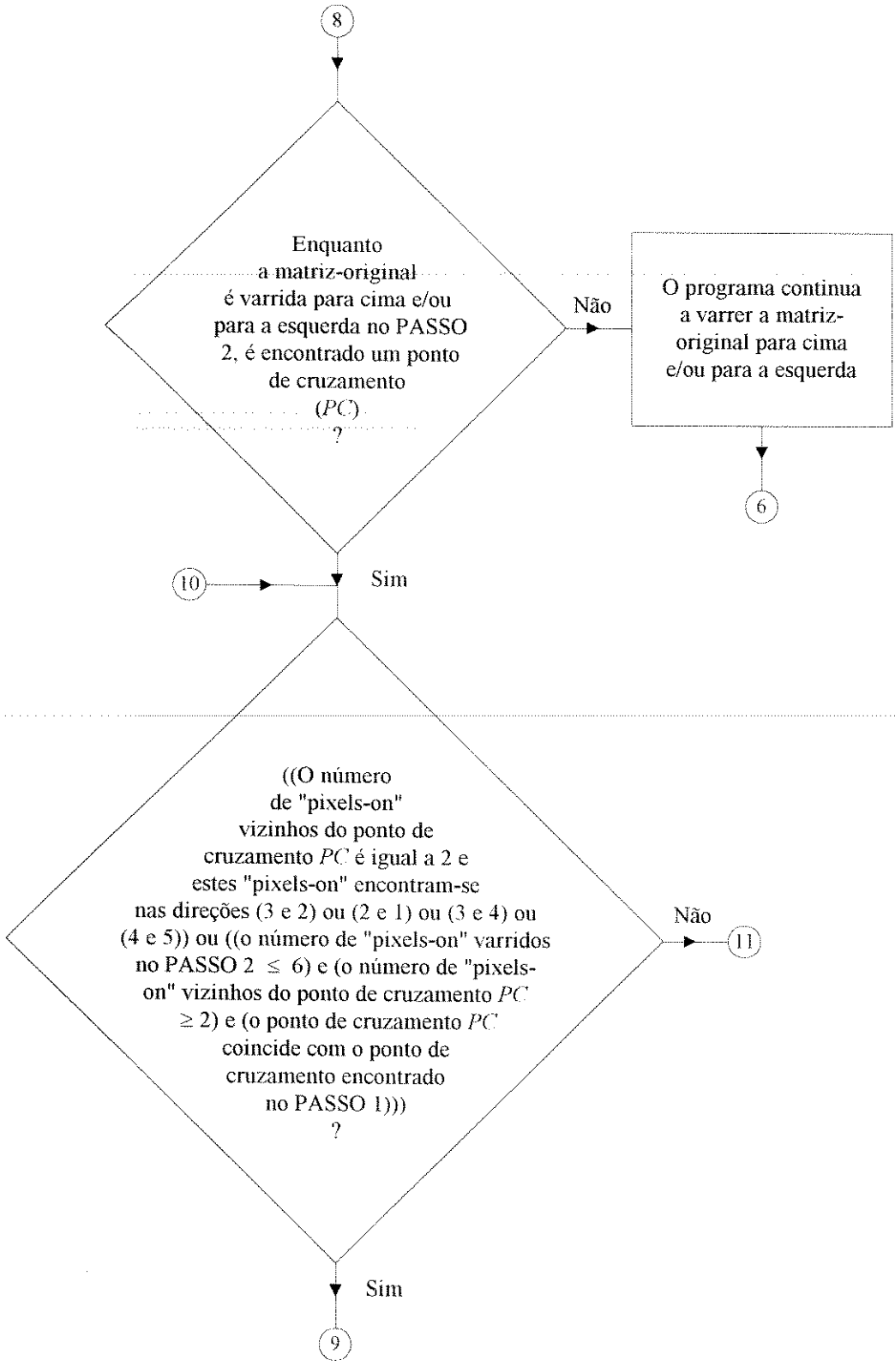


Figura 4.19(e) - Quinta parte do fluxograma da função 0-8-6-3-5-1-2-4.

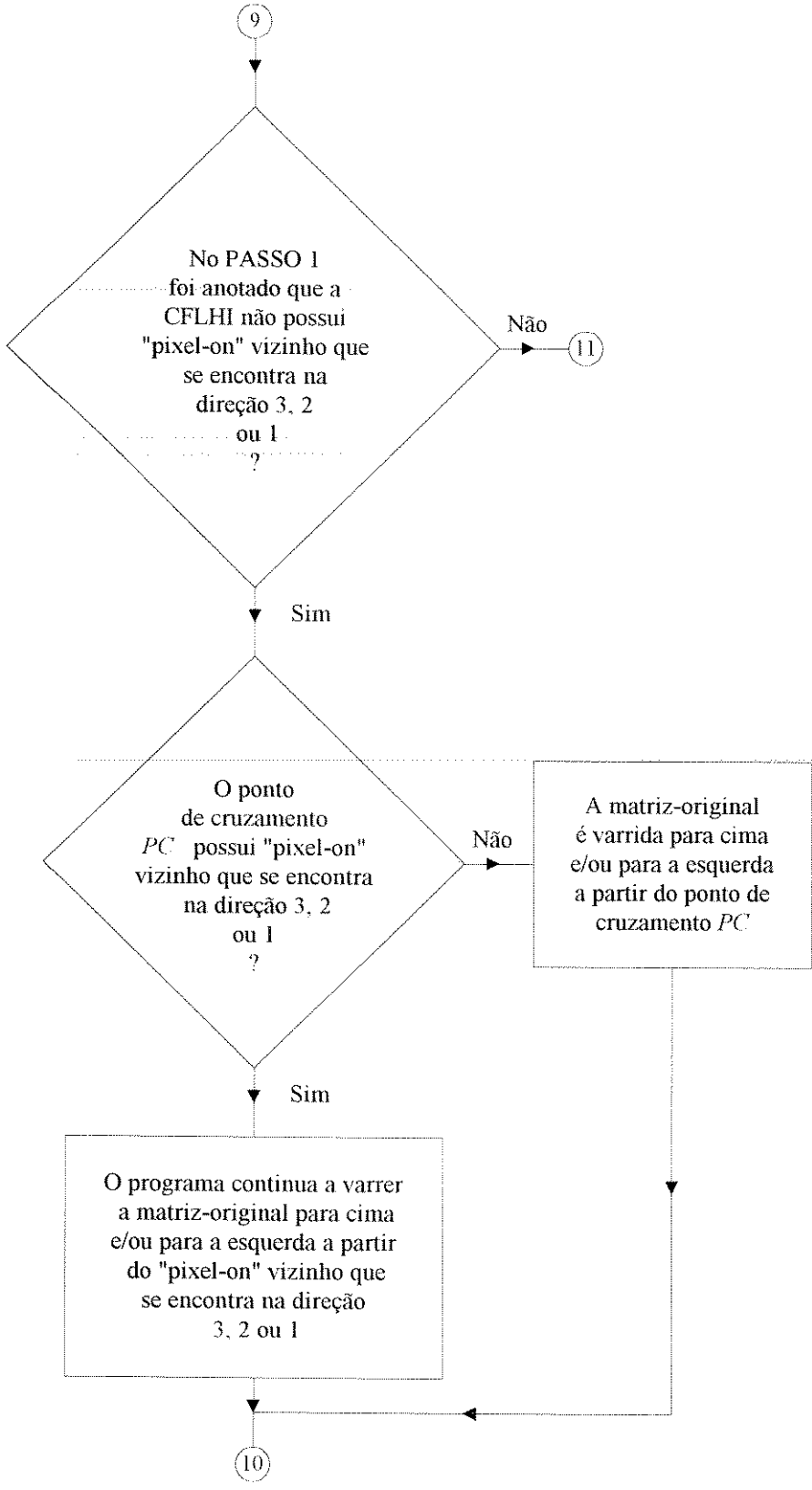


Figura 4.19(f) - Sexta parte do fluxograma da função 0-8-6-3-5-1-2-4.

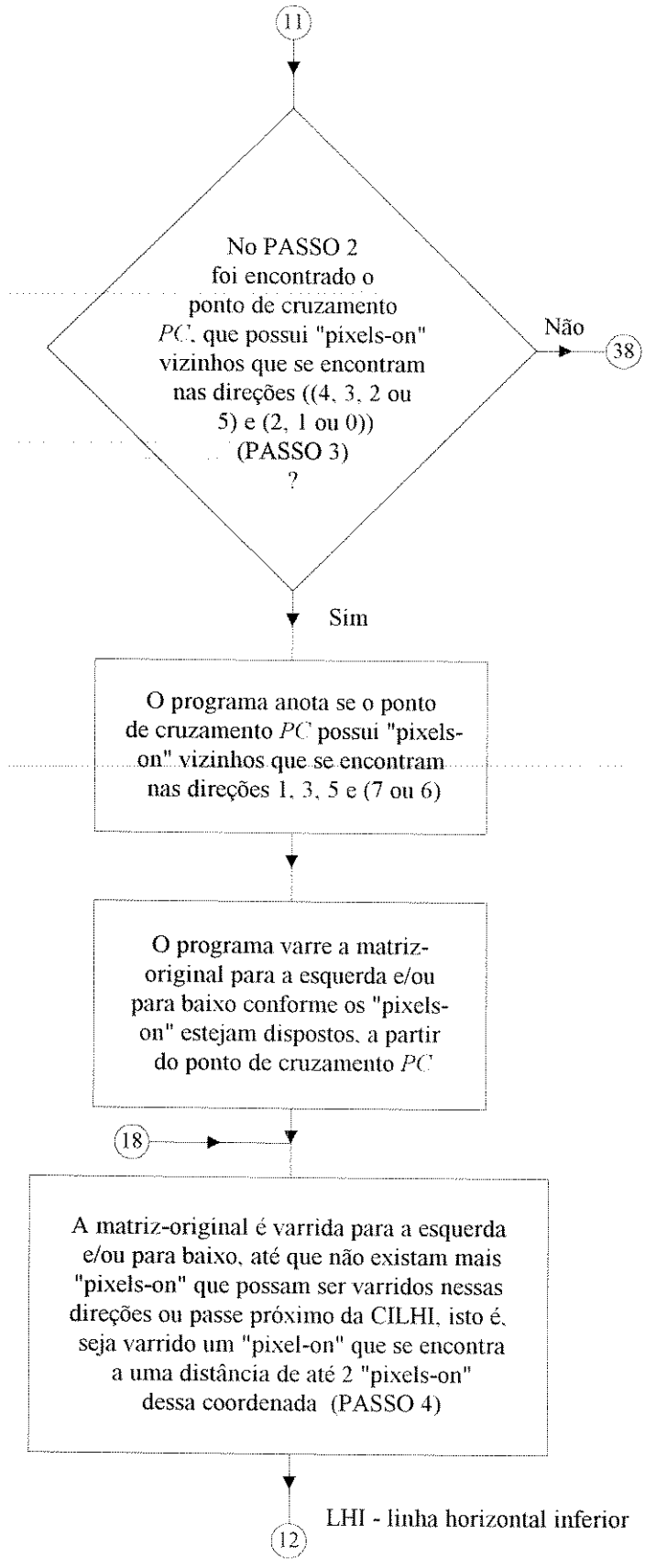


Figura 4.19(g) - Sétima parte do fluxograma da função 0-8-6-3-5-1-2-4.

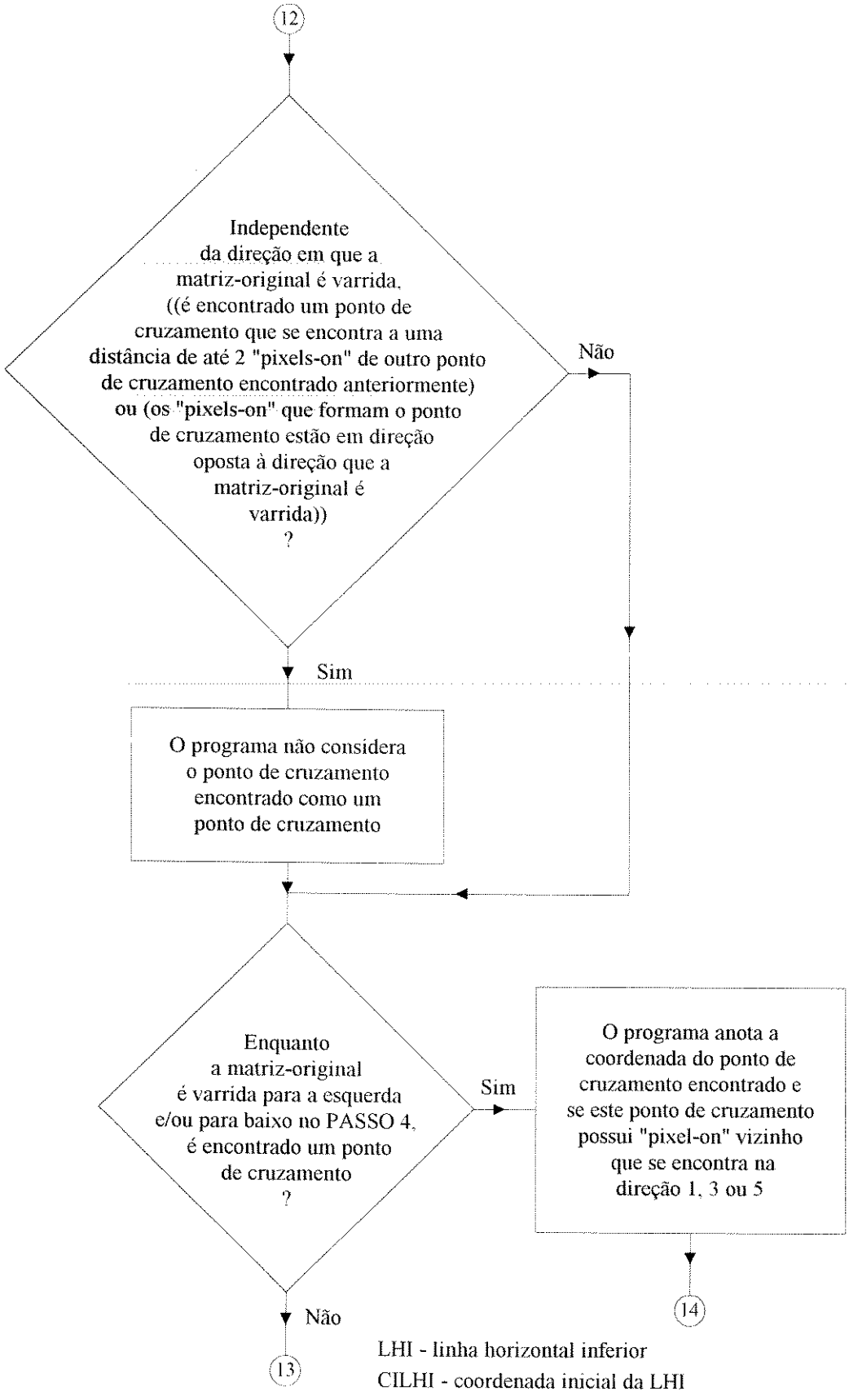


Figura 4.19(h) - Oitava parte do fluxograma da função 0-8-6-3-5-1-2-4.

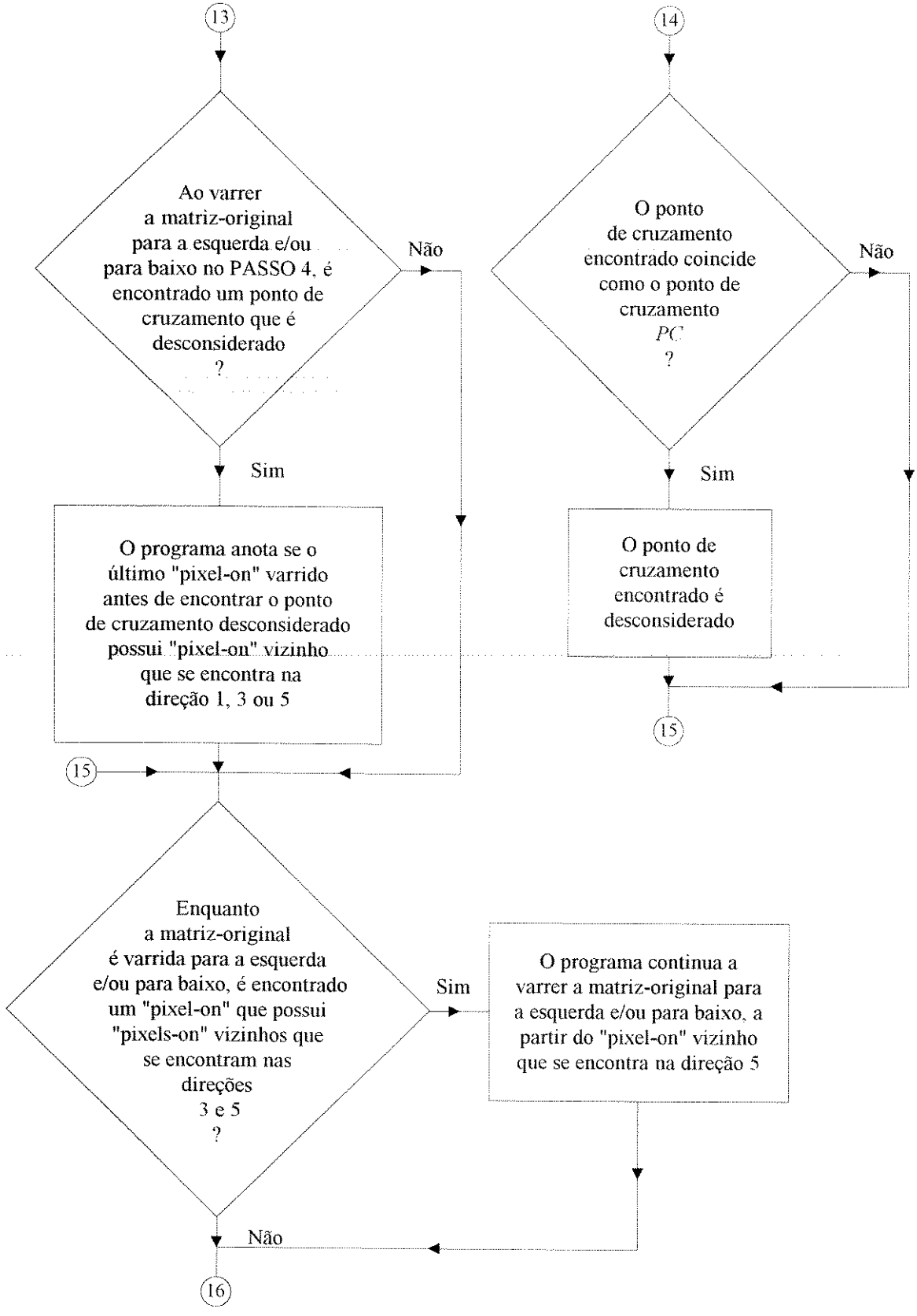


Figura 4.19(i) - Nona parte do fluxograma da função 0-8-6-3-5-1-2-4.

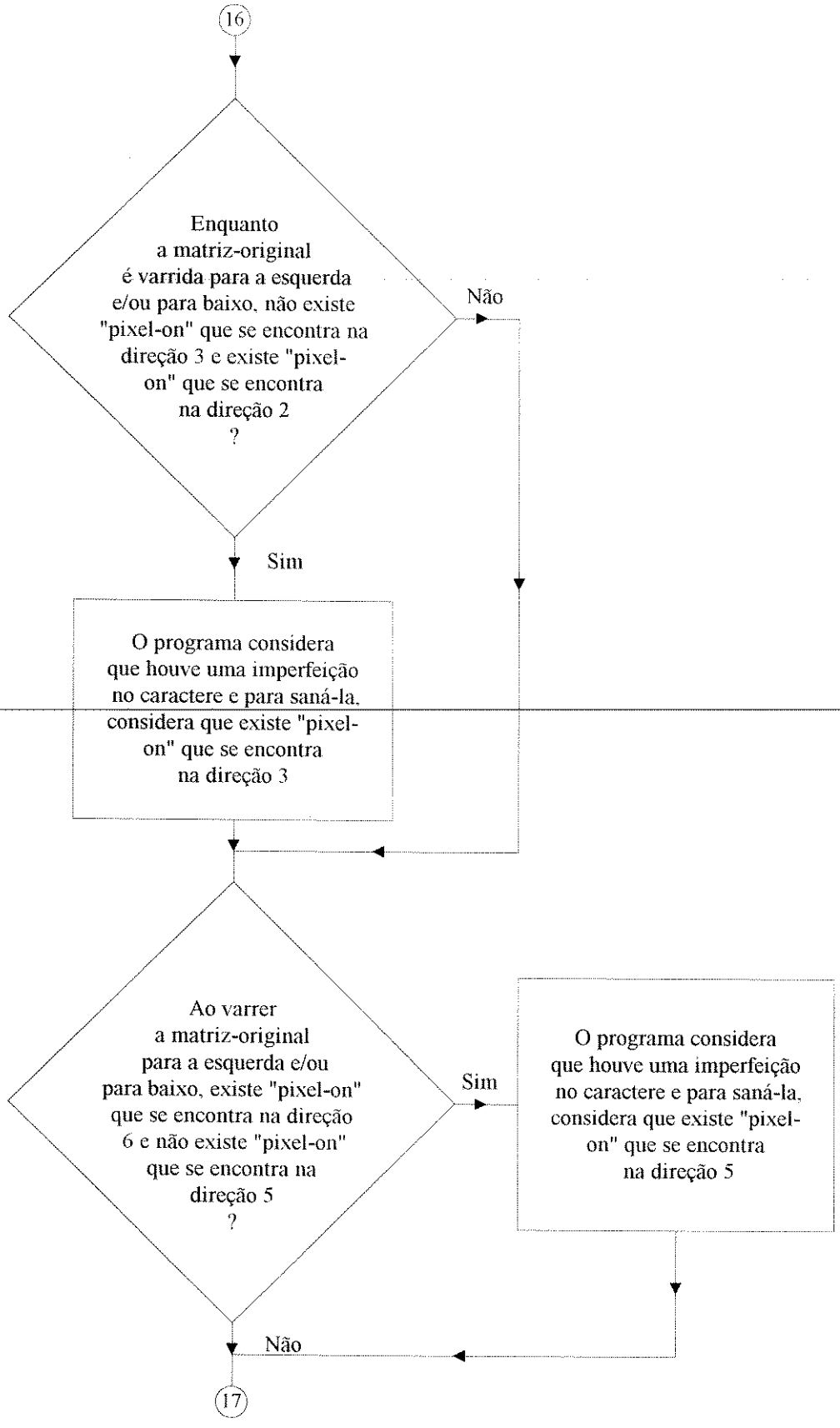


Figura 4.19(j) - Décima parte do fluxograma da função 0-8-6-3-5-1-2-4.

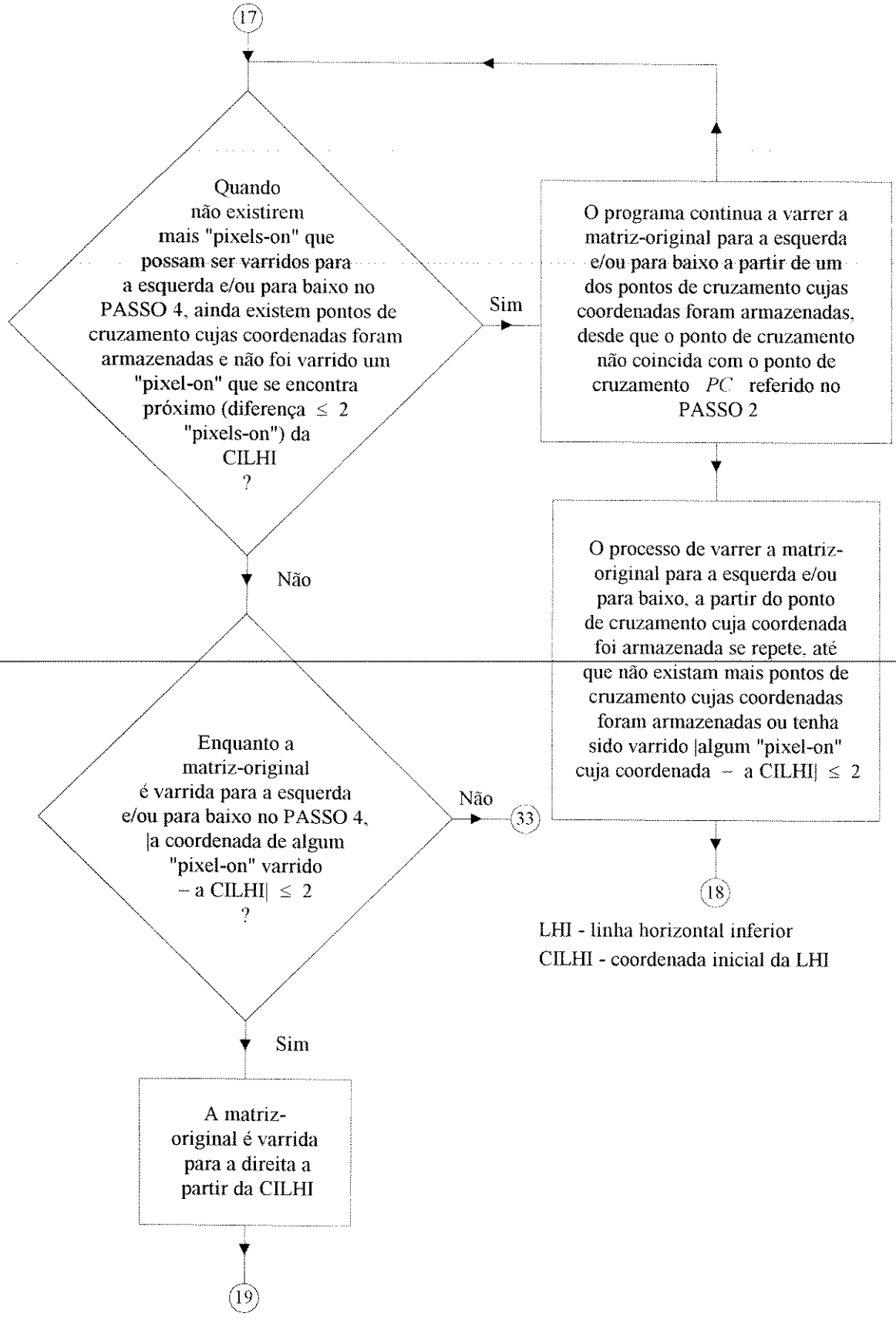


Figura 4.19(k) - Décima primeira parte do fluxograma da função 0-8-6-3-5-1-2-4.

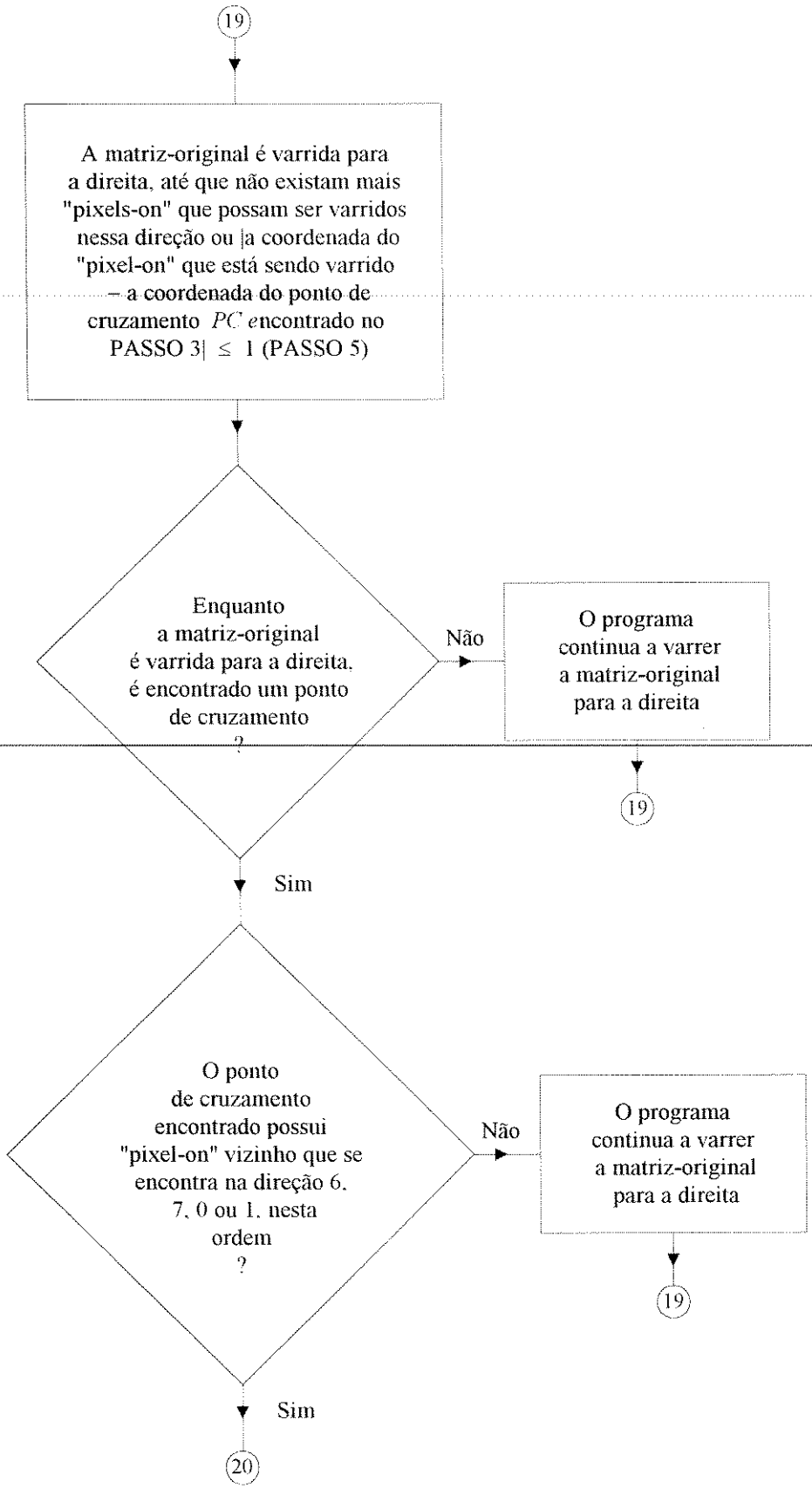
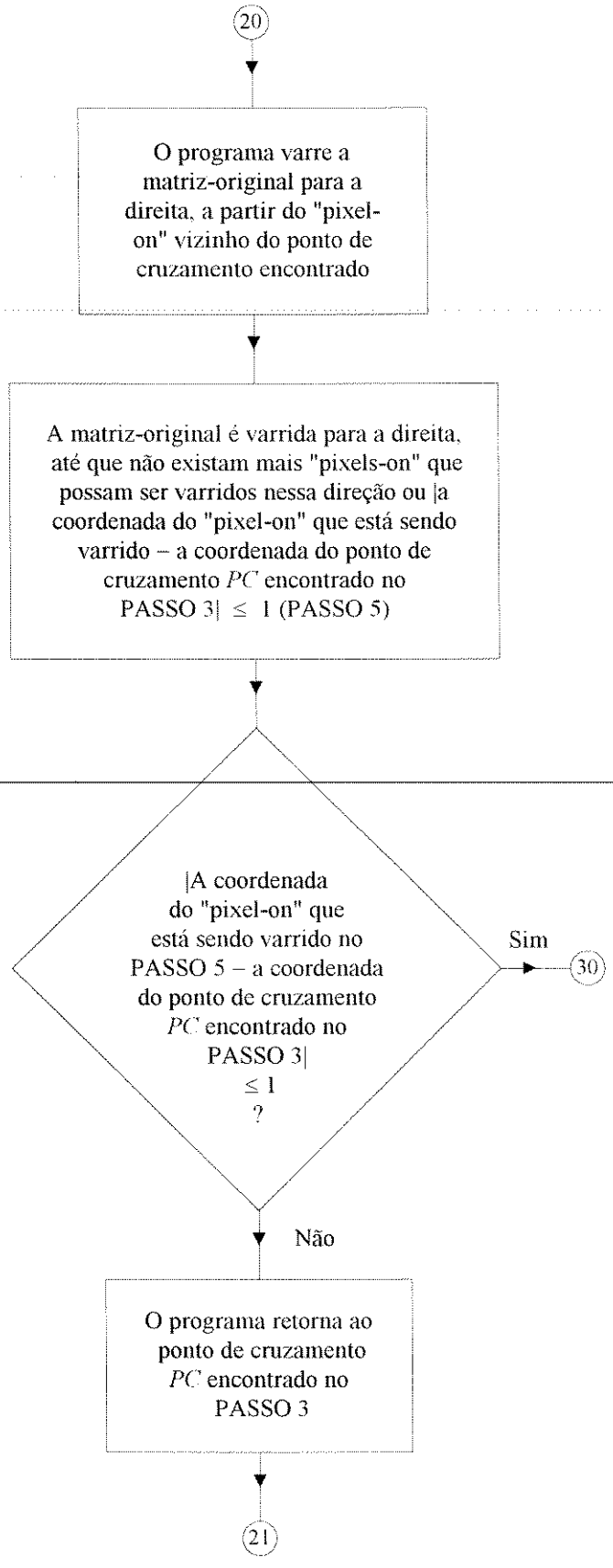


Figura 4.19(l) - Décima segunda parte do fluxograma da função 0-8-6-3-5-1-2-4.



igura 4.19(m) - Décima terceira parte do fluxograma da função 0-8-6-3-5-1-2-4.

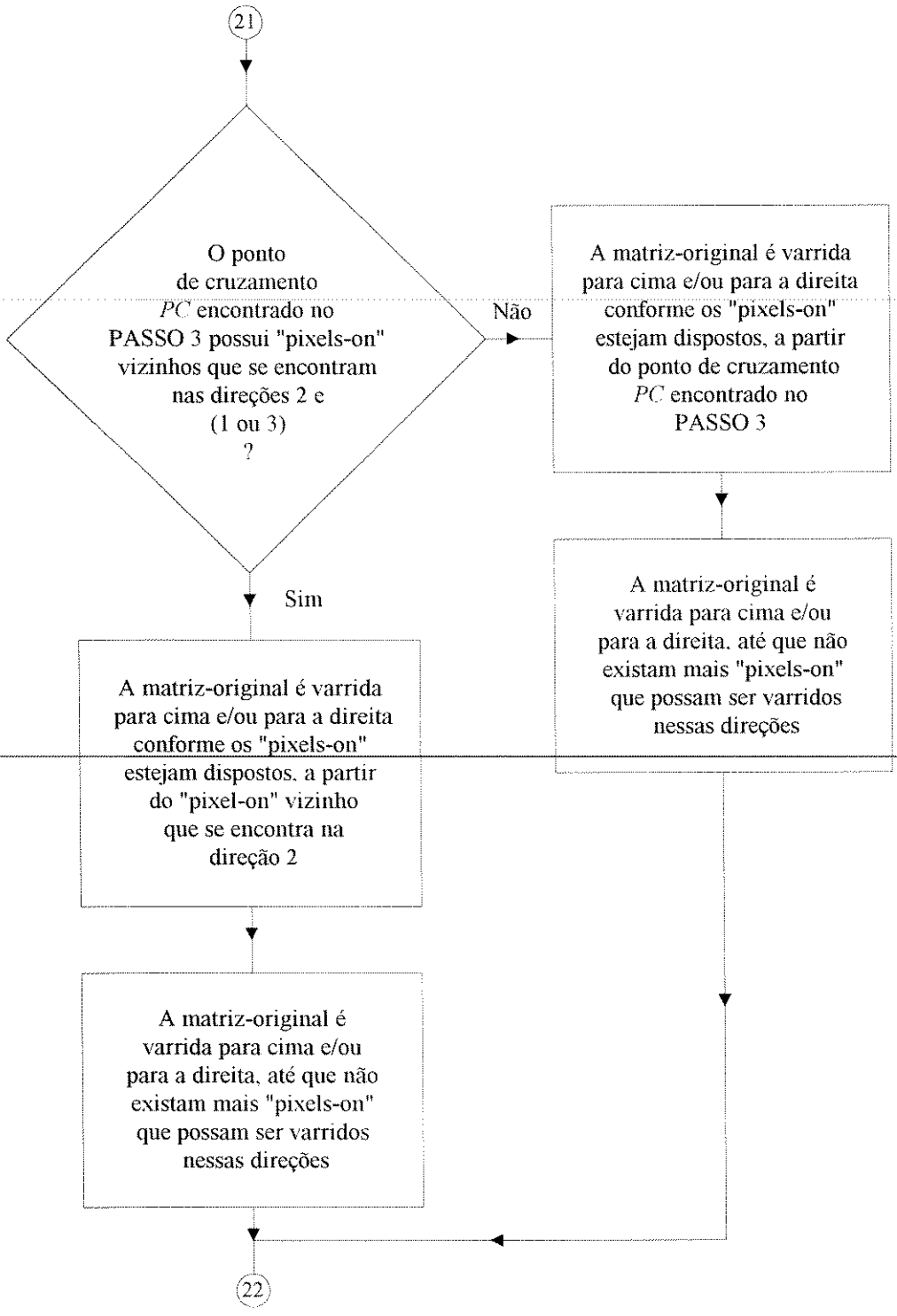
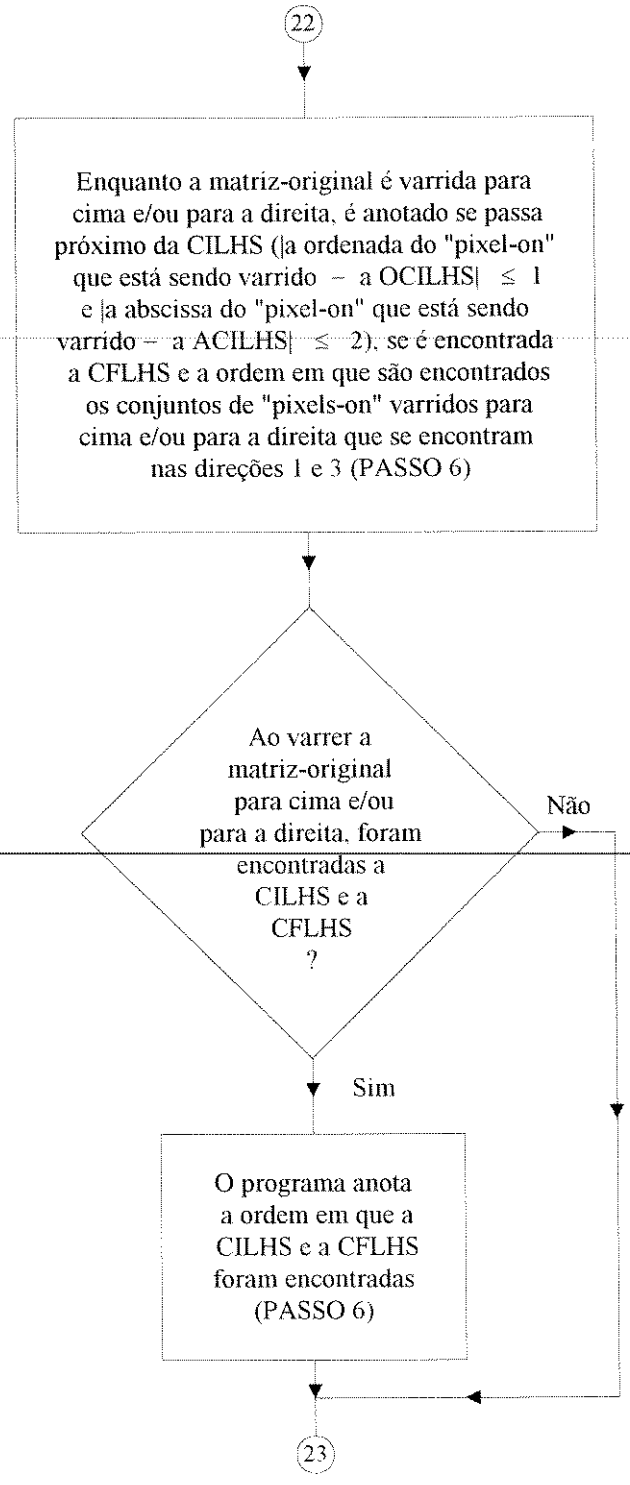


Figura 4.19(n) - Décima quarta parte do fluxograma da função 0-8-6-3-5-1-2-4.



LHS - linha horizontal superior
CILHS - coordenada inicial da LHS
ACILHS - abscissa da coordenada inicial da LHS
OCILHS - ordenada da coordenada inicial da LHS
CFLHS - coordenada final da LHS

Figura 4.19(o) - Décima quinta parte do fluxograma da função 0-8-6-3-5-1-2-4.

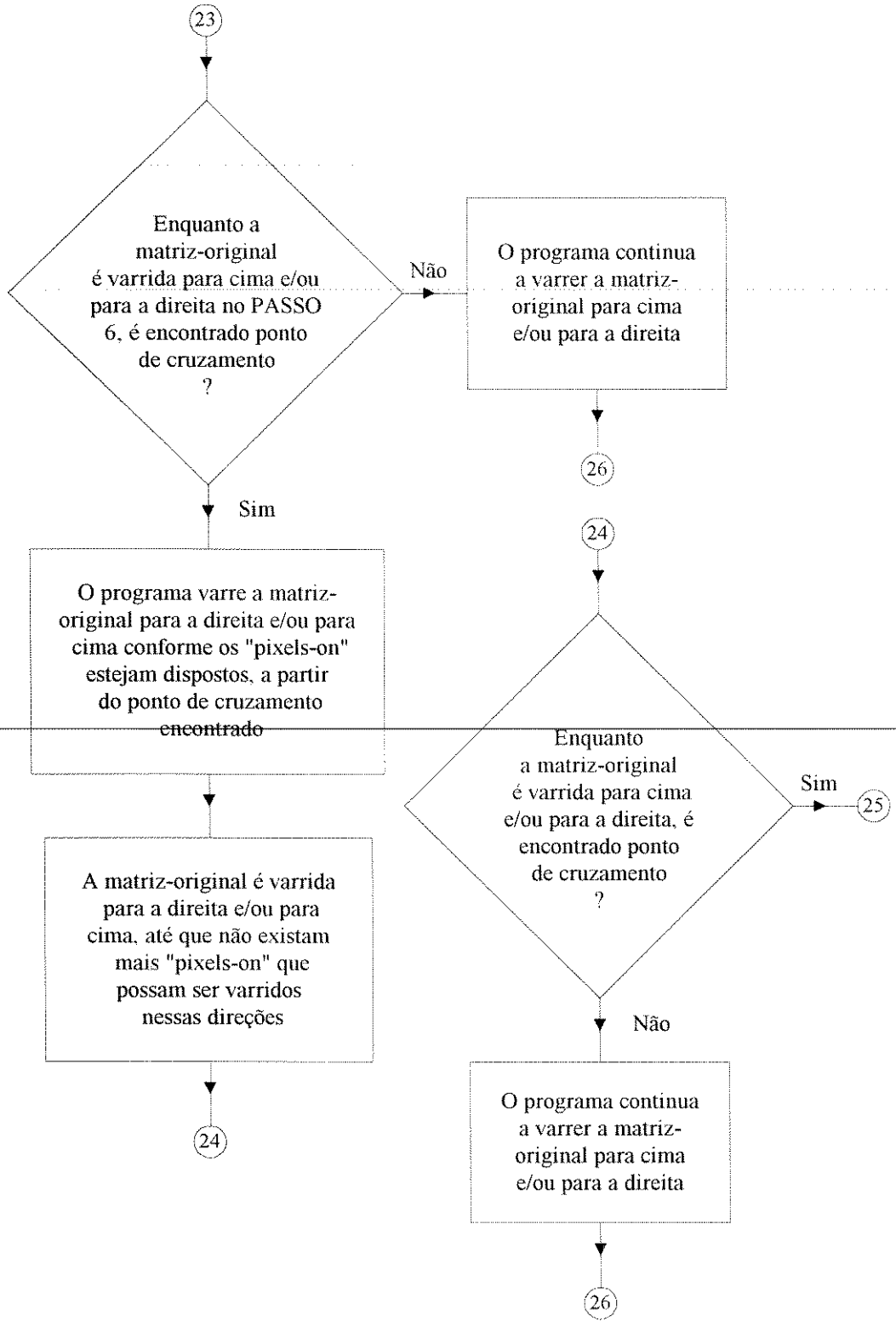


Figura 4.19(p) - Décima sexta parte do fluxograma da função 0-8-6-3-5-1-2-4.

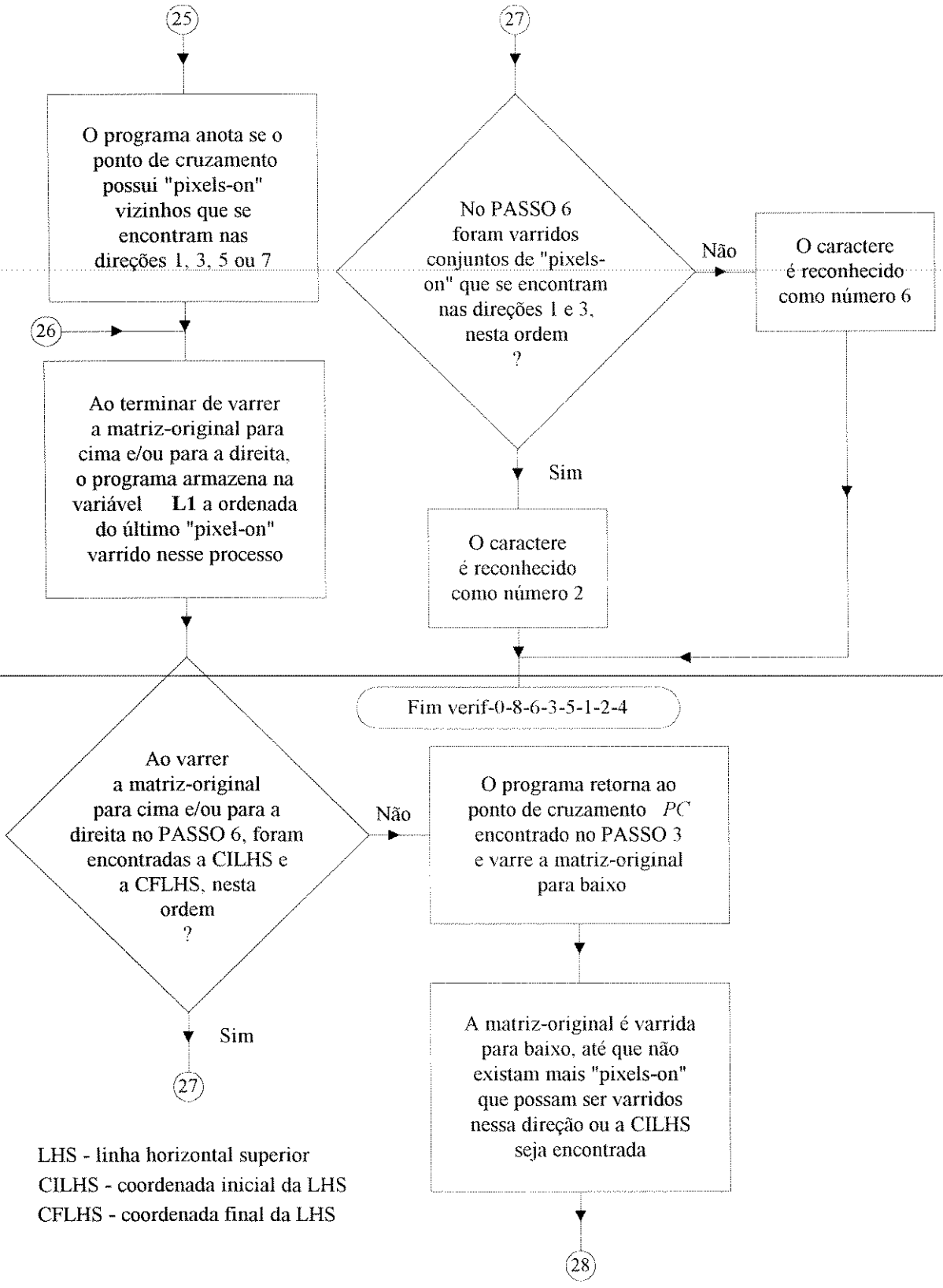


Figura 4.19(q) - Décima sétima parte do fluxograma da função 0-8-6-3-5-1-2-4.

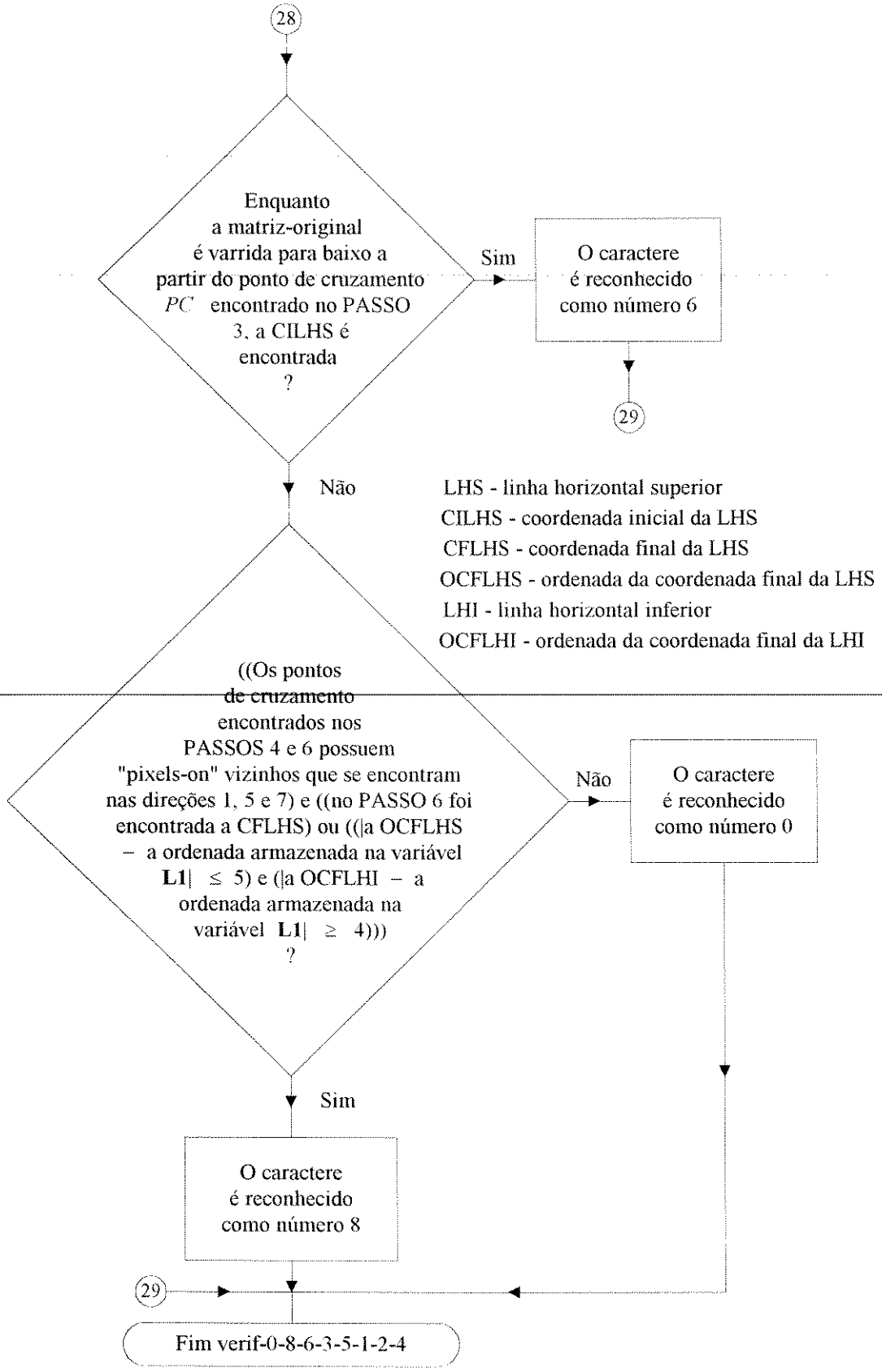


Figura 4.19(r) - Décima oitava parte do fluxograma da função 0-8-6-3-5-1-2-4.

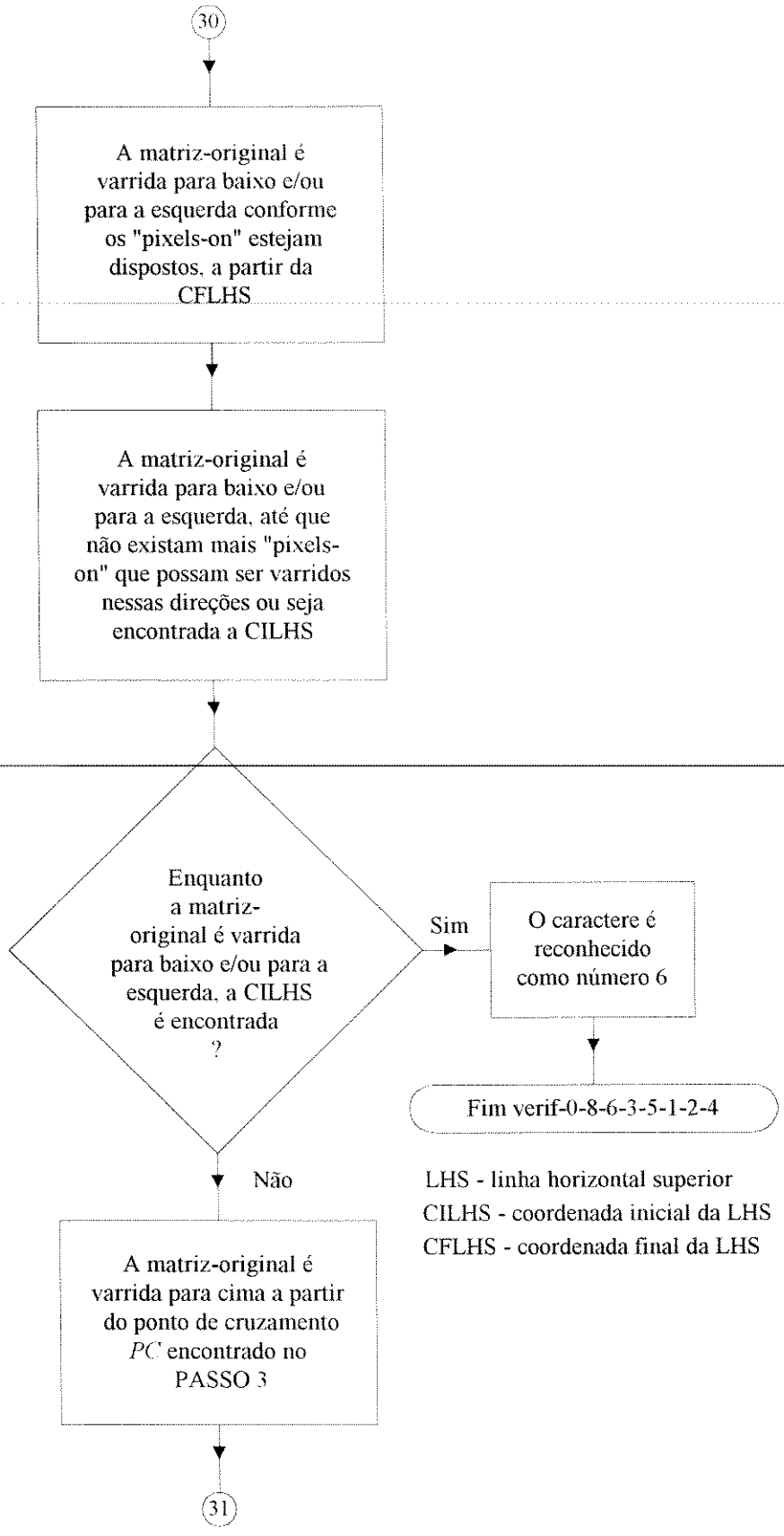


Figura 4.19(s) - Décima nona parte do fluxograma da função 0-8-6-3-5-1-2-4.

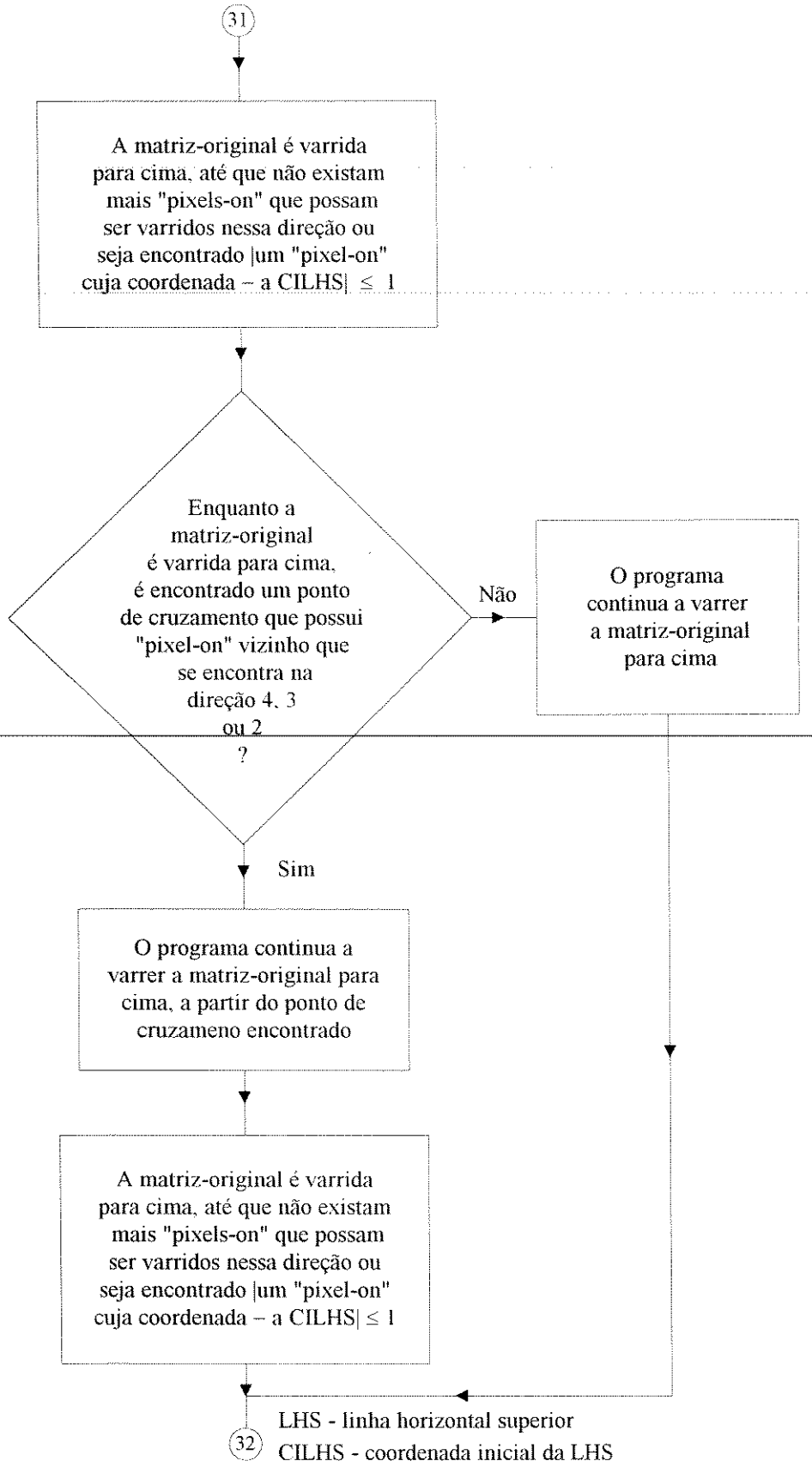


Figura 4.19(t) - Vigésima parte do fluxograma da função 0-8-6-3-5-1-2-4.

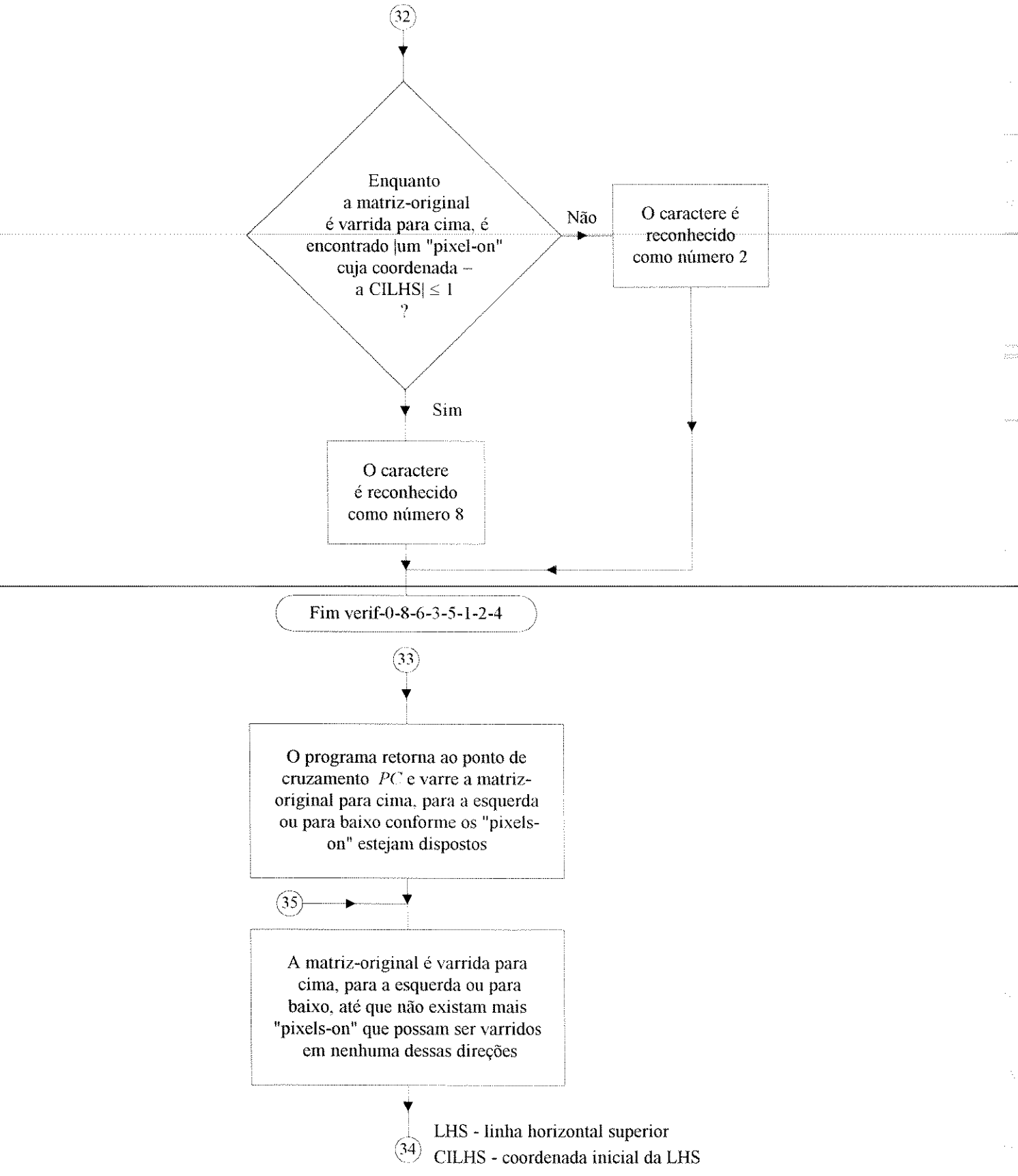


Figura 4.19(u) - Vigésima primeira parte do fluxograma da função 0-8-6-3-5-1-2-4.

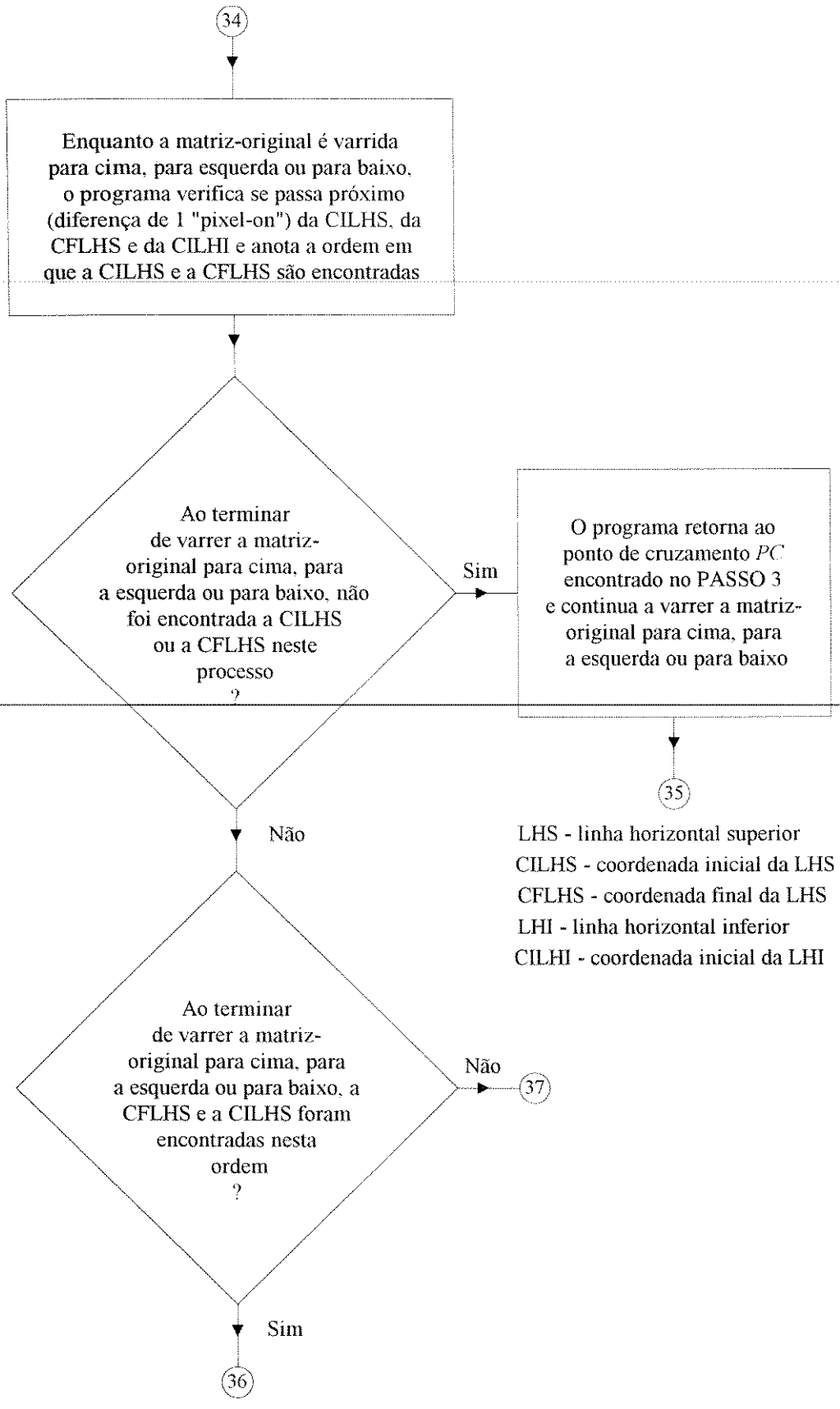


Figura 4.19(v) - Vigésima segunda parte do fluxograma da função 0-8-6-3-5-1-2-4.

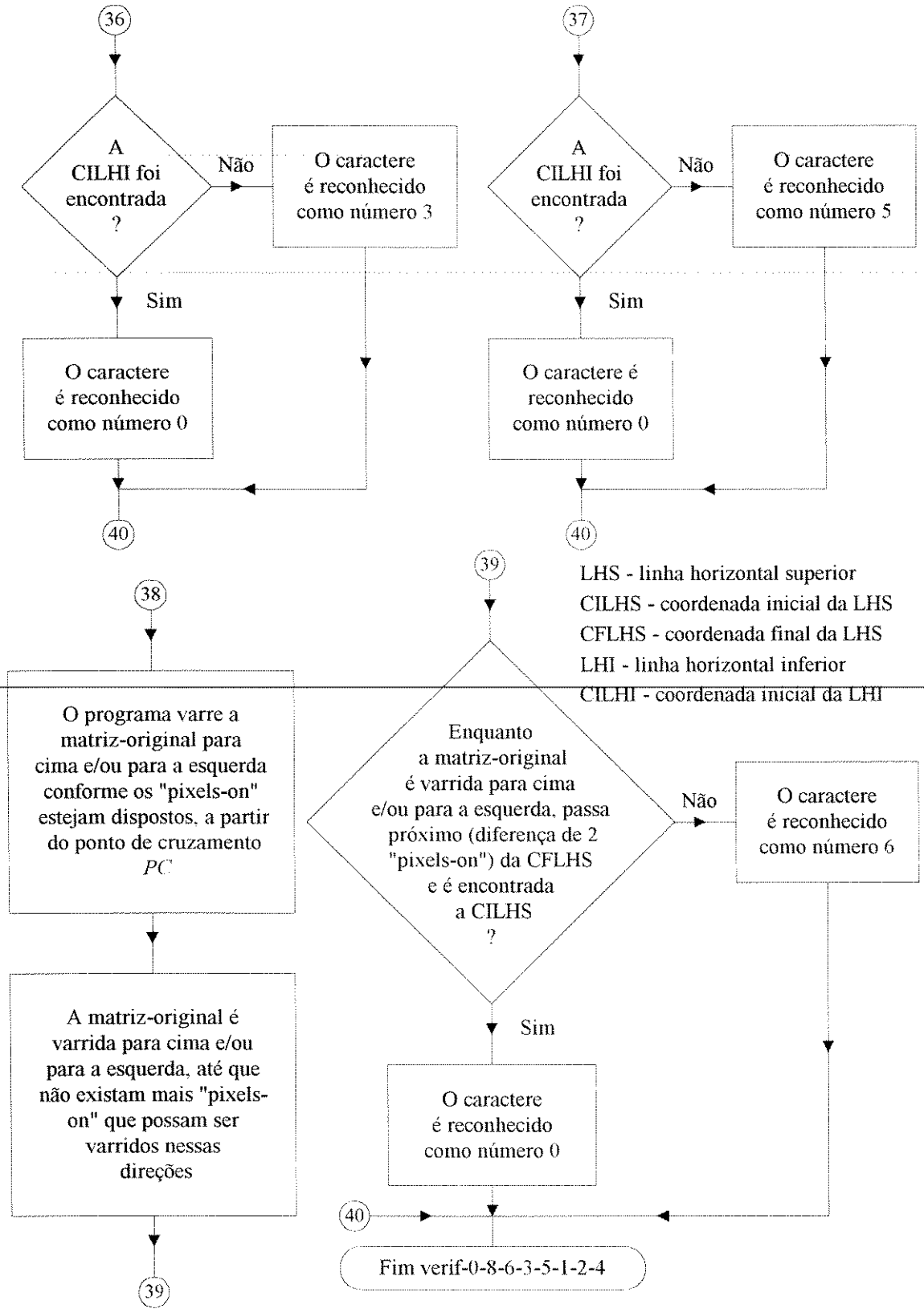


Figura 4.19(w) - Vigésima terceira parte do fluxograma da função 0-8-6-3-5-1-2-4.

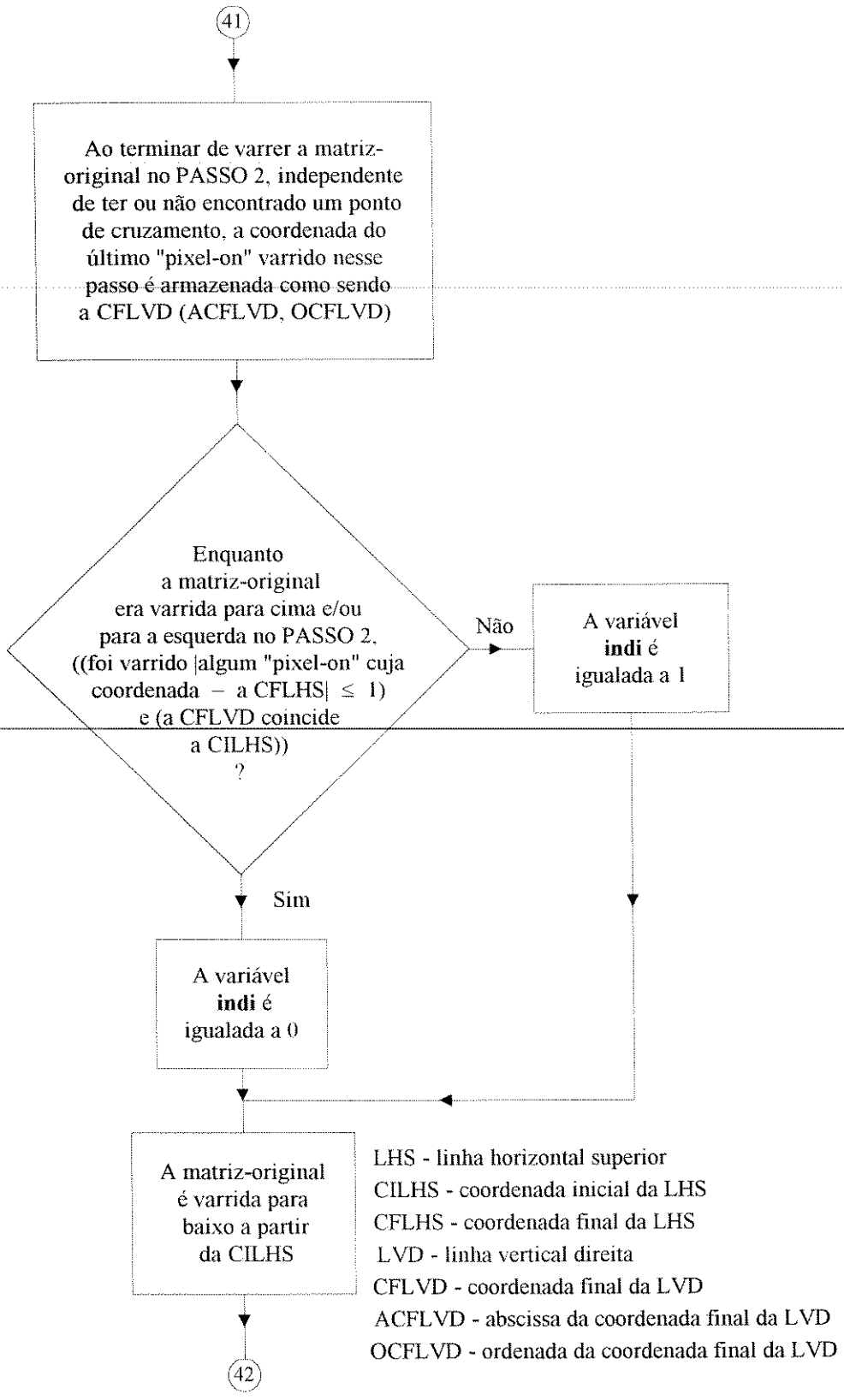
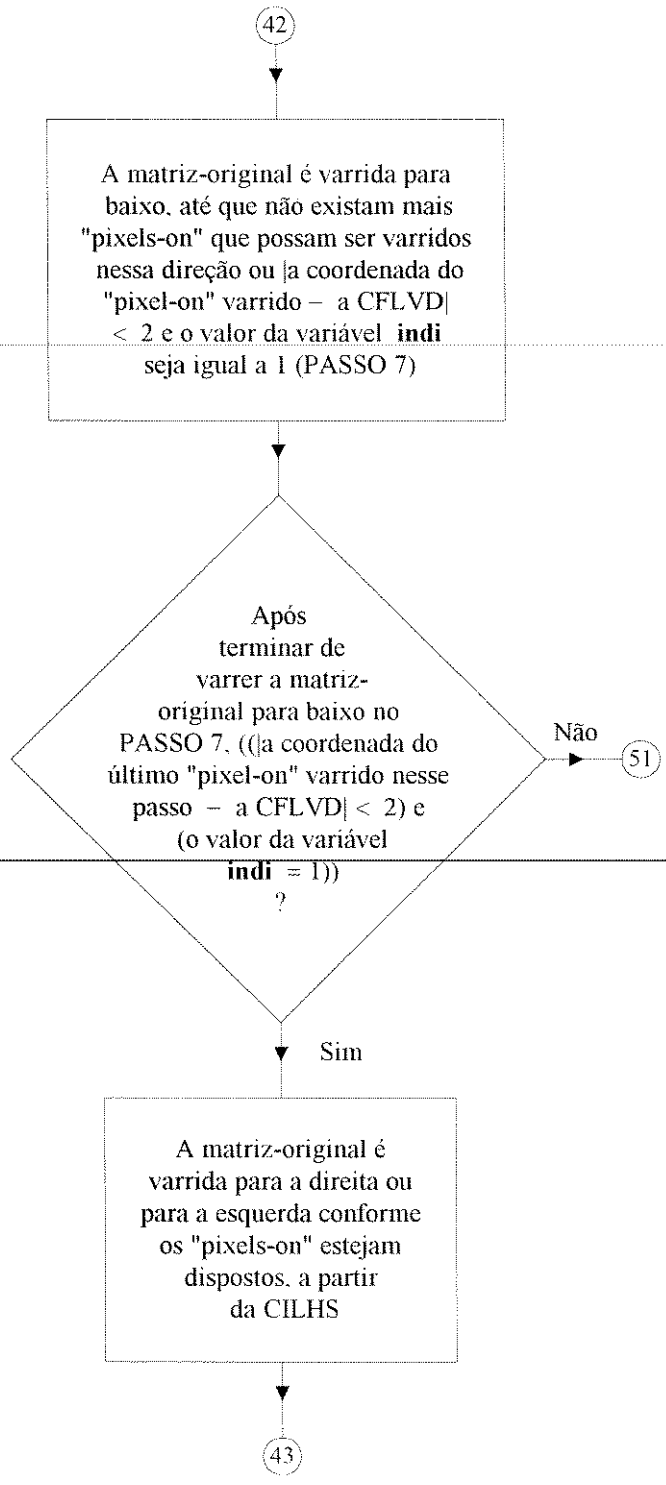


Figura 4.19(x) - Vigésima quarta parte do fluxograma da função 0-8-6-3-5-1-2-4.



LHS - linha horizontal superior
CILHS - coordenada inicial da LHS
LVD - linha vertical direita
CFLVD - coordenada final da LVD

Figura 4.19(y) - Vigésima quinta parte do fluxograma da função 0-8-6-3-5-1-2-4.

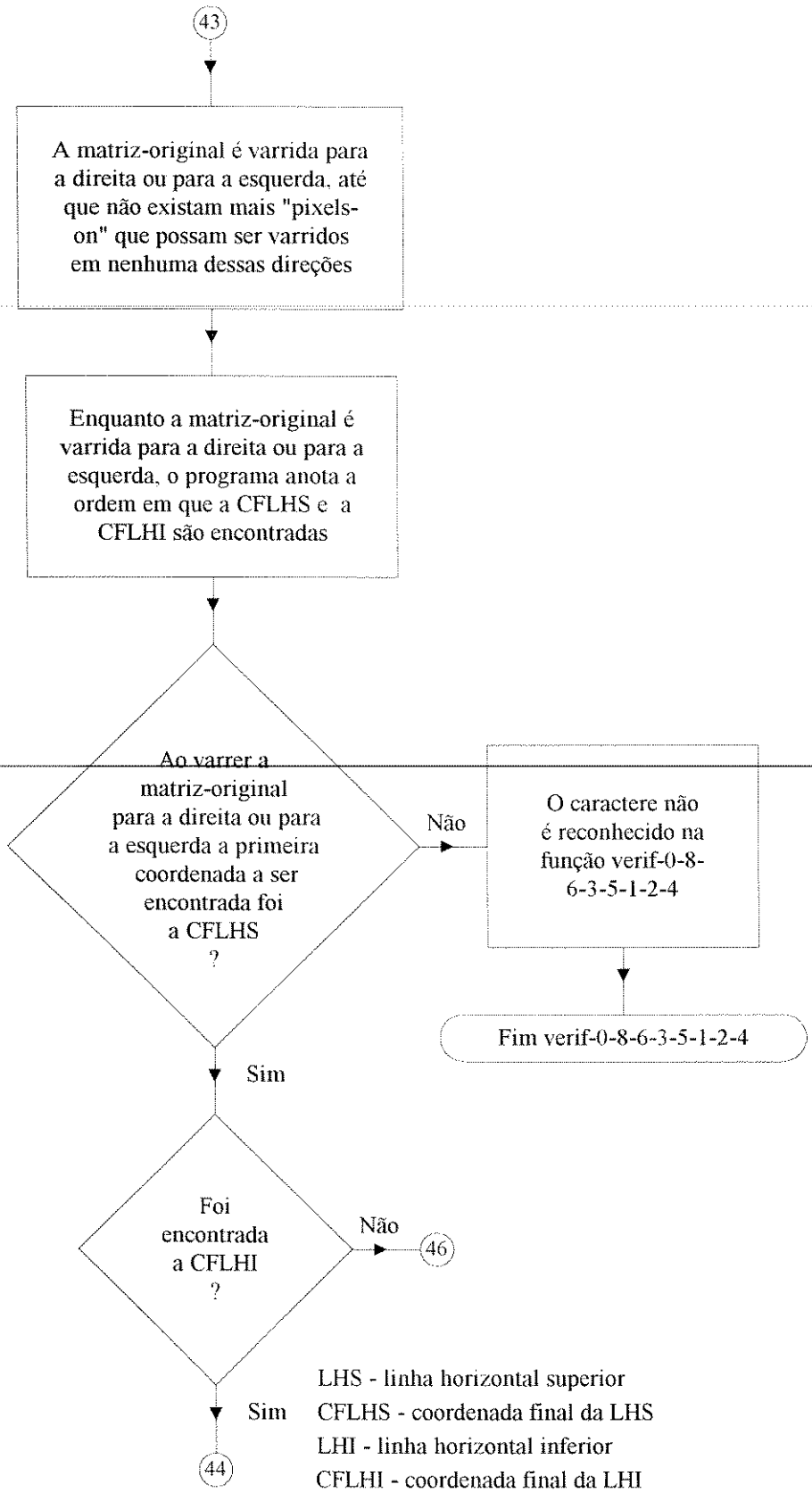
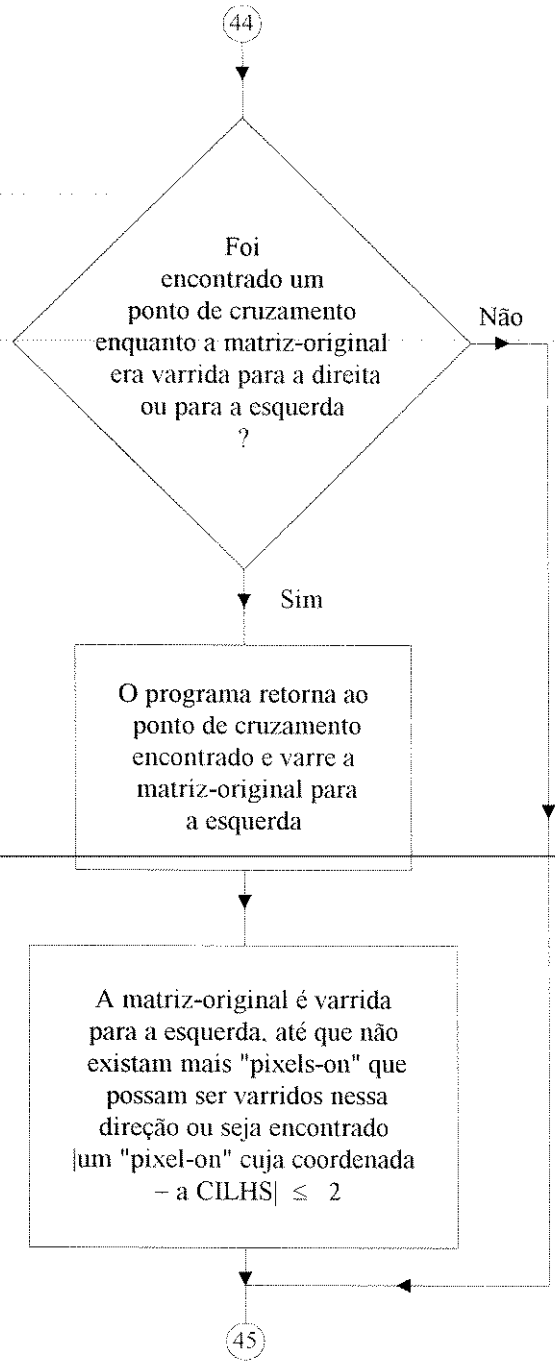


Figura 4.19(z) - Vigésima sexta parte do fluxograma da função 0-8-6-3-5-1-2-4.



LHS - linha horizontal superior
CILHS - coordenada inicial da LHS

Figura 4.19(a1) - Vigésima sétima parte do fluxograma da função 0-8-6-3-5-1-2-4.

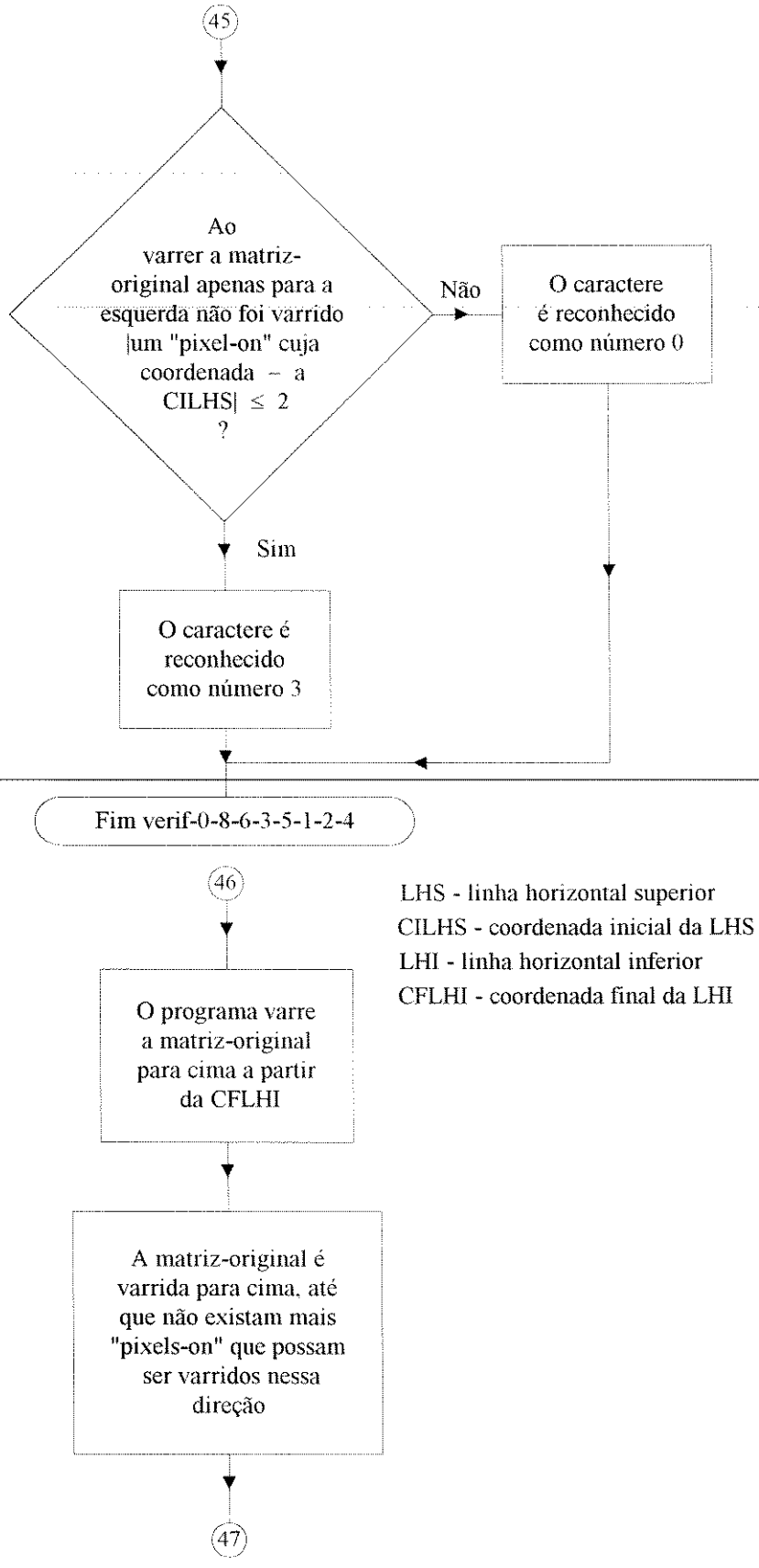


Figura 4.19(b1) - Vigésima oitava parte do fluxograma da função 0-8-6-3-5-1-2-4.

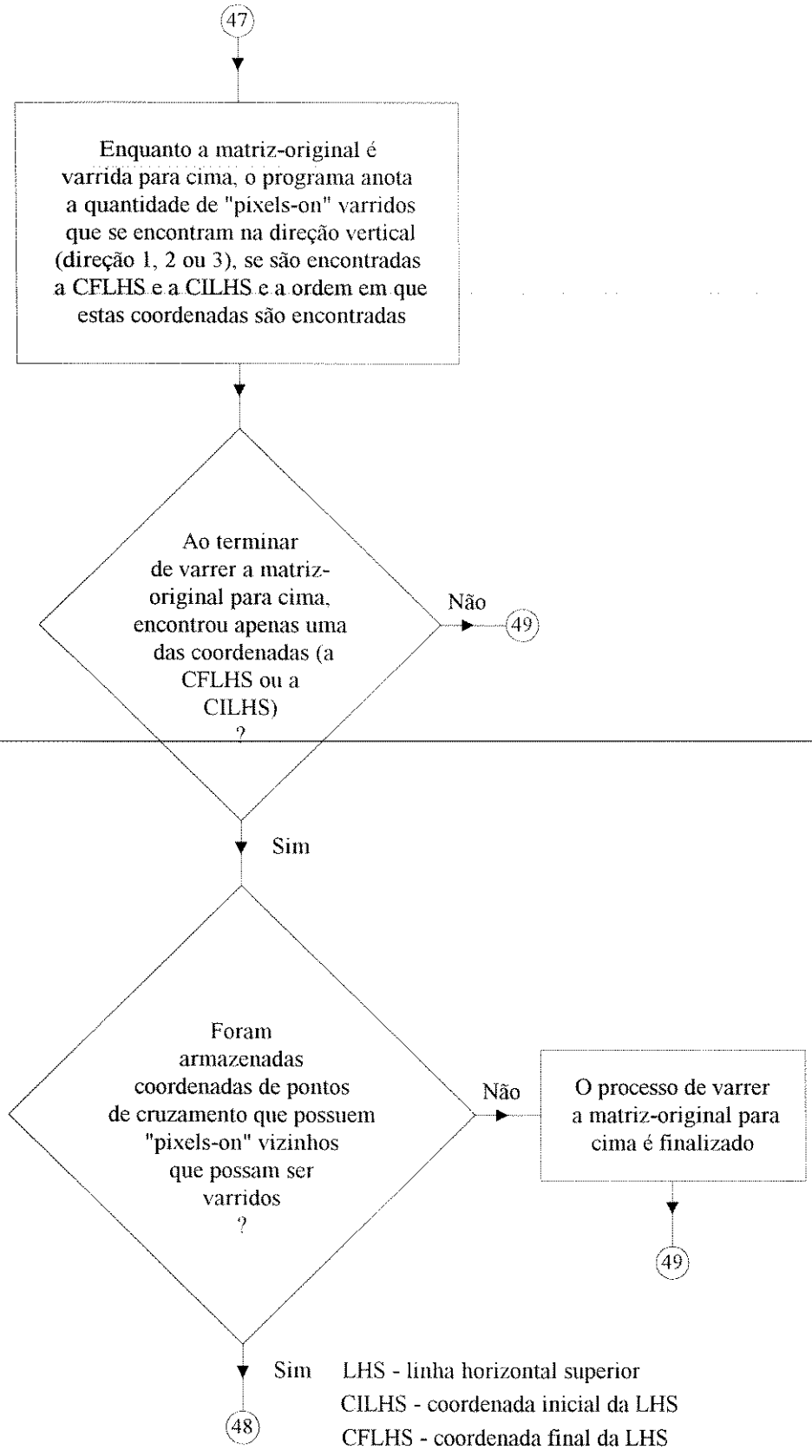


Figura 4.19(c1) - Vigésima nona parte do fluxograma da função 0-8-6-3-5-1-2-4.

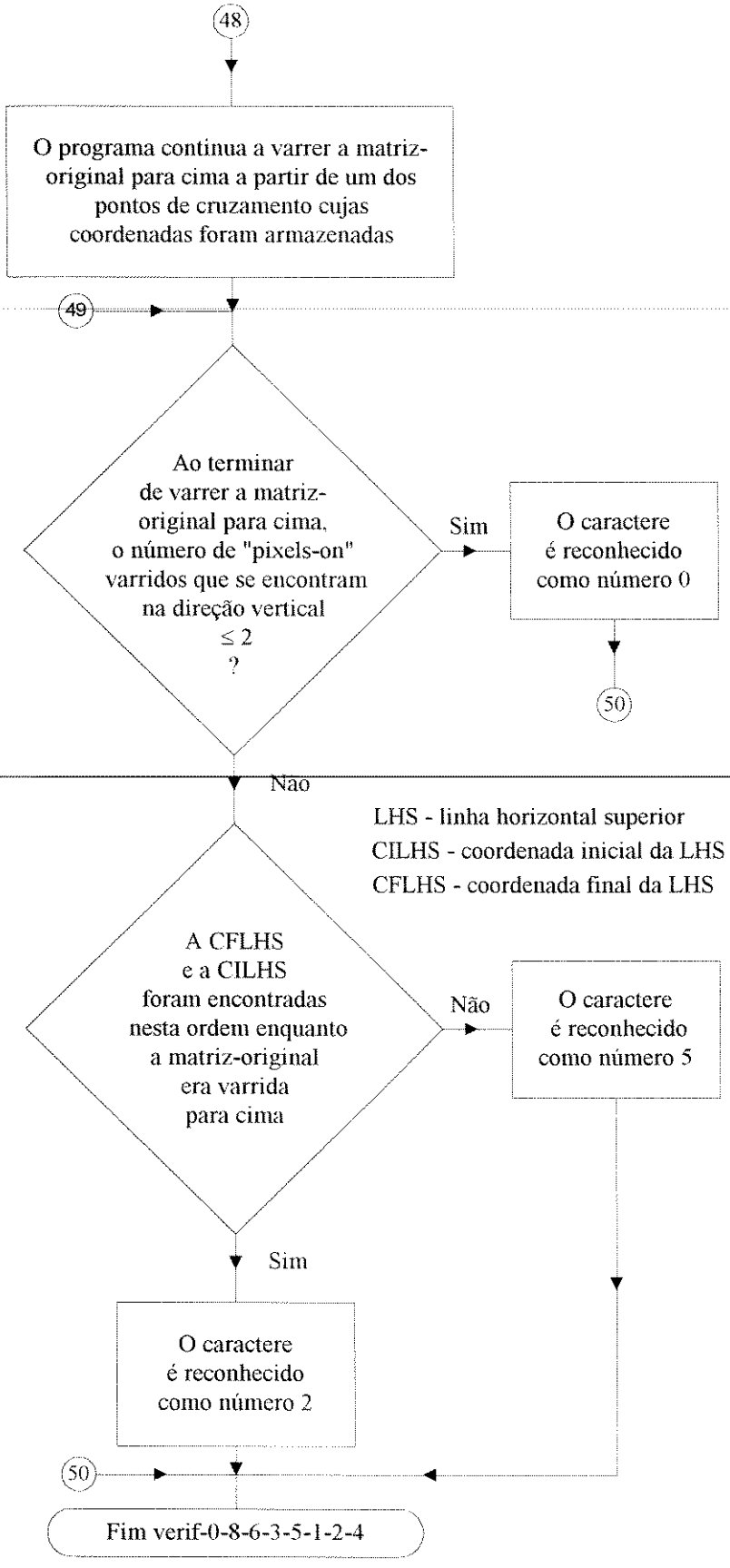
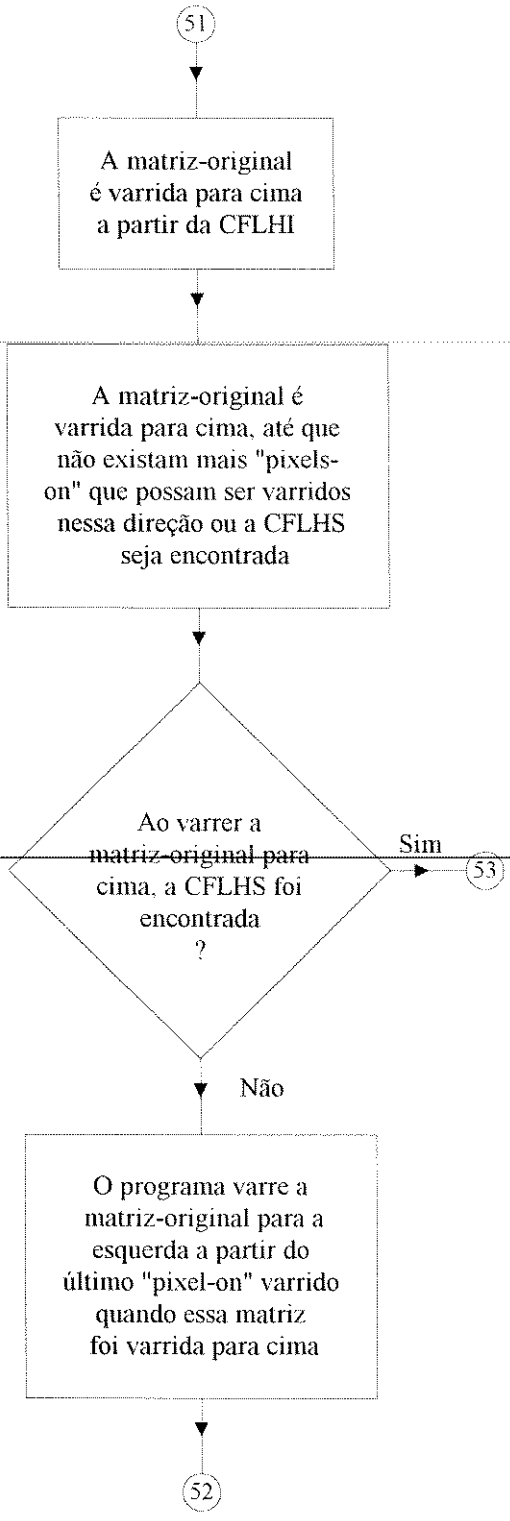


Figura 4.19(d1) - Trigésima parte do fluxograma da função 0-8-6-3-5-1-2-4.



LHS - linha horizontal superior
CFLHS - coordenada final da LHS
LHI - linha horizontal inferior
CFLHI - coordenada final da LHI

Figura 4.19(e1) - Trigésima primeira parte do fluxograma da função 0-8-6-3-5-1-2-4.

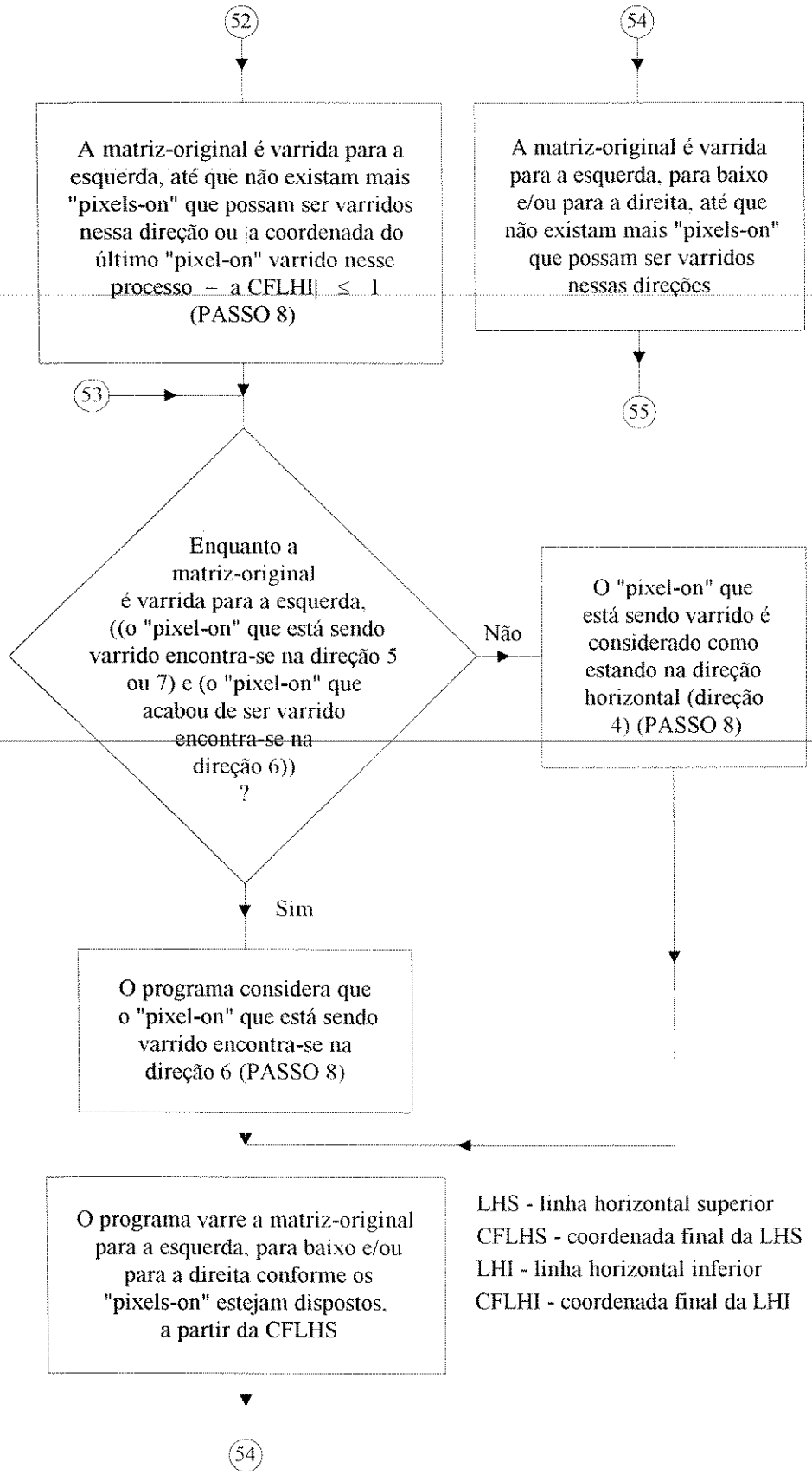


Figura 4.19(f1) - Trigésima segunda parte do fluxograma da função 0-8-6-3-5-1-2-4.

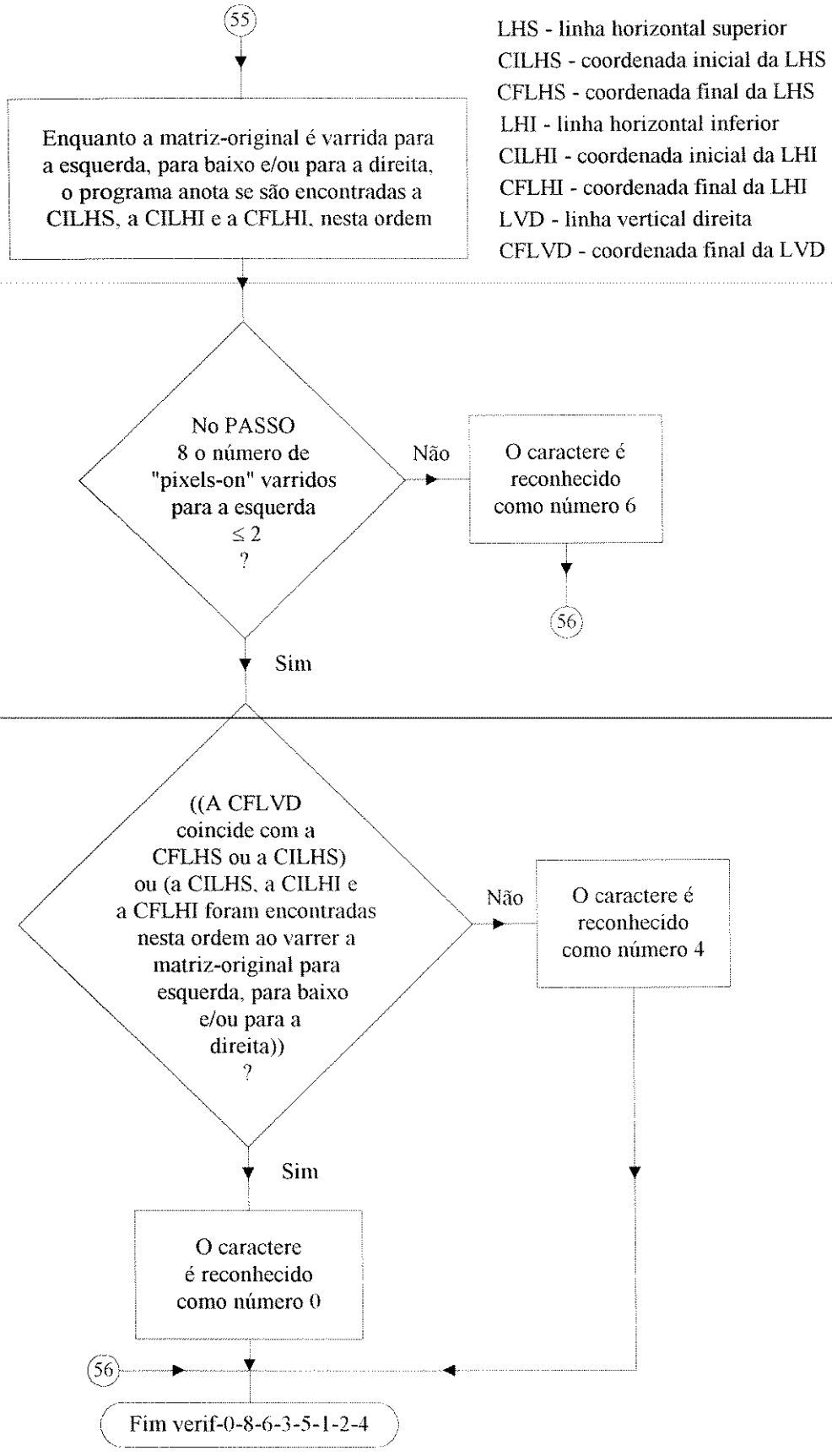


Figura 4.19(g1) - Trigésima terceira parte do fluxograma da função 0-8-6-3-5-1-2-4.

Fluxograma da função verif-2-3-5-7-9-8-0

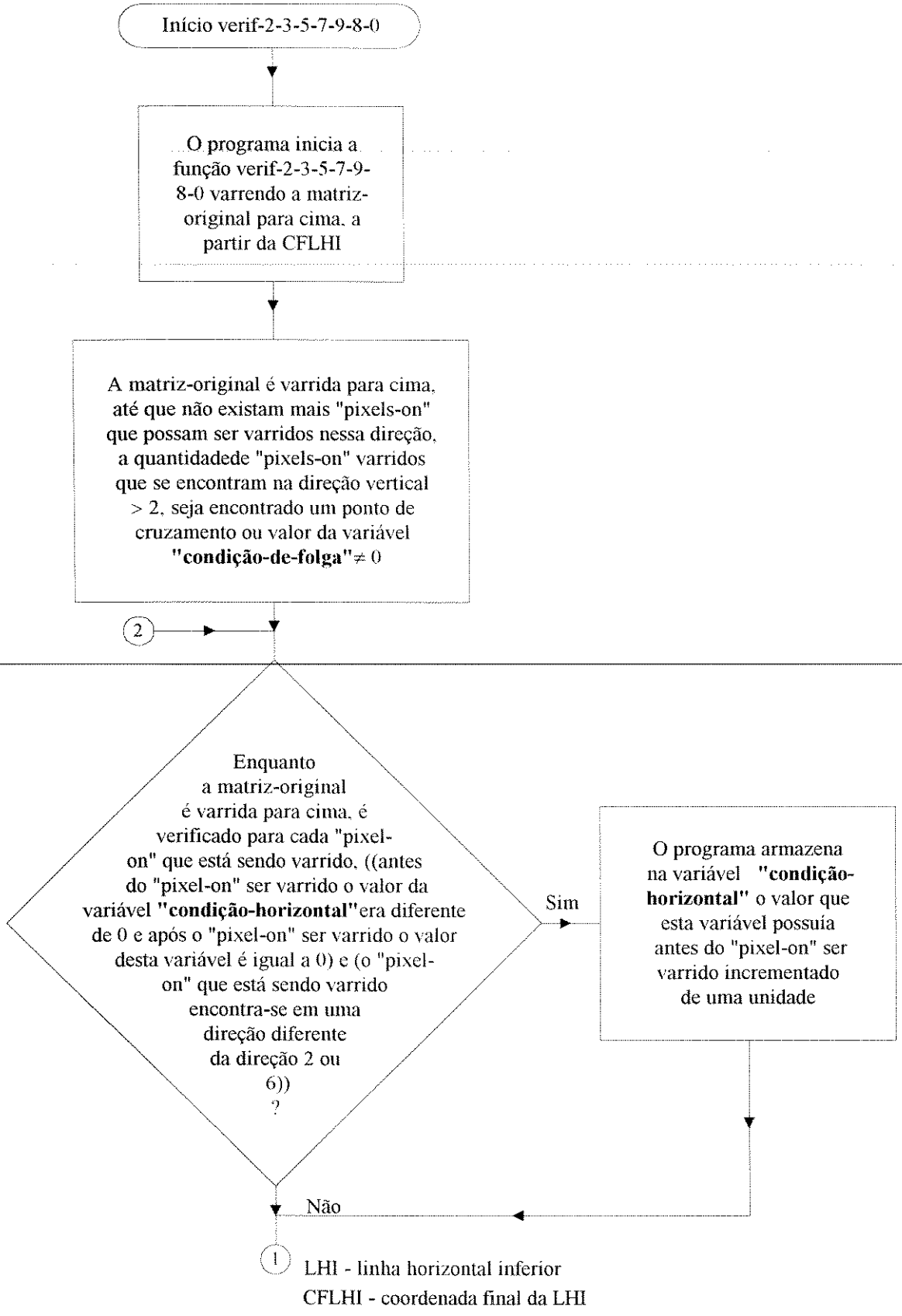


Figura 4.21(a) - Primeira parte do fluxograma da função verif-2-3-5-7-9-8-0.

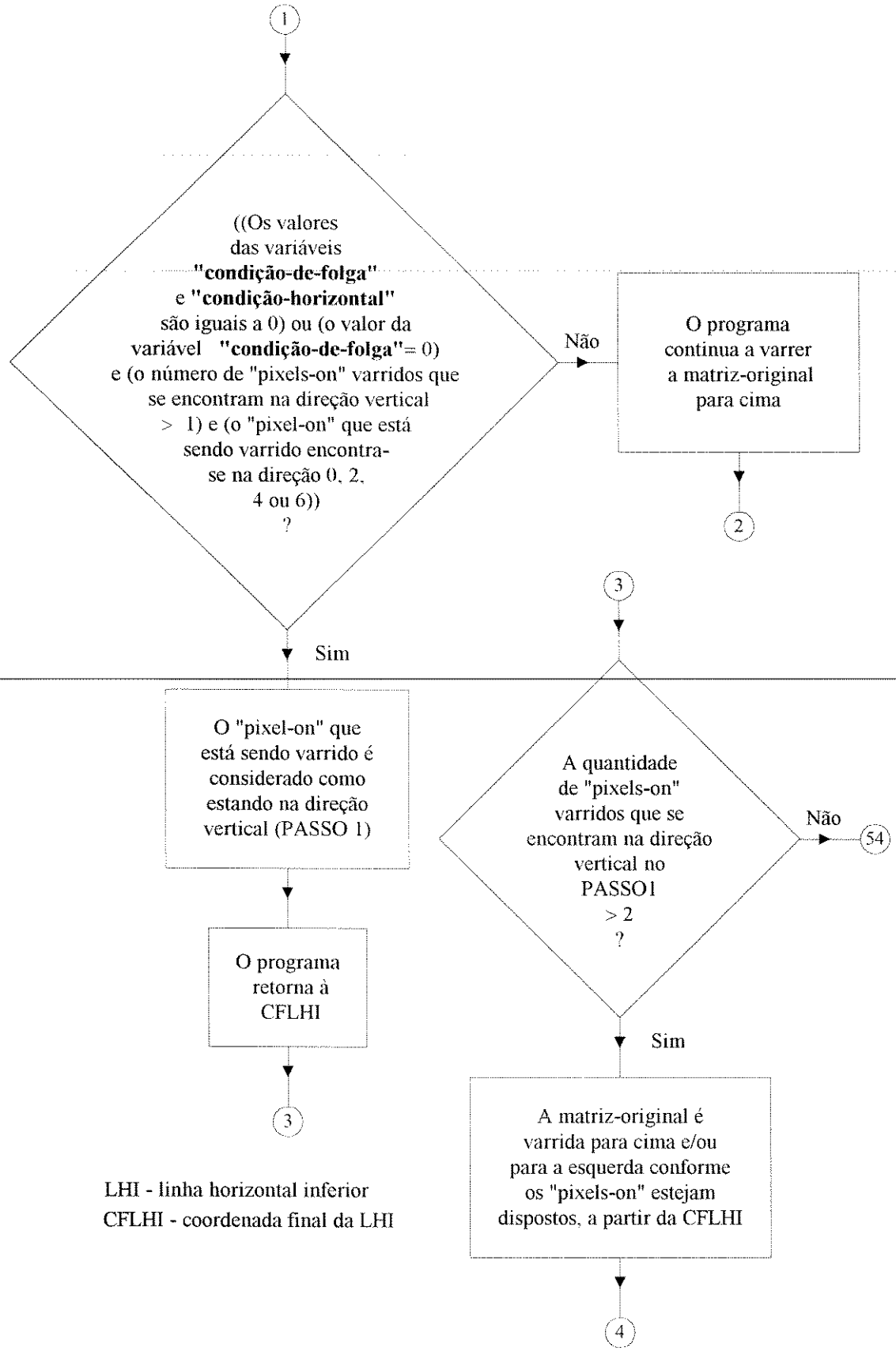


Figura 4.21(b) - Segunda parte do fluxograma a função verif-2-3-5-7-9-8-0.

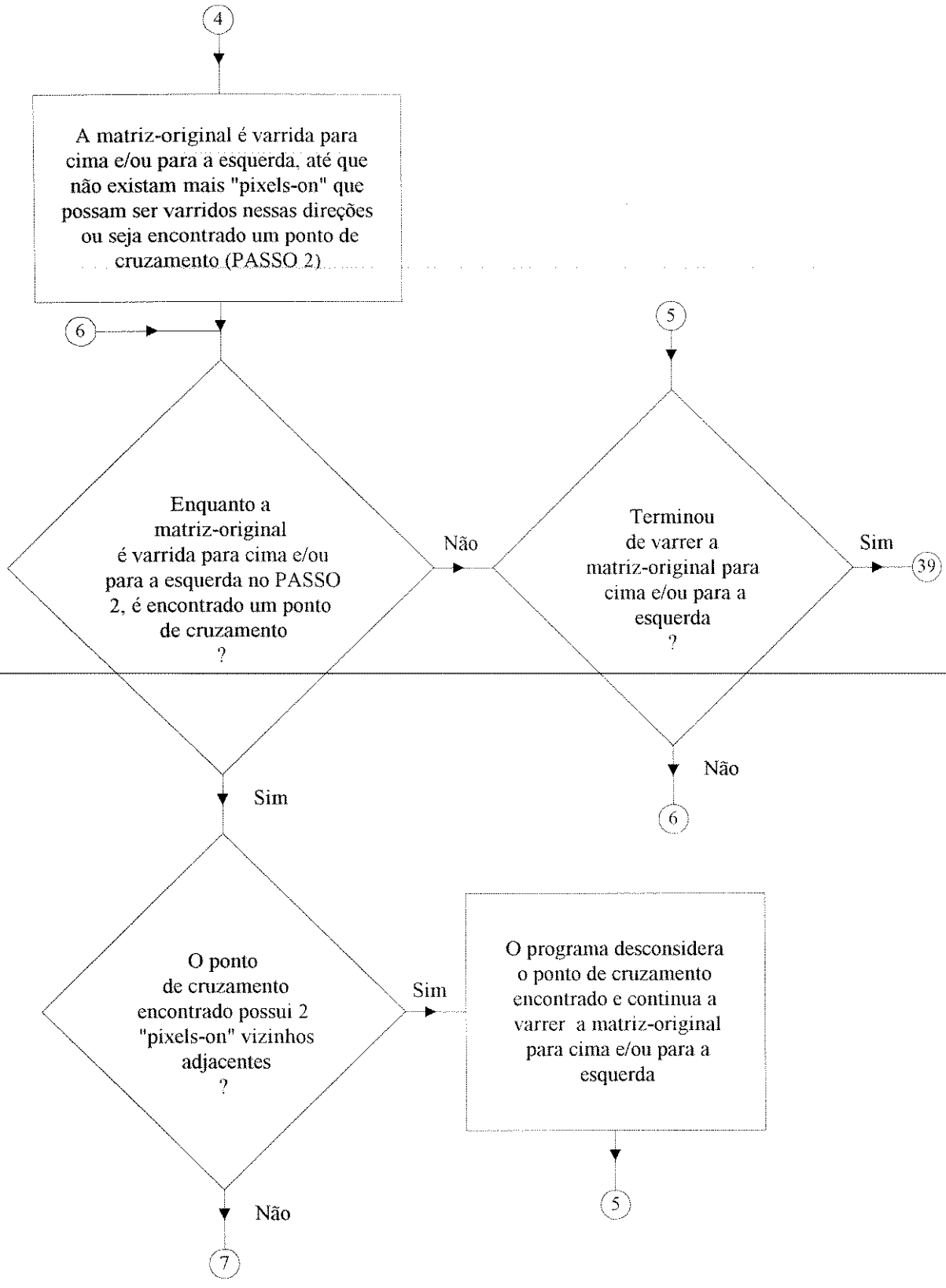


Figura 4.21(c) - Terceira parte do fluxograma da função verif-2-3-5-7-9-8-0.

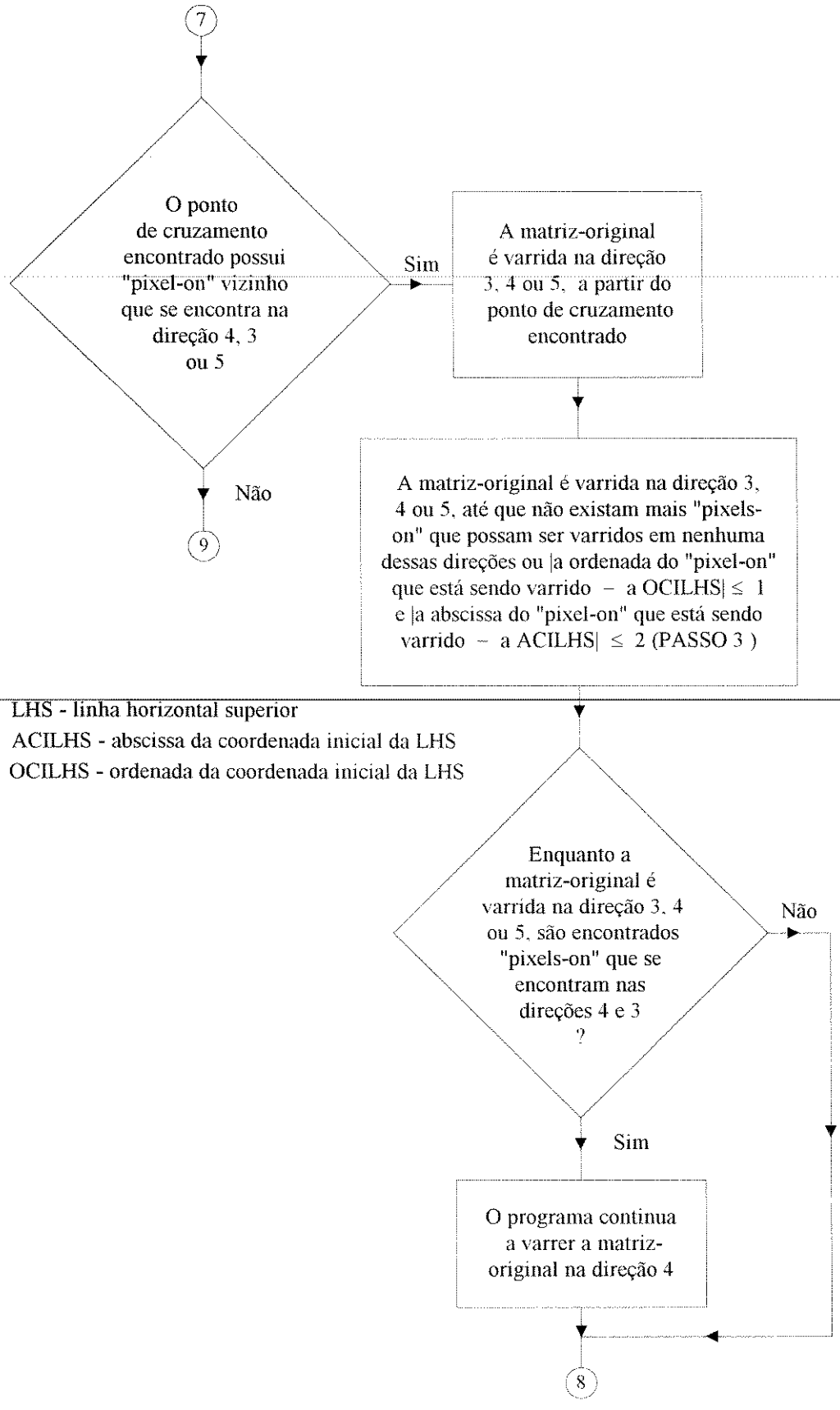


Figura 4.21(d) - Quarta parte do fluxograma da função verif-2-3-5-7-9-8-0.

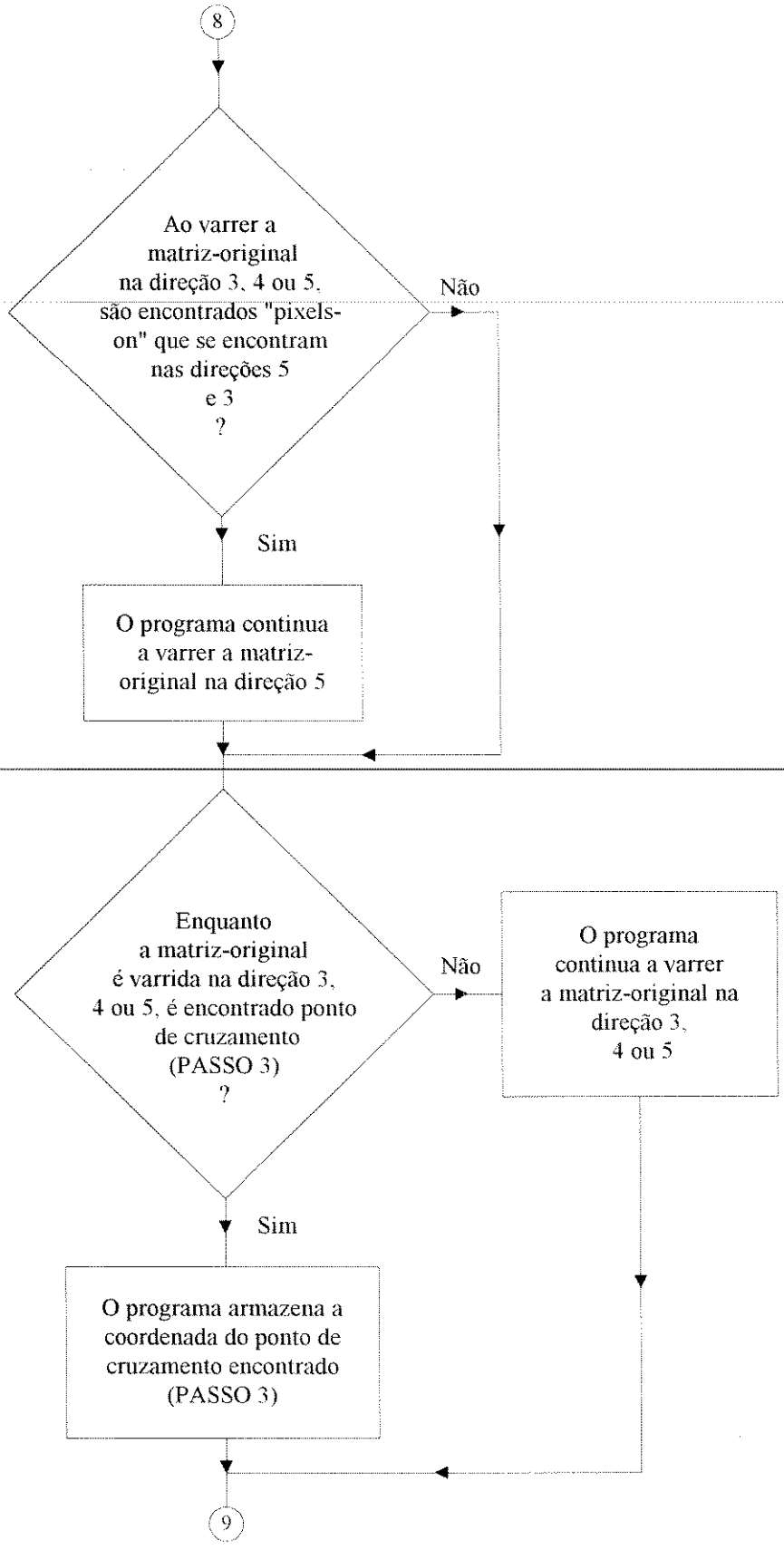


Figura 4.21(e) - Quinta parte do fluxograma da função verif-2-3-5-7-9-8-0.

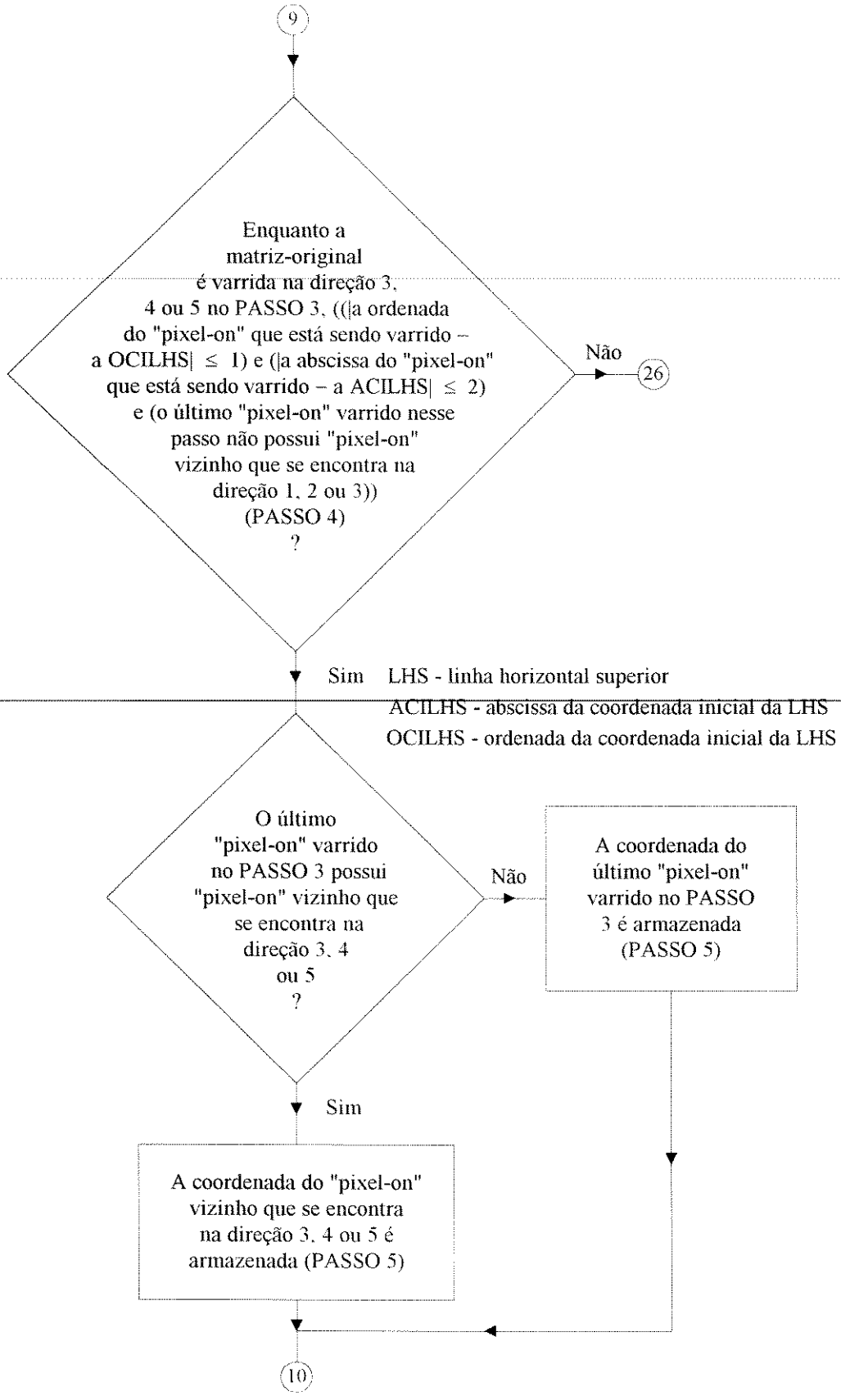


Figura 4.21(f) - Sexta parte do fluxograma da função verif-2-3-5-7-9-8-0.

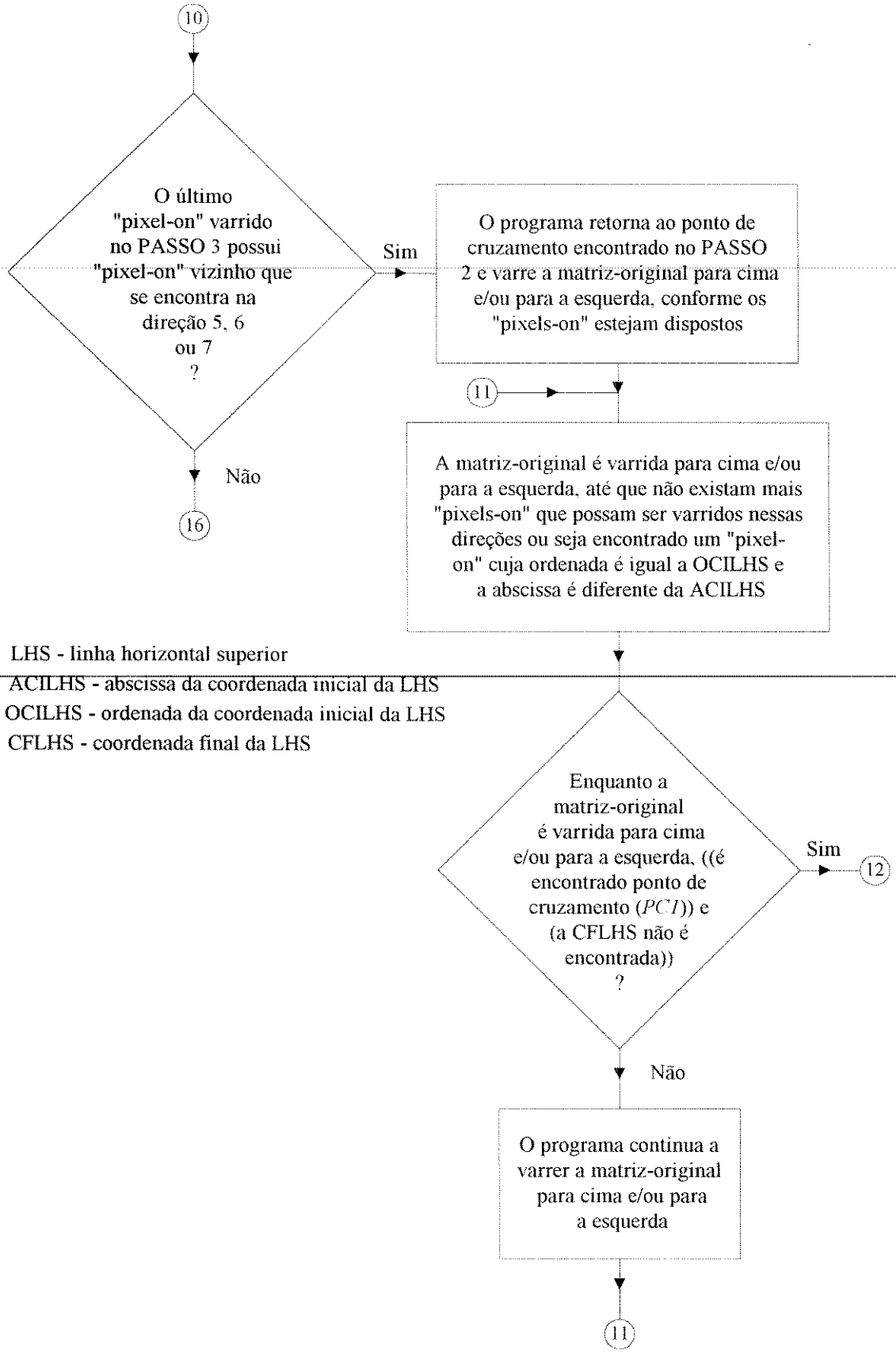


Figura 4.21(g) - Sétima parte do fluxograma da função verif-2-3-5-7-9-8-0.

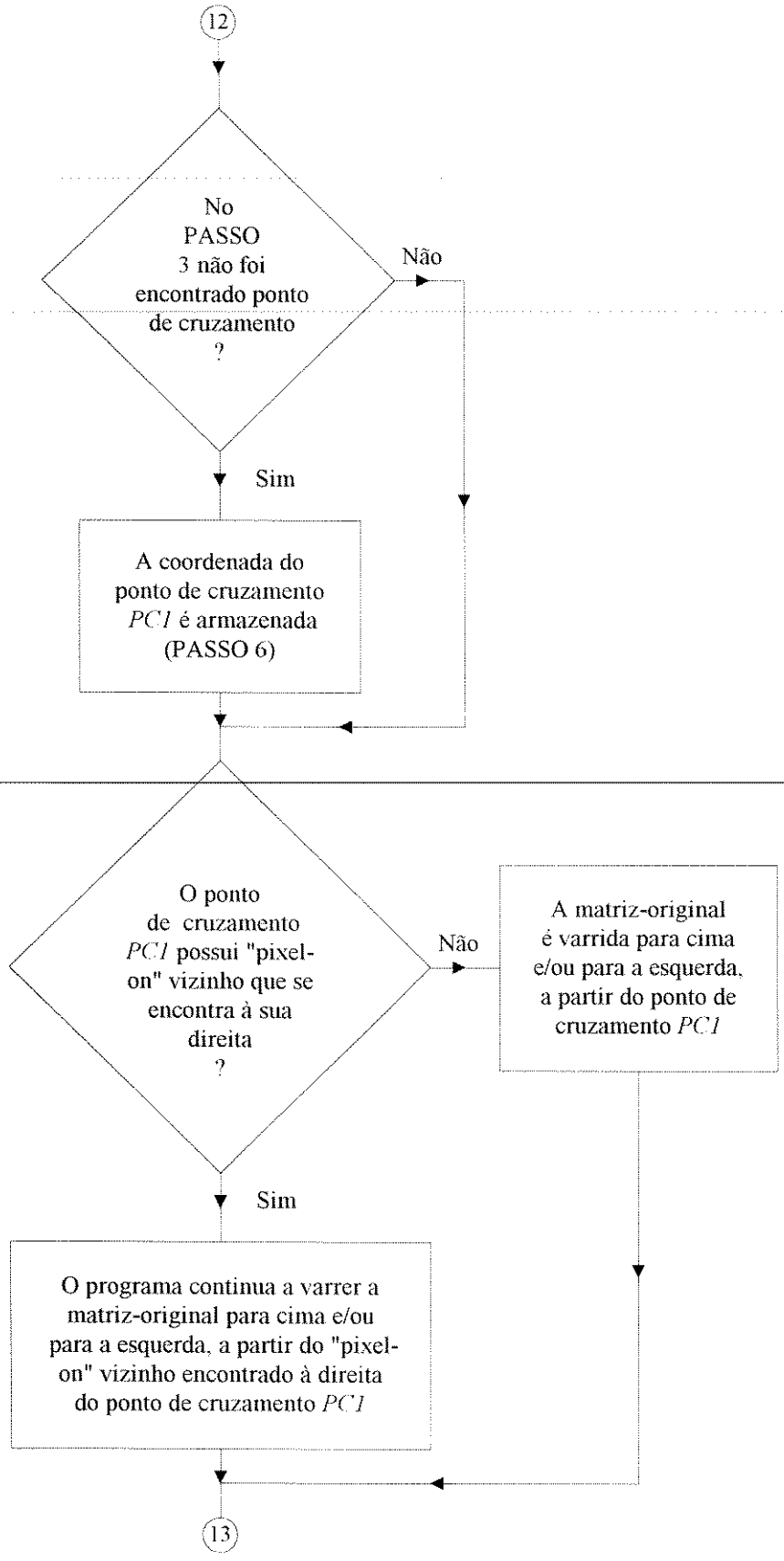


Figura 4.21(h) - Oitava parte do fluxograma da função verif-2-3-5-7-9-8-0.

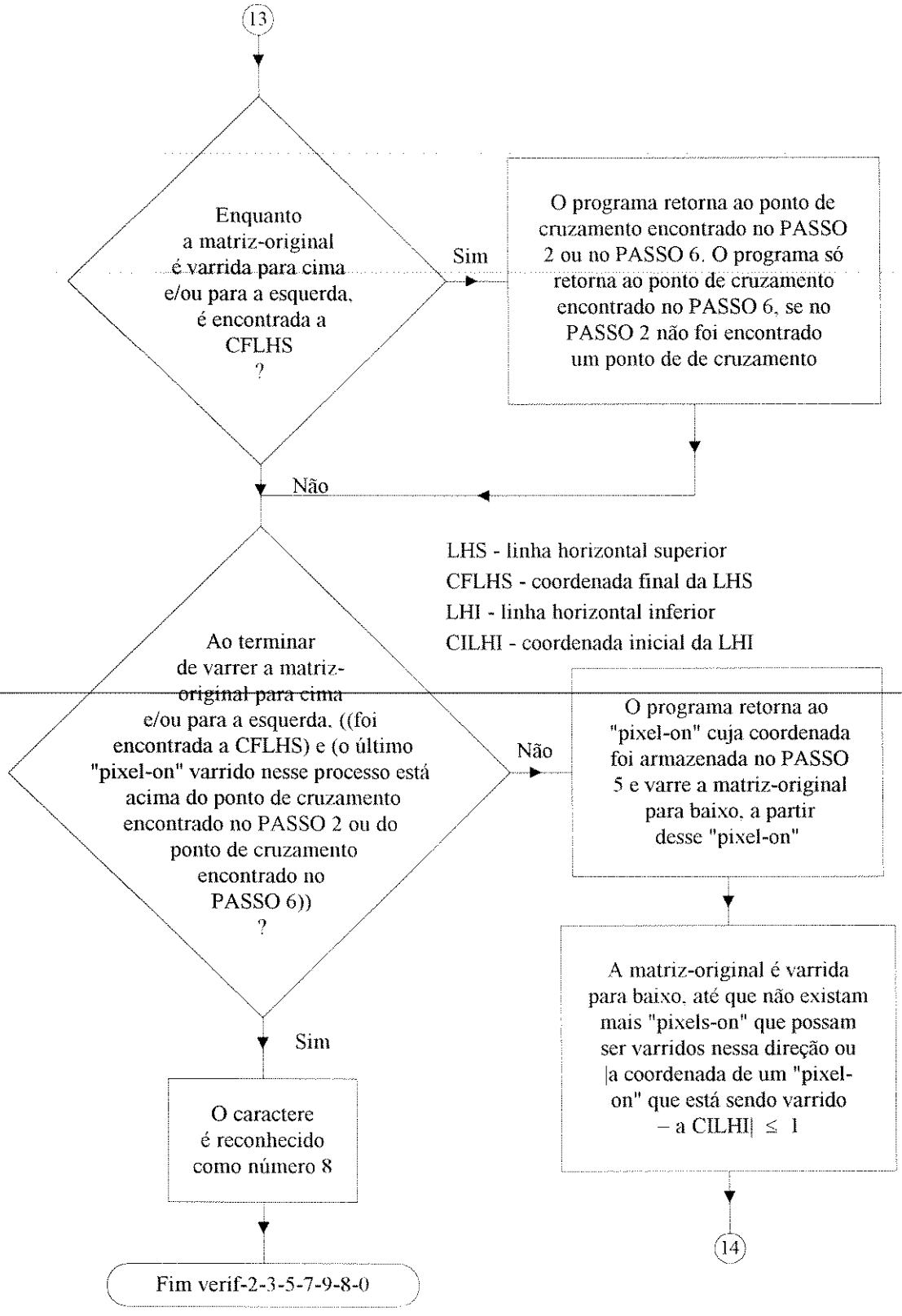
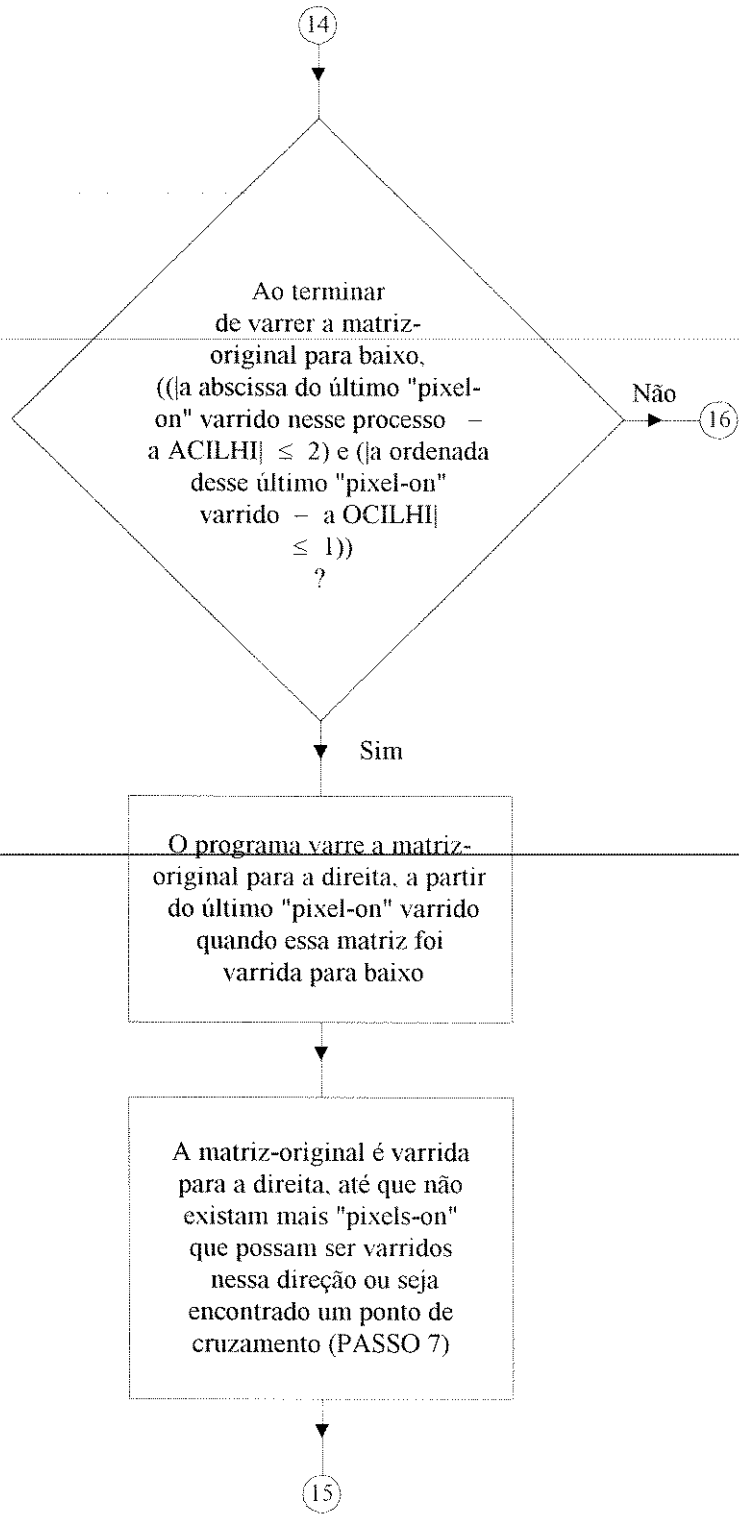


Figura 4.21(i) - Nona parte do fluxograma da função verif-2-3-5-7-9-8-0.



LHI - linha horizontal inferior
ACILHI - abscissa da coordenada inicial da LHI
OCILHI - ordenada da coordenada inicial da LHI

Figura 4.21(j) - Décima parte do fluxograma da função verif-2-3-5-7-9-8-0.

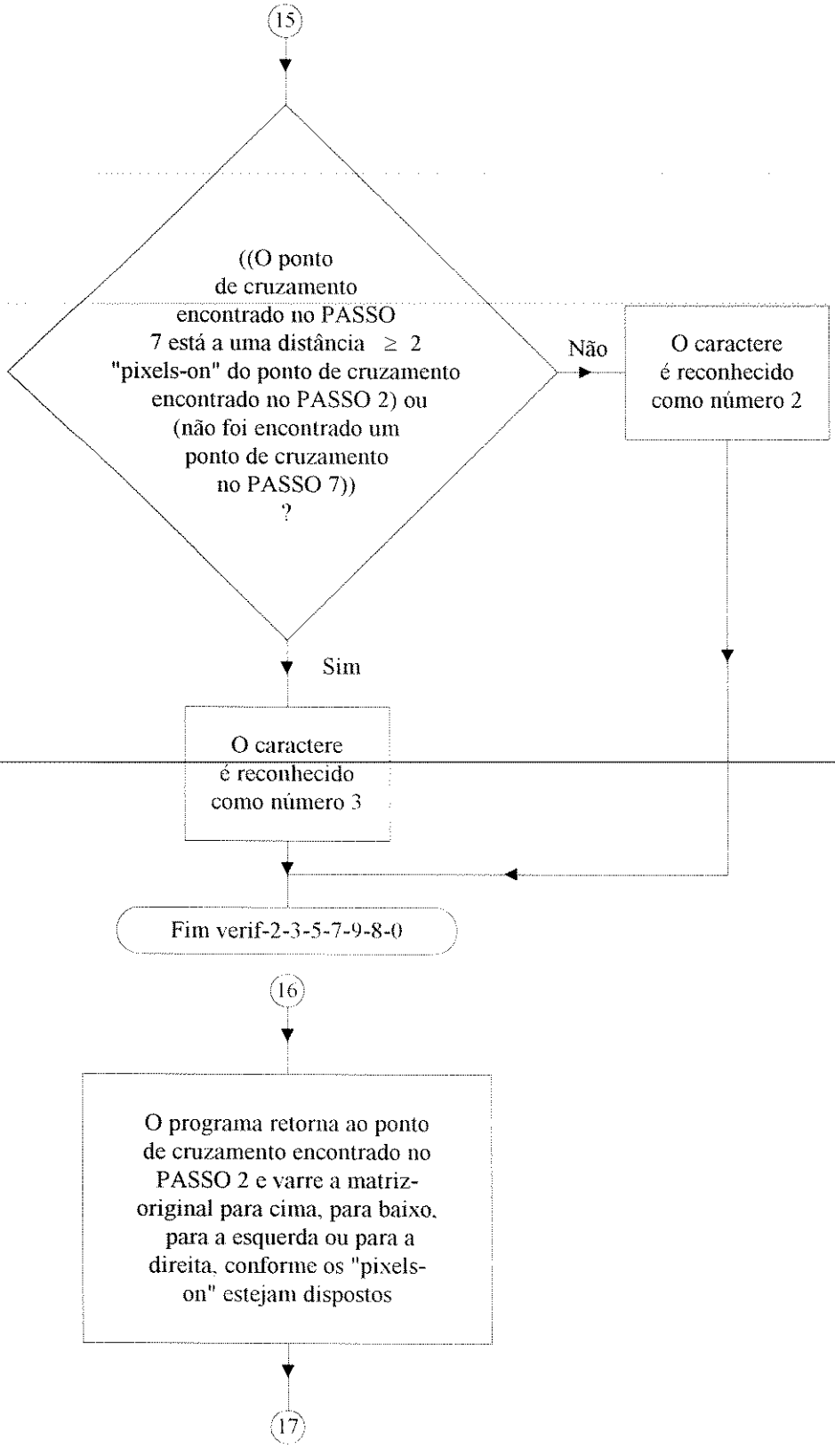


Figura 4.21(k) - Décima primeira parte do fluxograma da função verif-2-3-5-7-9-8-0.

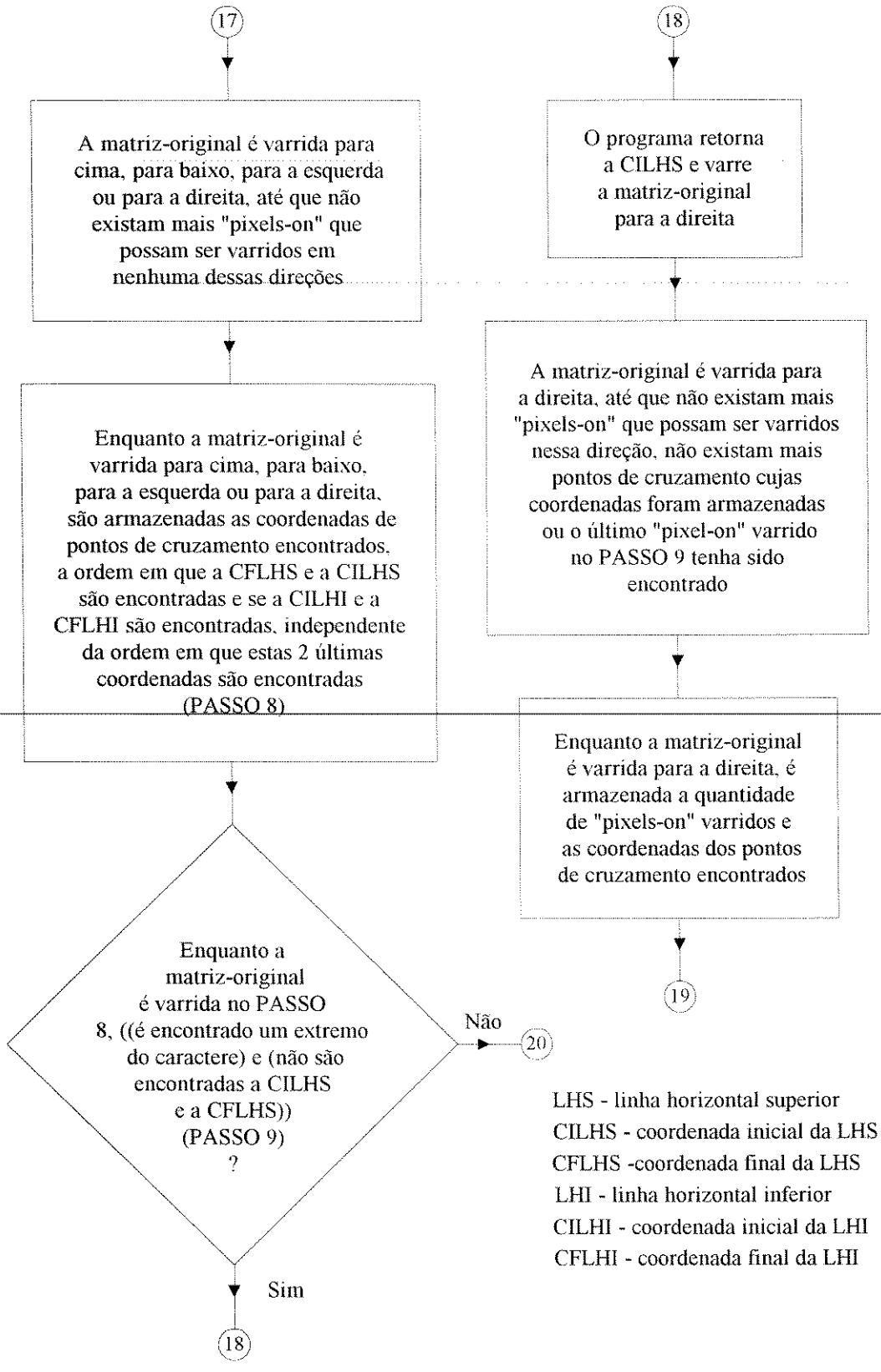


Figura 4.21(l) - Décima segunda parte do fluxograma da função verif-2-3-5-7-9-8-0.

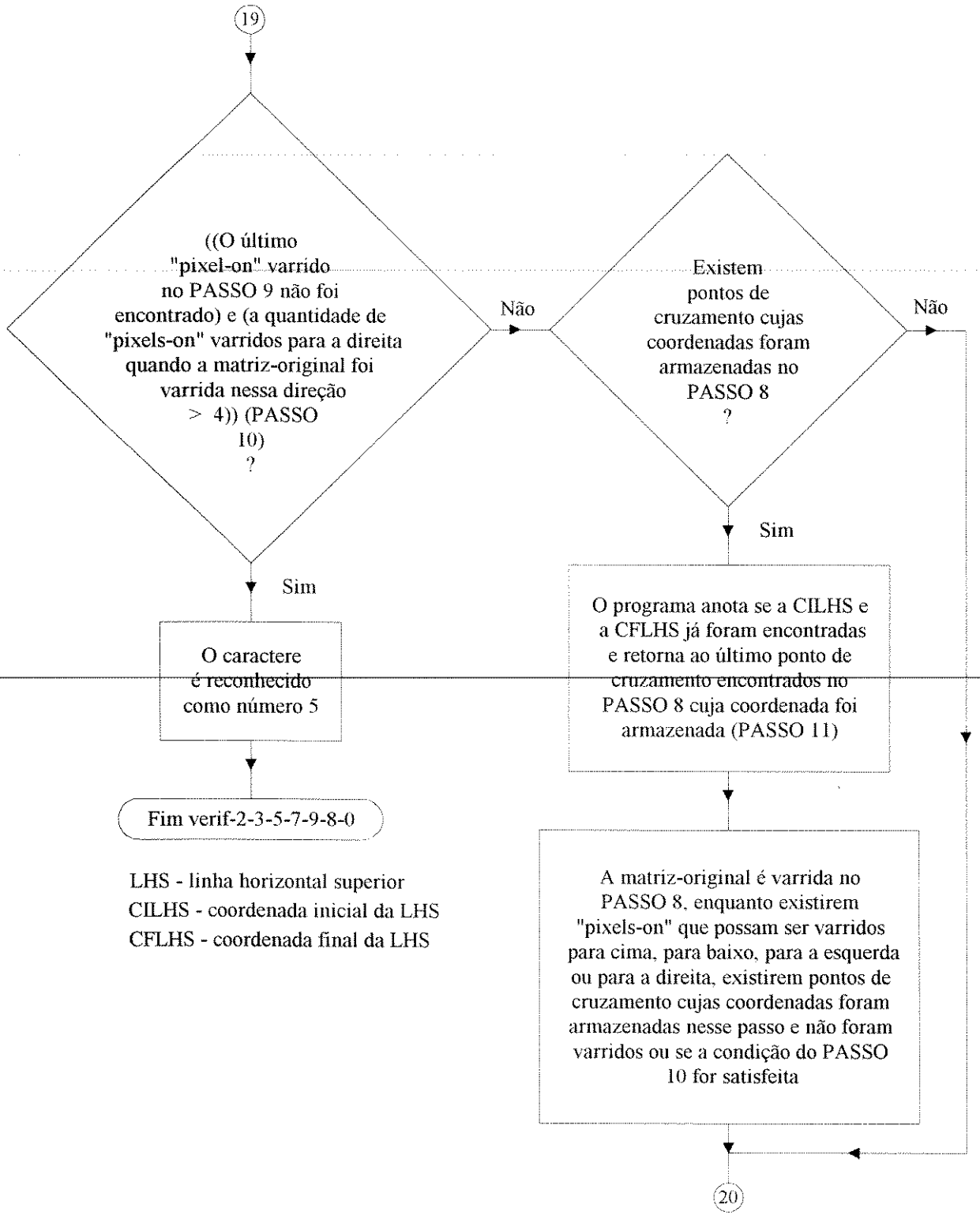


Figura 4.21(m) - Décima terceira parte do fluxograma da função verif-2-3-5-7-9-8-0.

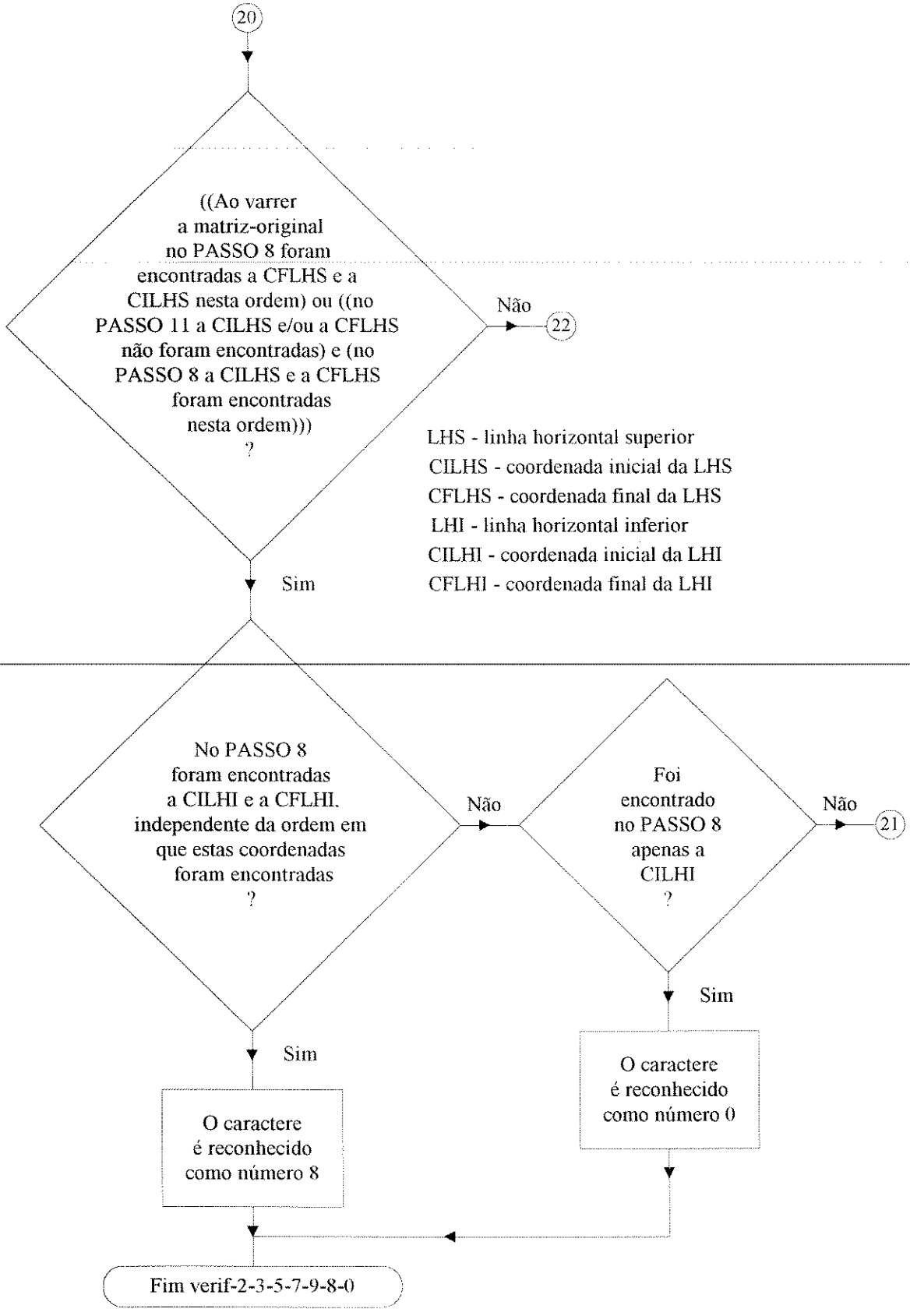


Figura 4.21(n) - Décima quarta parte do fluxograma da função verif-2-3-5-7-9-8-0.

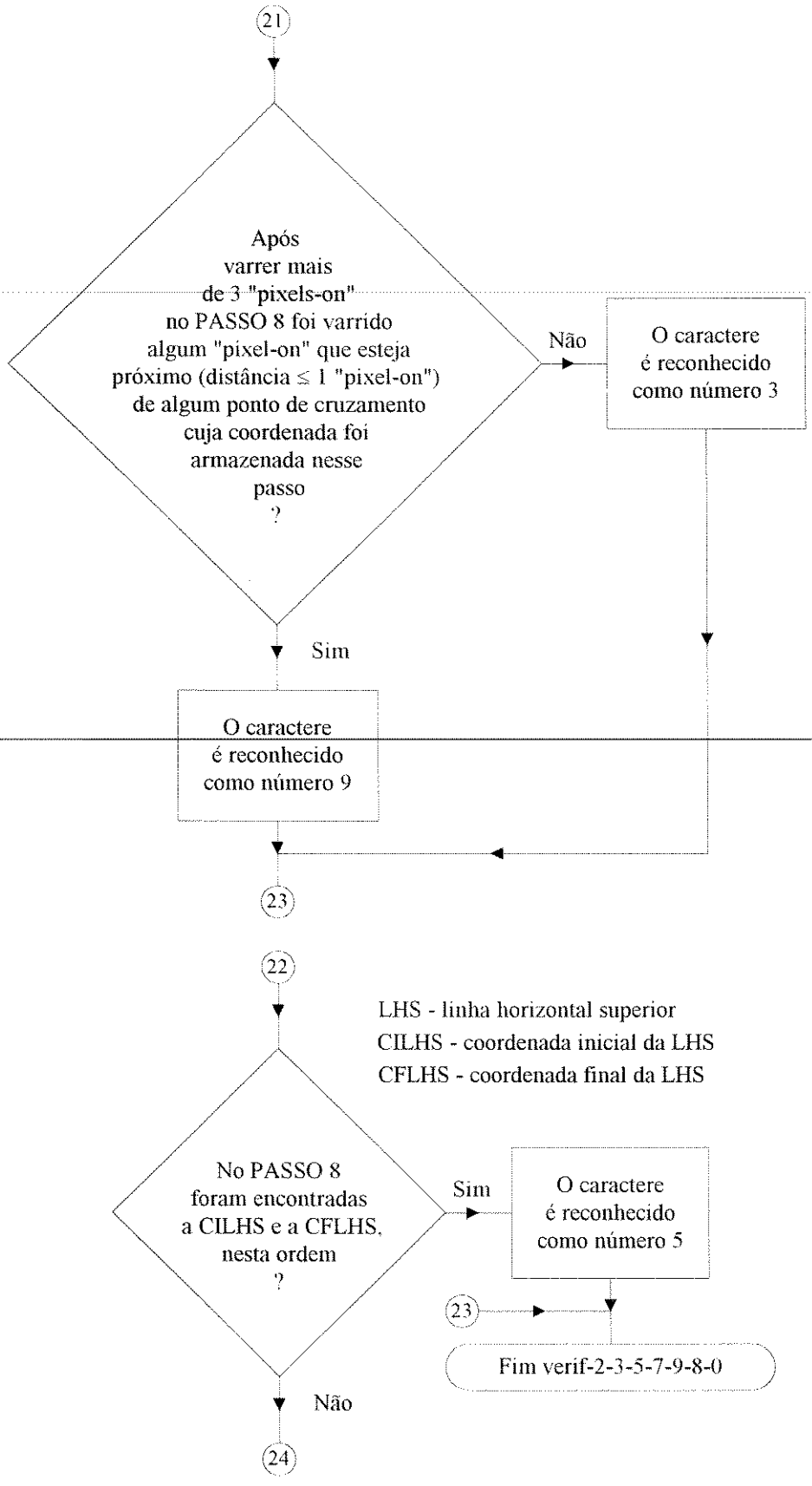


Figura 4.21(o) - Décima quinta parte do fluxograma da função verif-2-3-5-7-9-8-0.

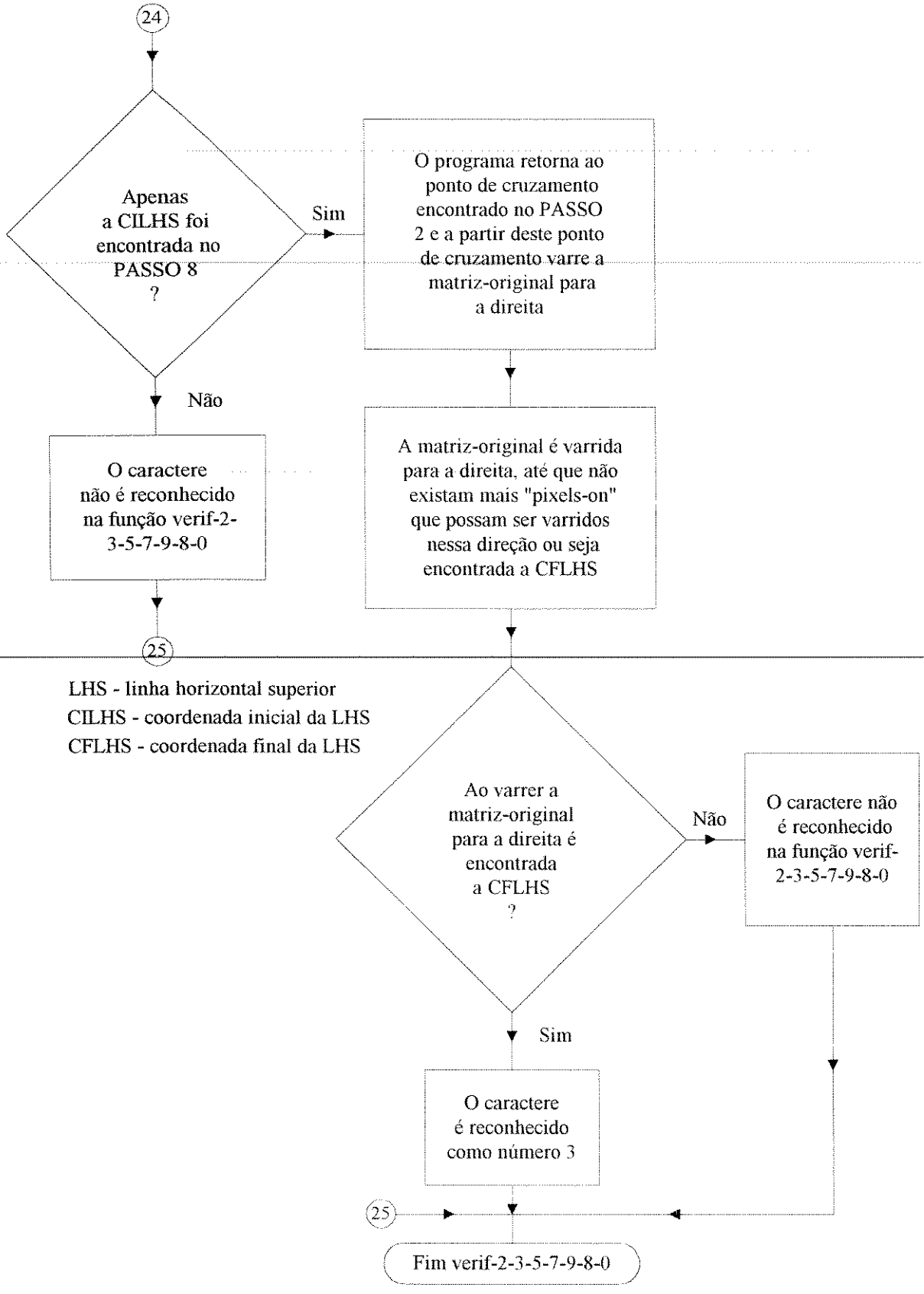


Figura 4.21(p) - Décima sexta parte do fluxograma da função verif-2-3-5-7-9-8-0.

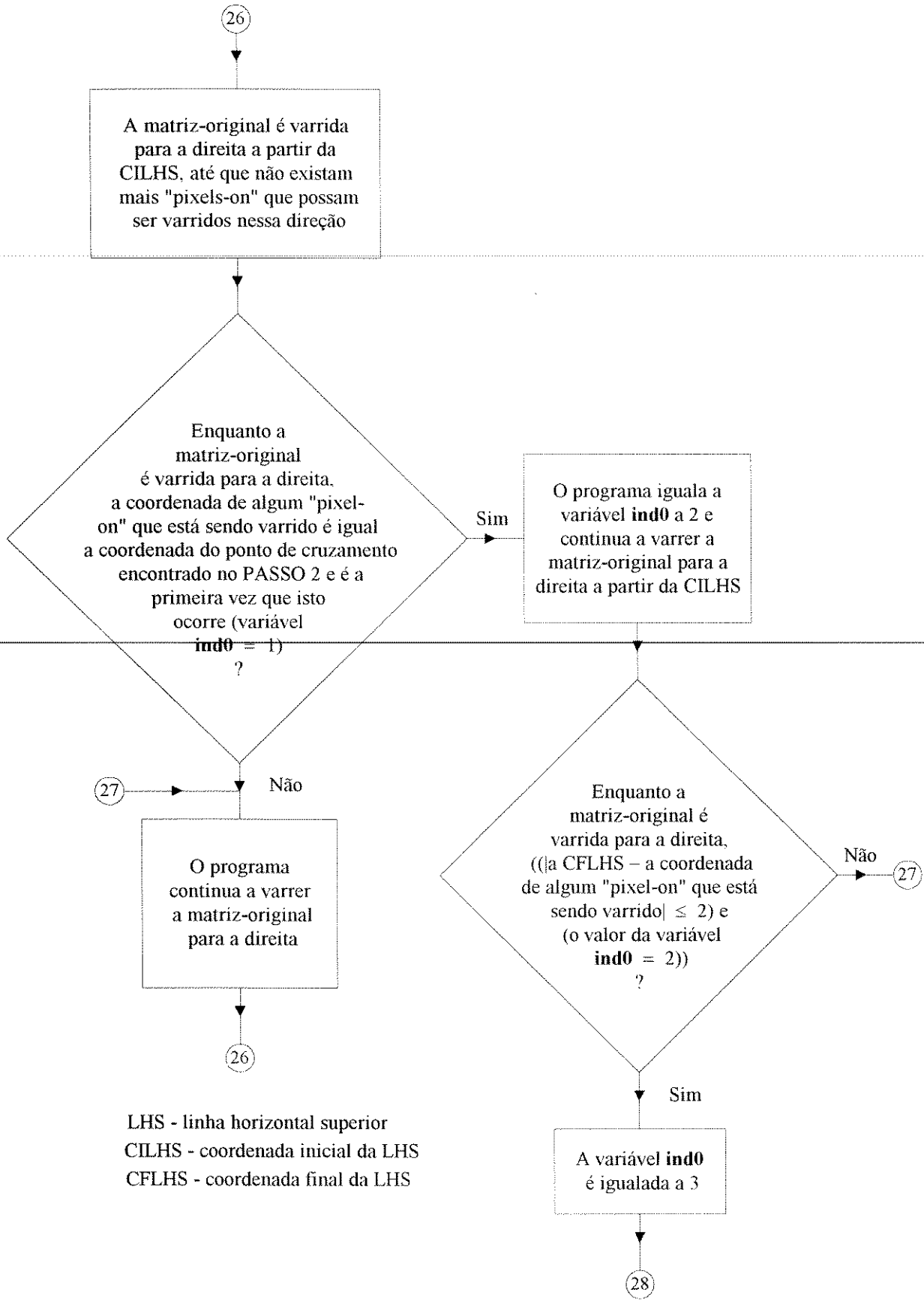
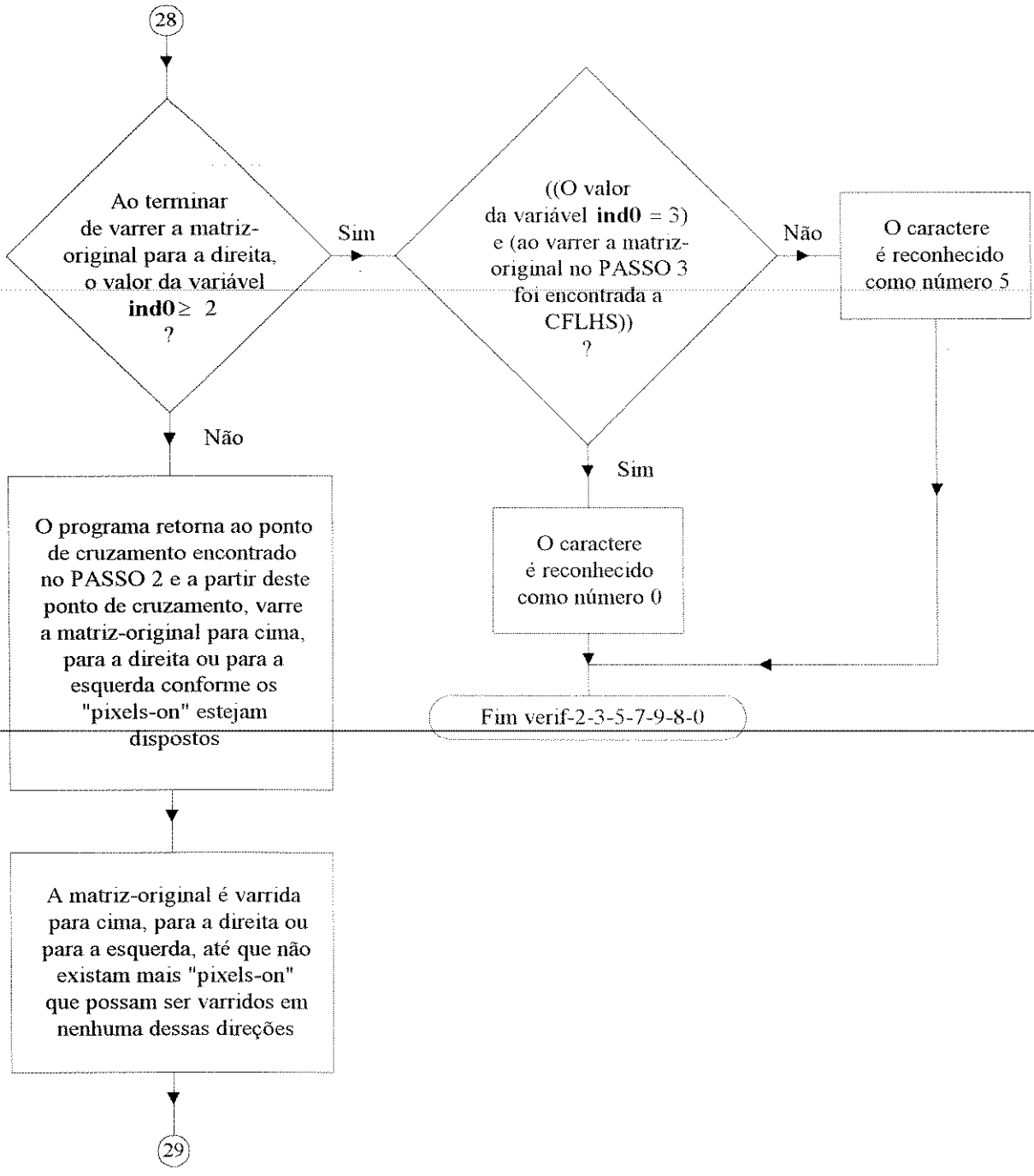


Figura 4.21(q) - Décima sétima parte do fluxograma da função verif-2-3-5-7-9-8-0.



LHS - linha horizontal superior
CFLHS - coordenada final da LHS

Figura 4.21(r) - Décima oitava parte do fluxograma da função verif-2-3-5-7-9-8-0.

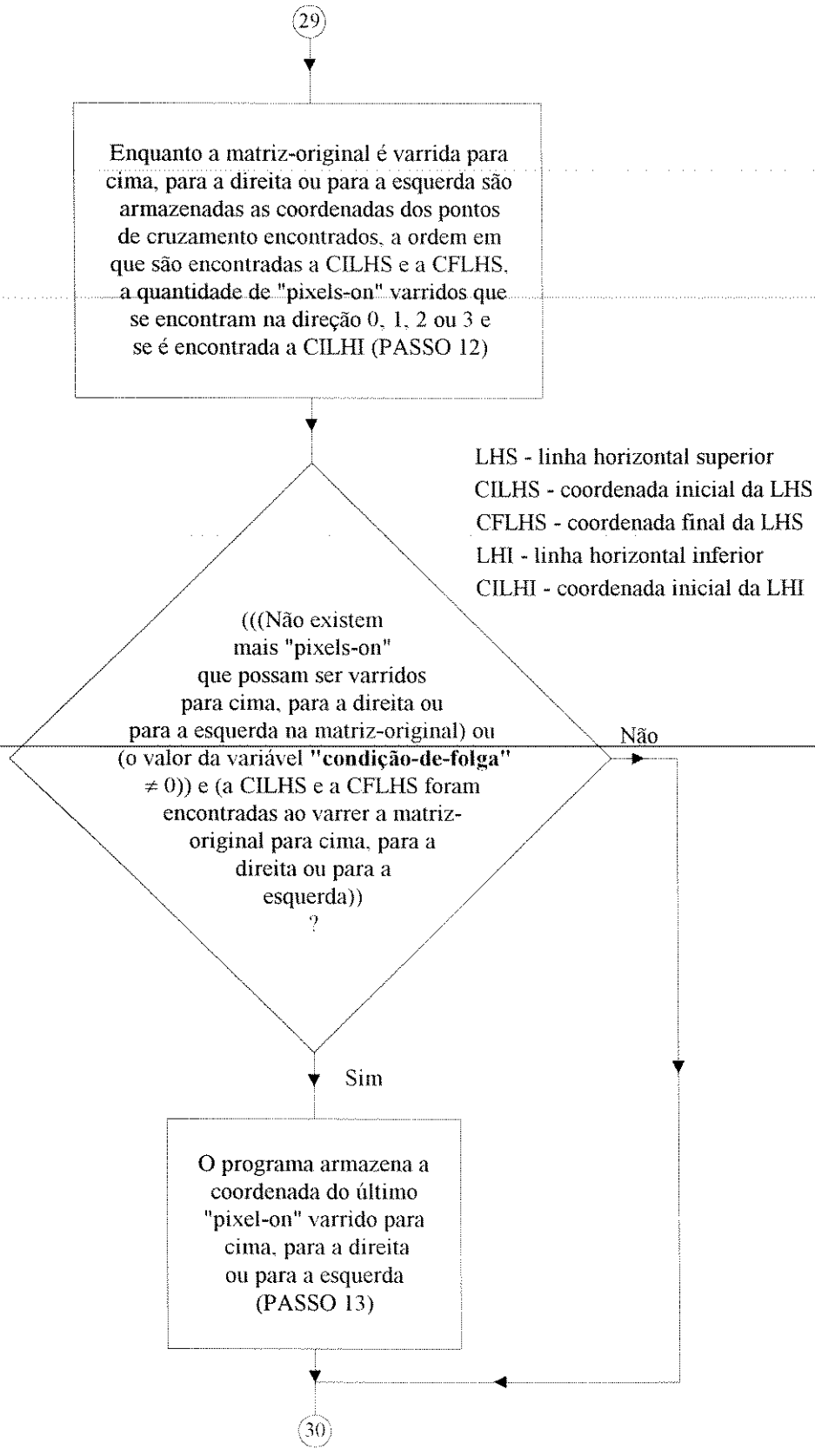


Figura 4.21(s) - Décima nona parte do fluxograma da função verif-2-3-5-7-9-8-0.

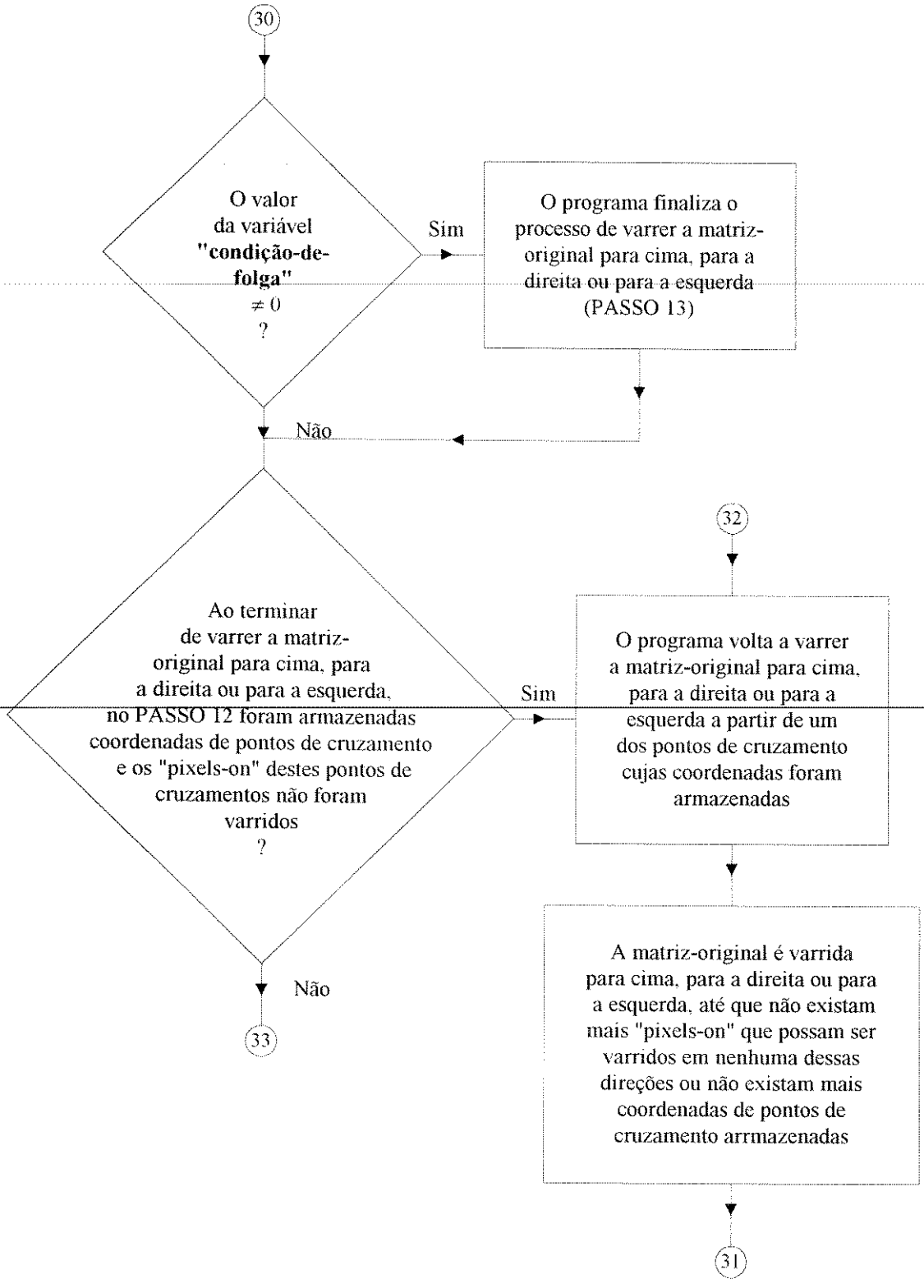


Figura 4.21(t) - Vigésima parte do fluxograma da função verif-2-3-5-7-9-8-0.

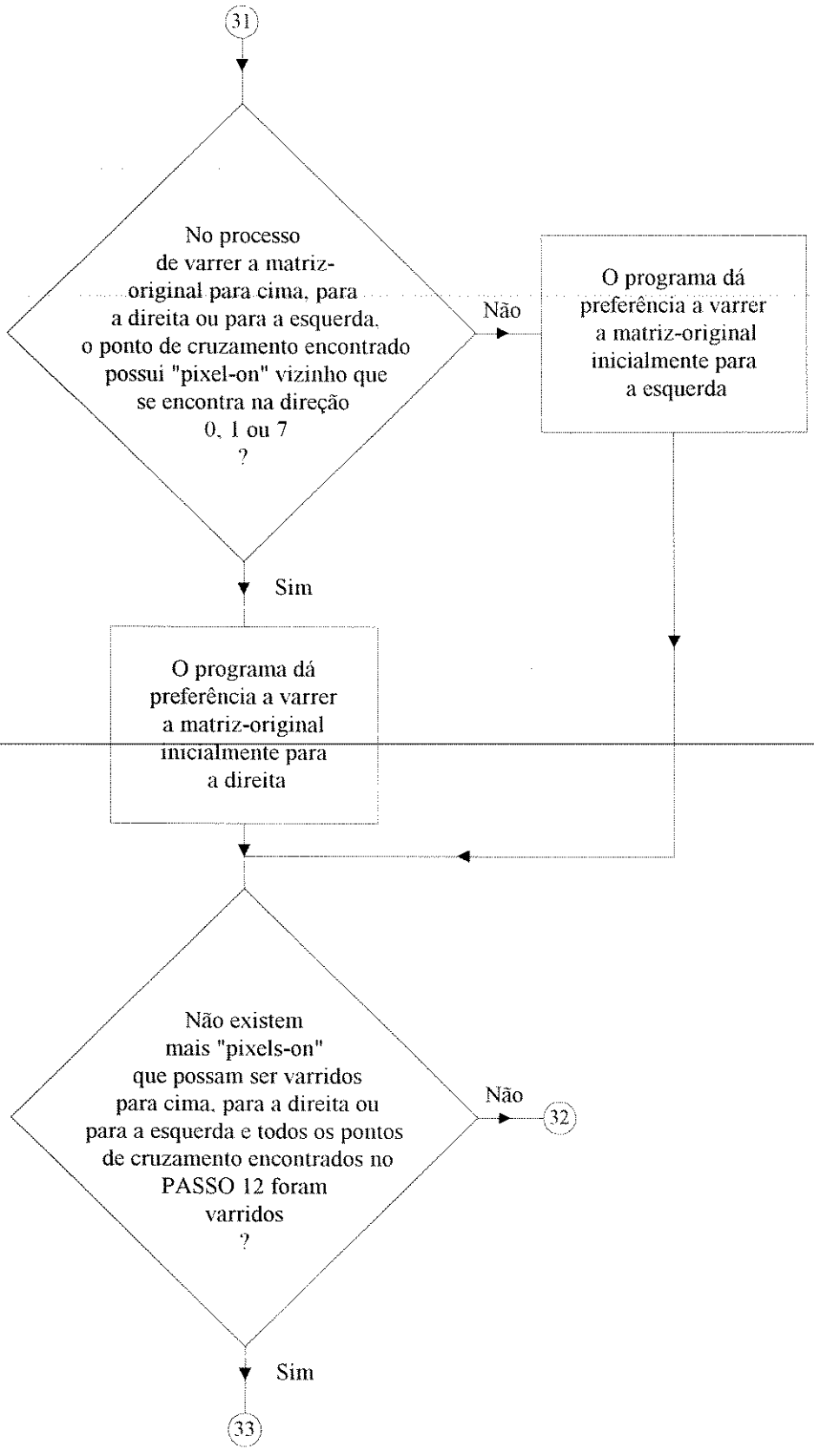


Figura 4.21(u) - Vigésima primeira parte do fluxograma da função verif-2-3-5-7-9-8-0.

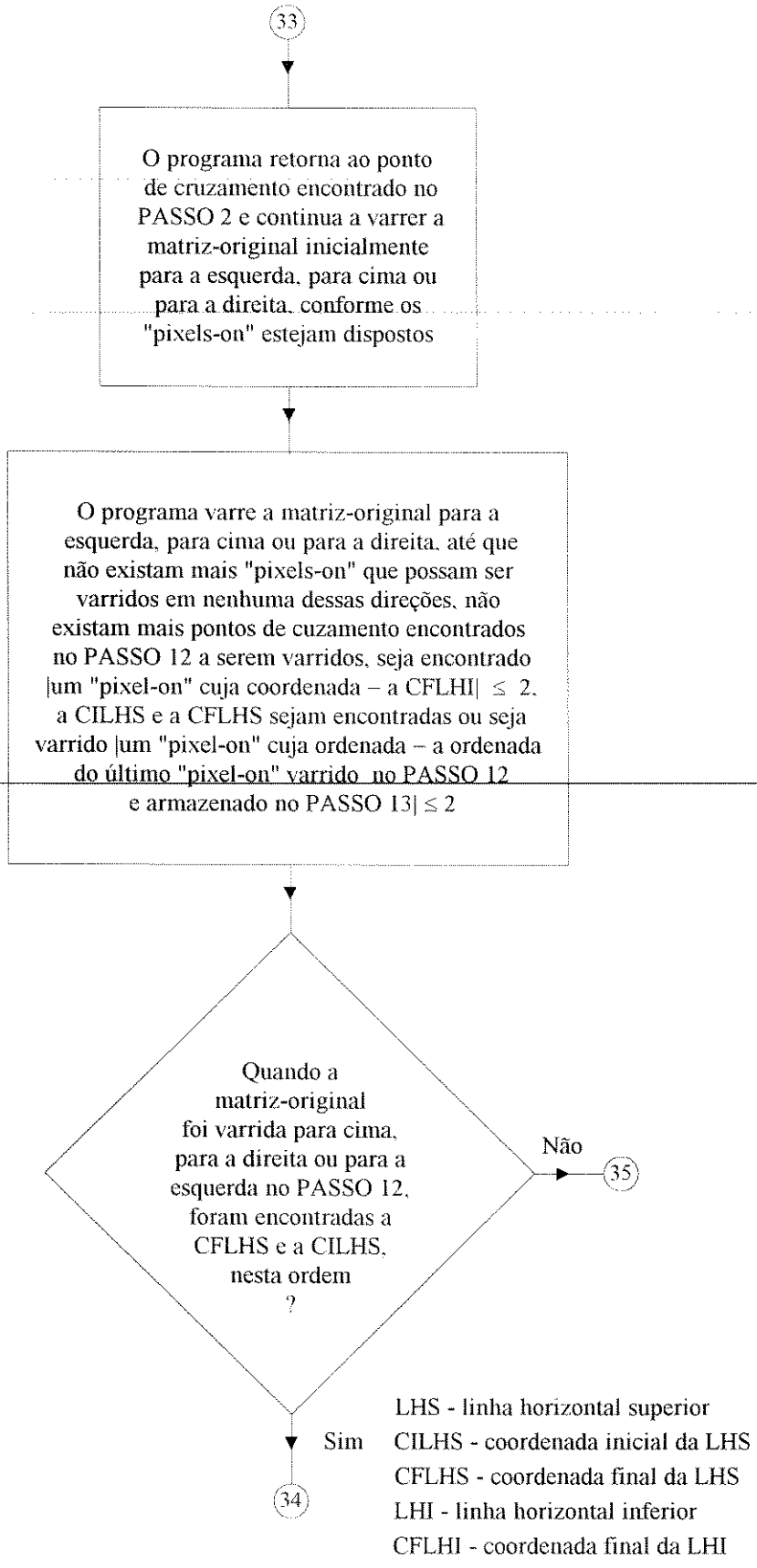


Figura 4.21(v) - Vigésima segunda parte do fluxograma da função verif-2-3-5-7-9-8-0.

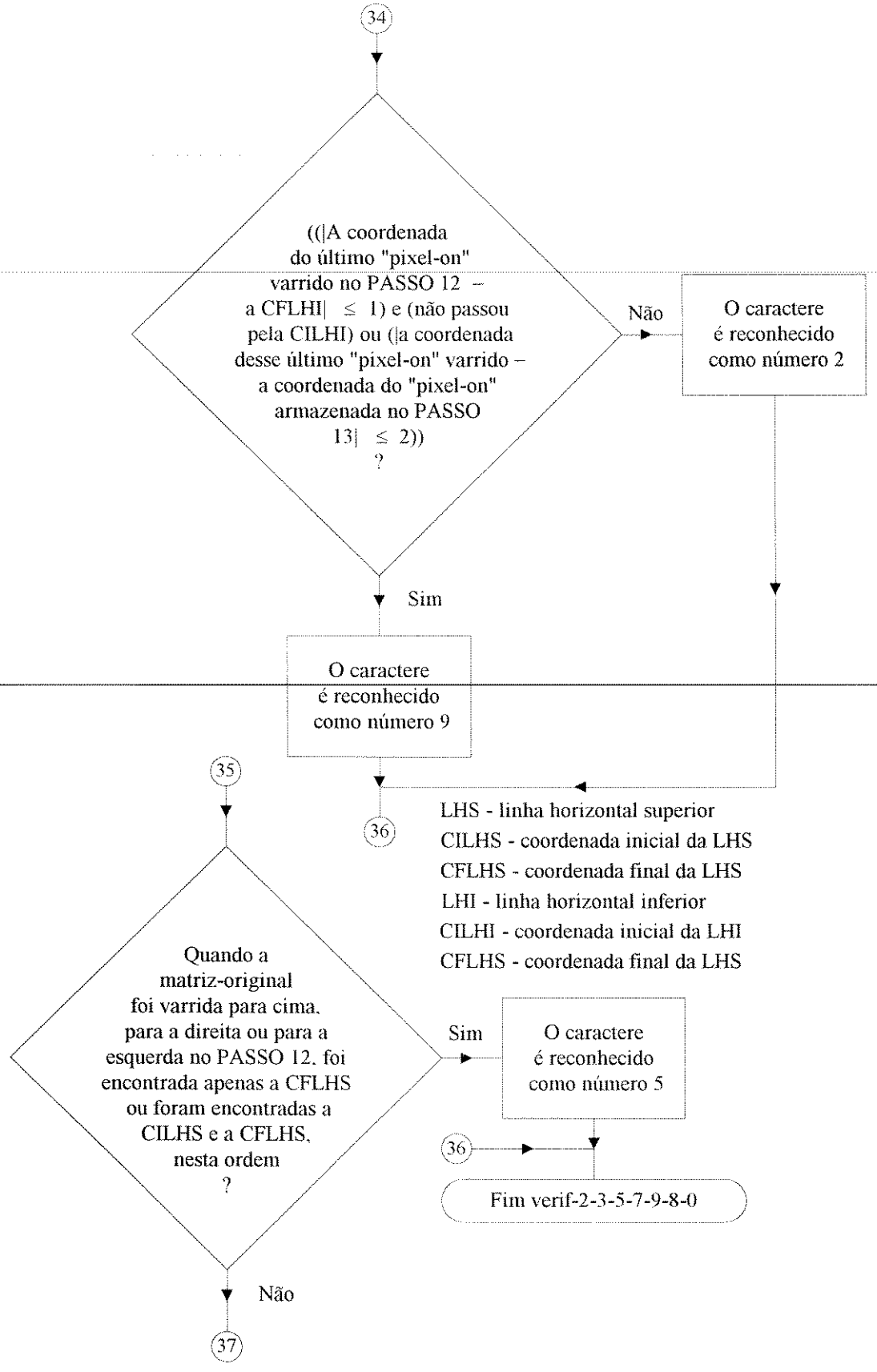


Figura 4.21(w) - Vigésima terceira parte do fluxograma da função verif-2-3-5-7-9-8-0.

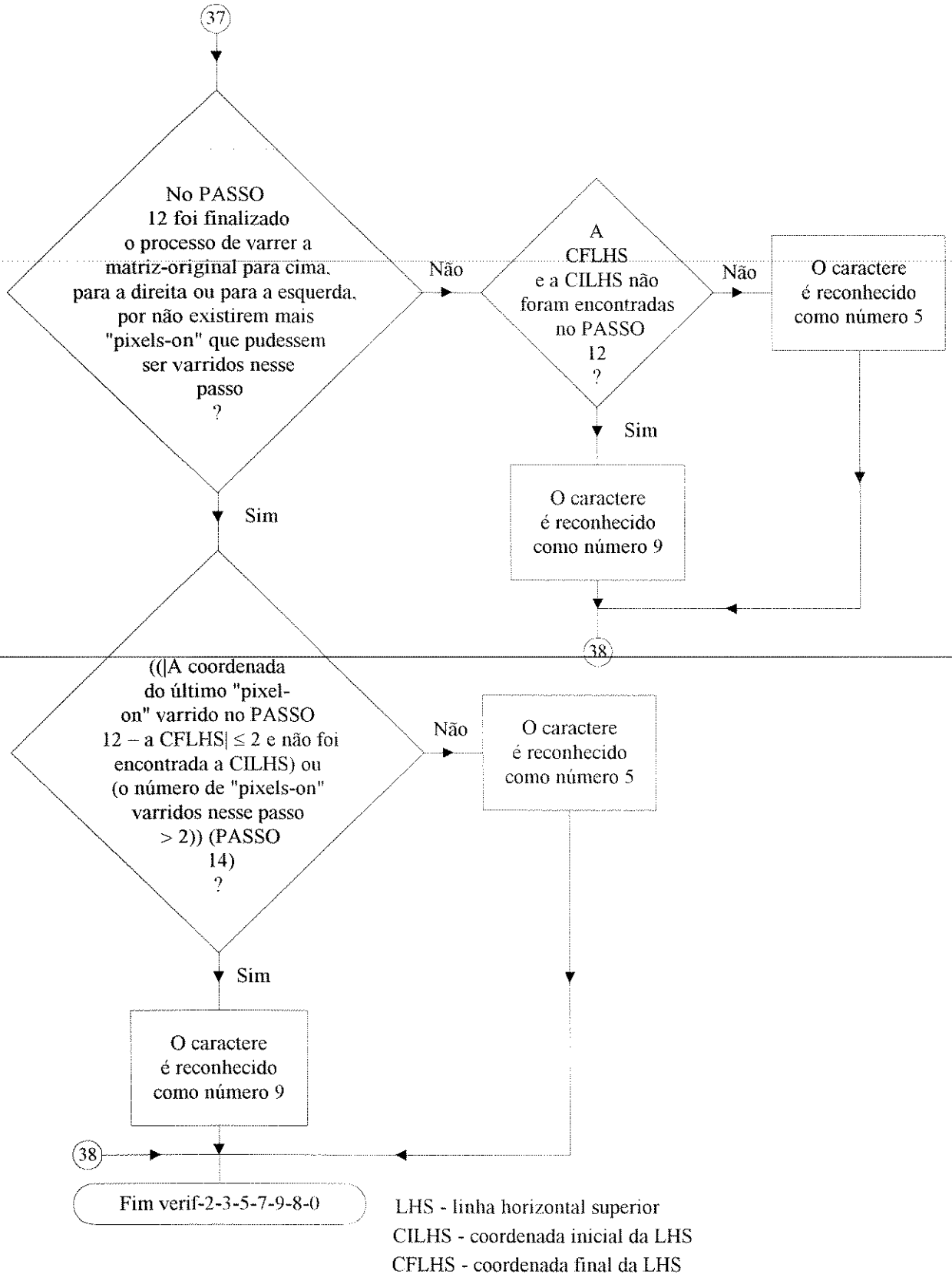


Figura 4.21(x) - Vigésima quarta parte do fluxograma da função verif-2-3-5-7-9-8-0.

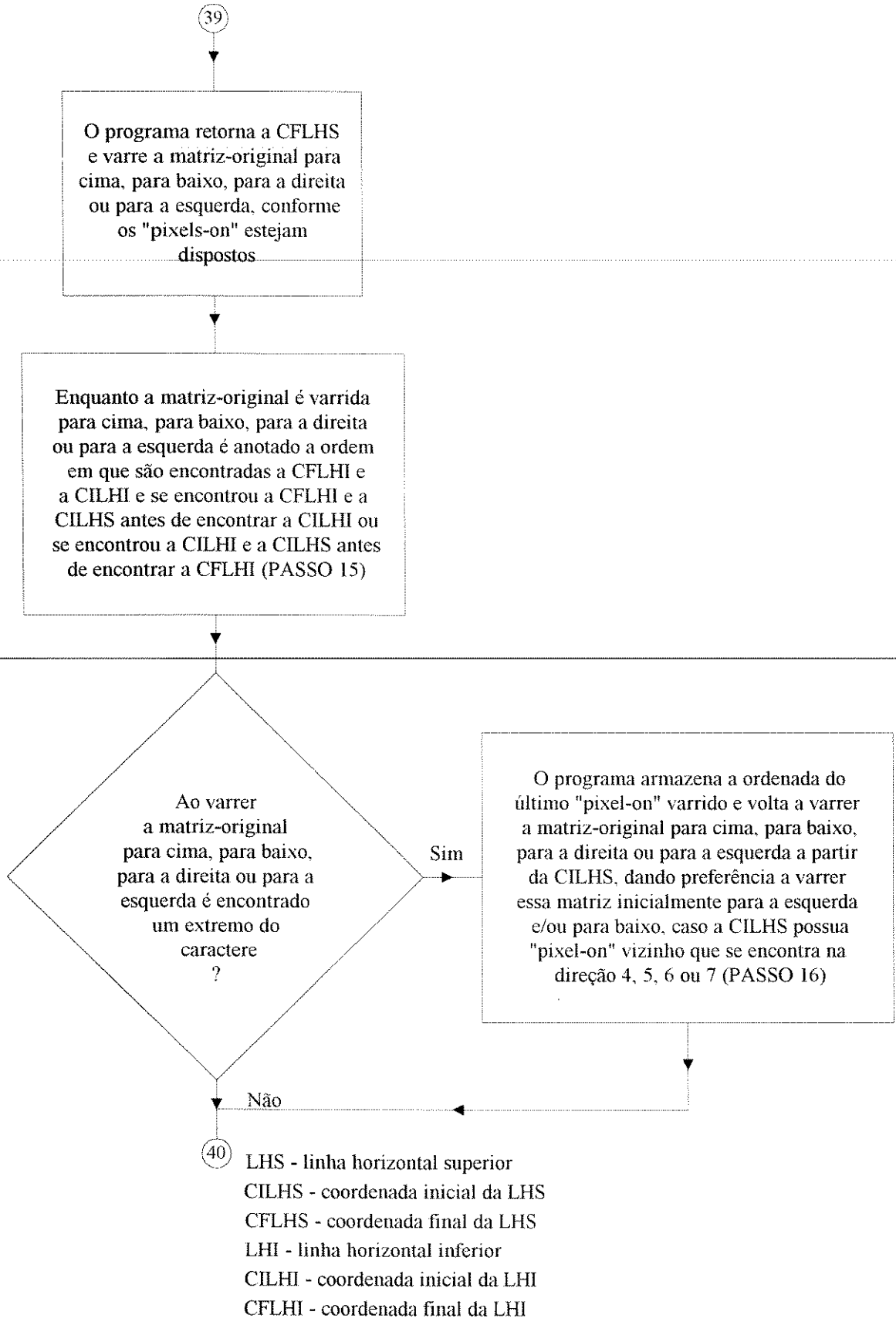


Figura 4.21(y) - Vigésima quinta parte do fluxograma da função verif-2-3-5-7-9-8-0.

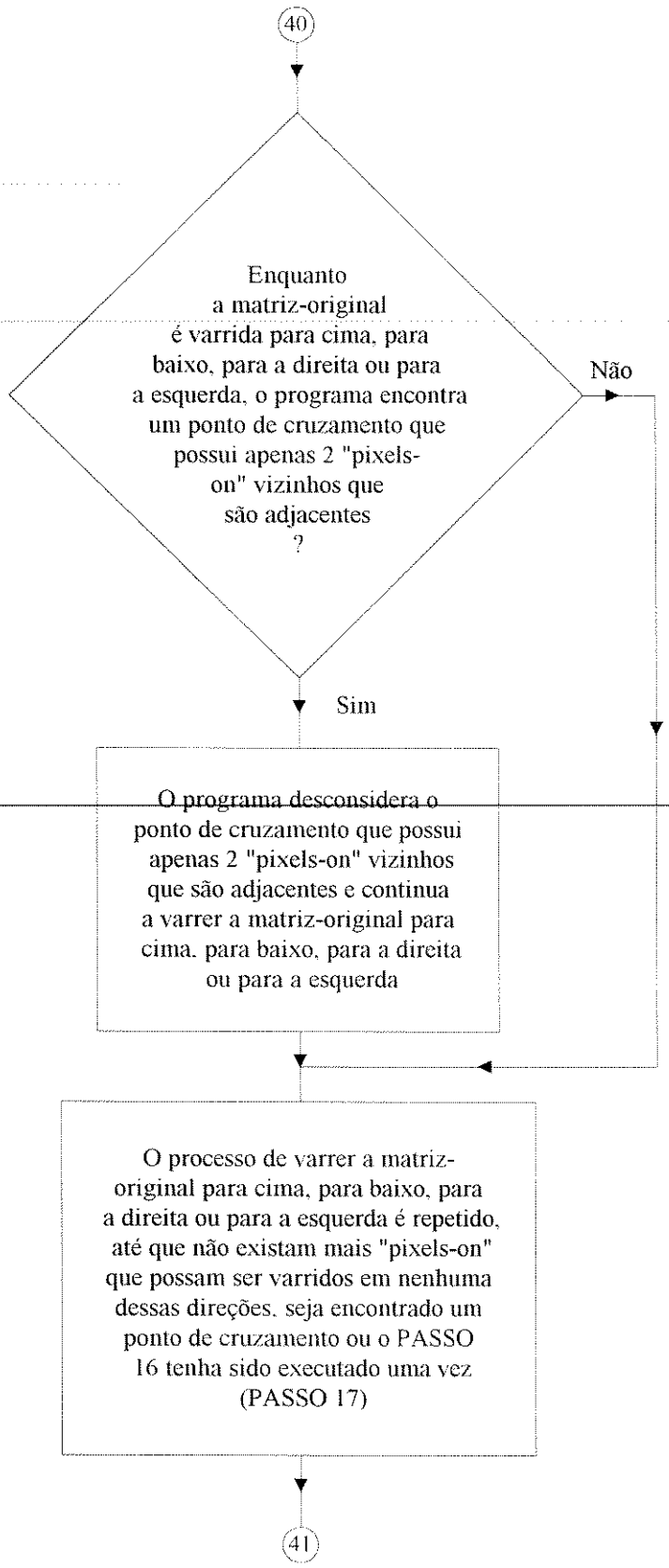


Figura 4.21(z) - Vigésima sexta parte do fluxograma da função verif-2-3-5-7-9-8-0.

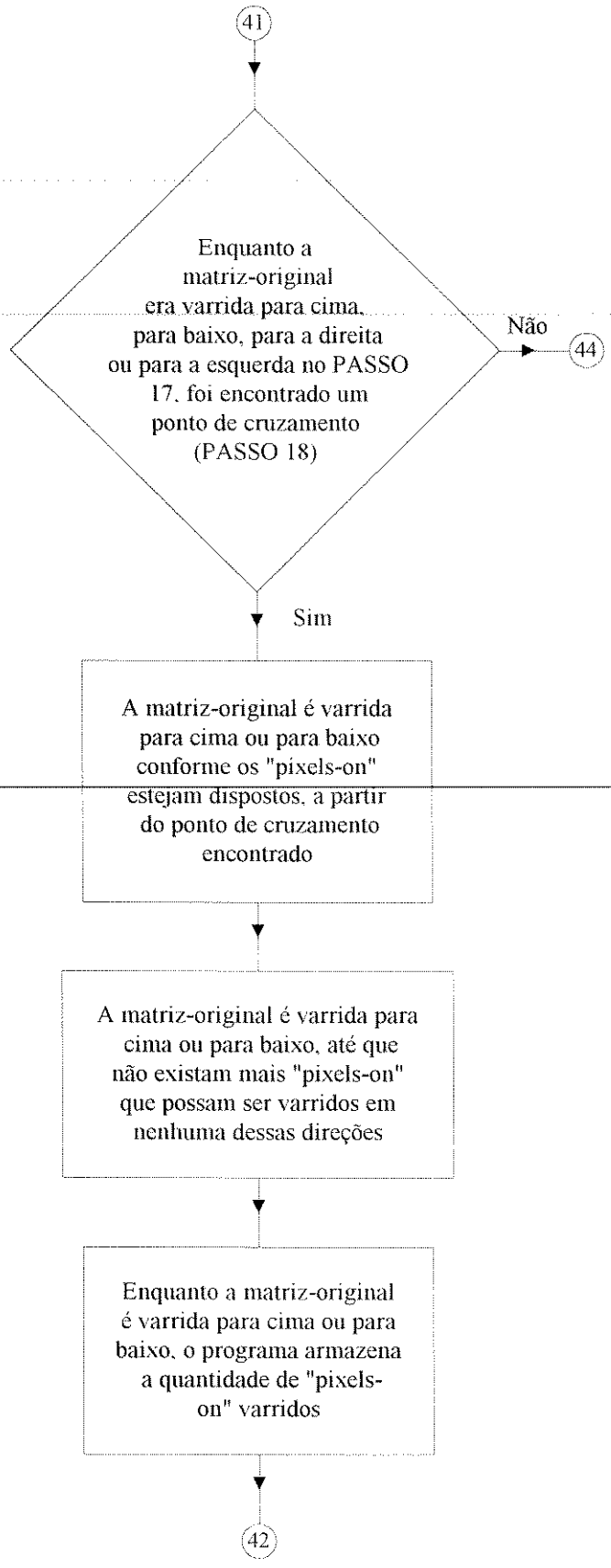


Figura 4.21(a1) - Vigésima sétima parte do fluxograma da função verif-2-3-5-7-9-8-0.

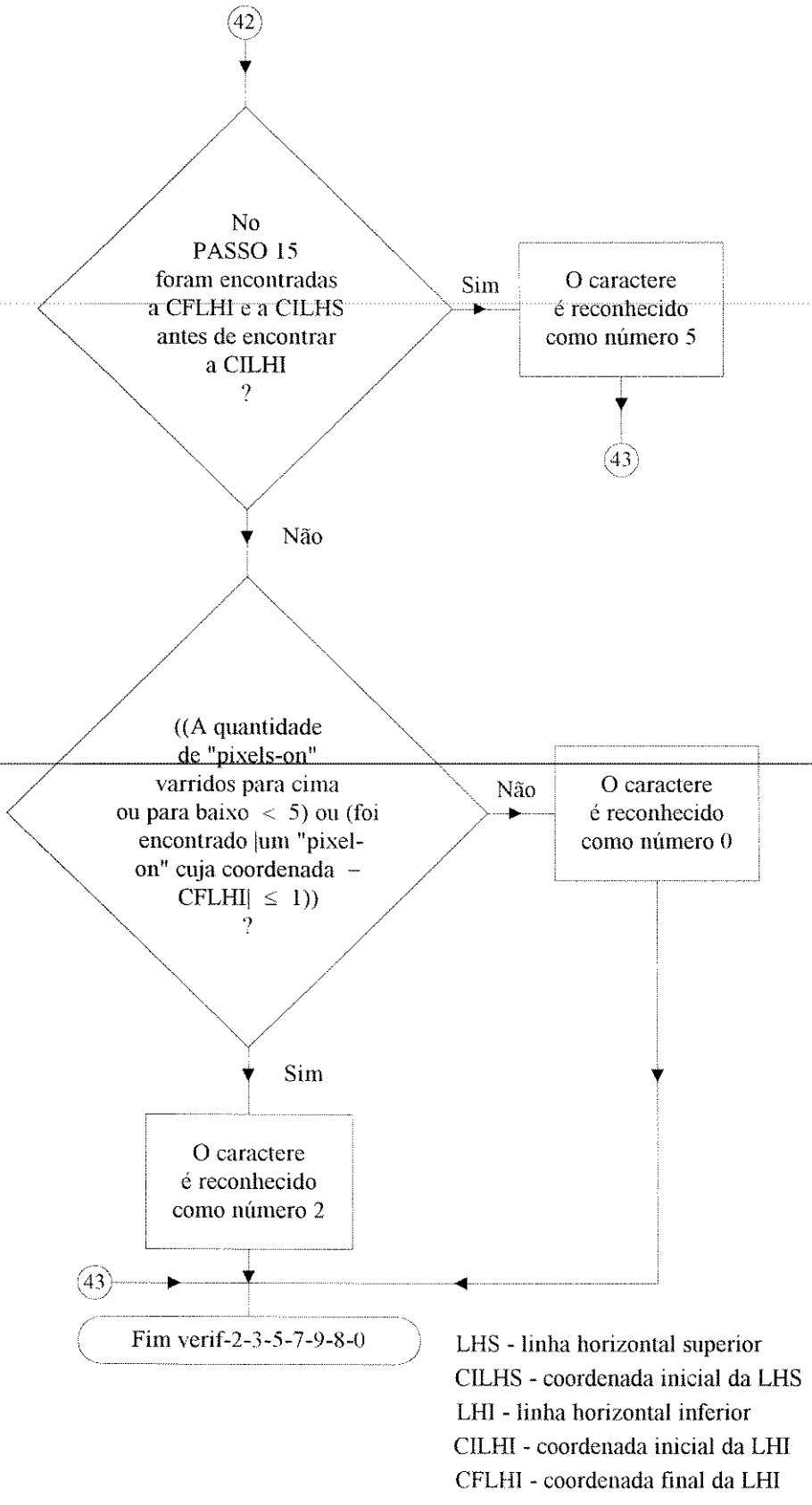


Figura 4.21(b1) - Vigésima oitava parte do fluxograma da função verif-2-3-5-7-9-8-0.

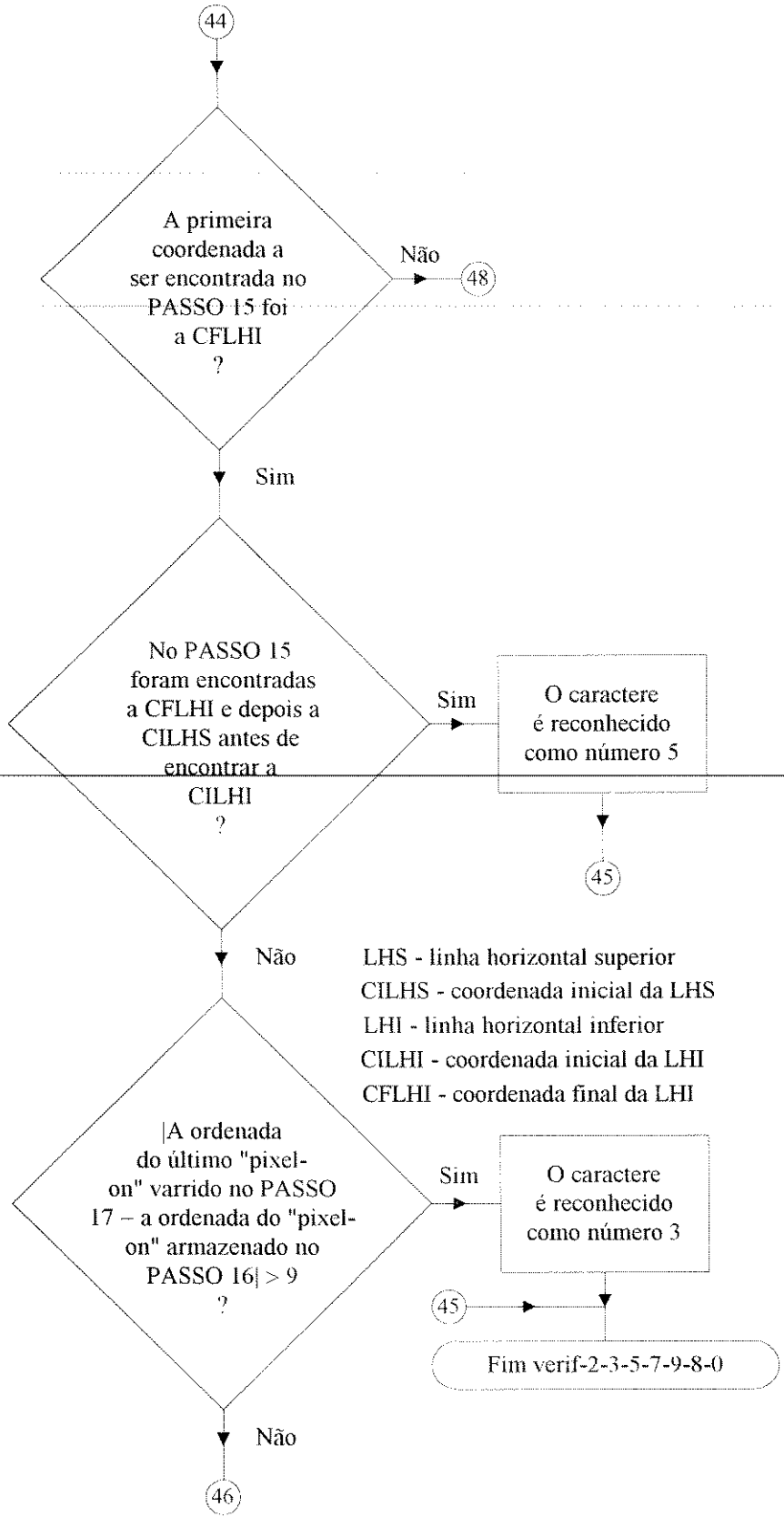
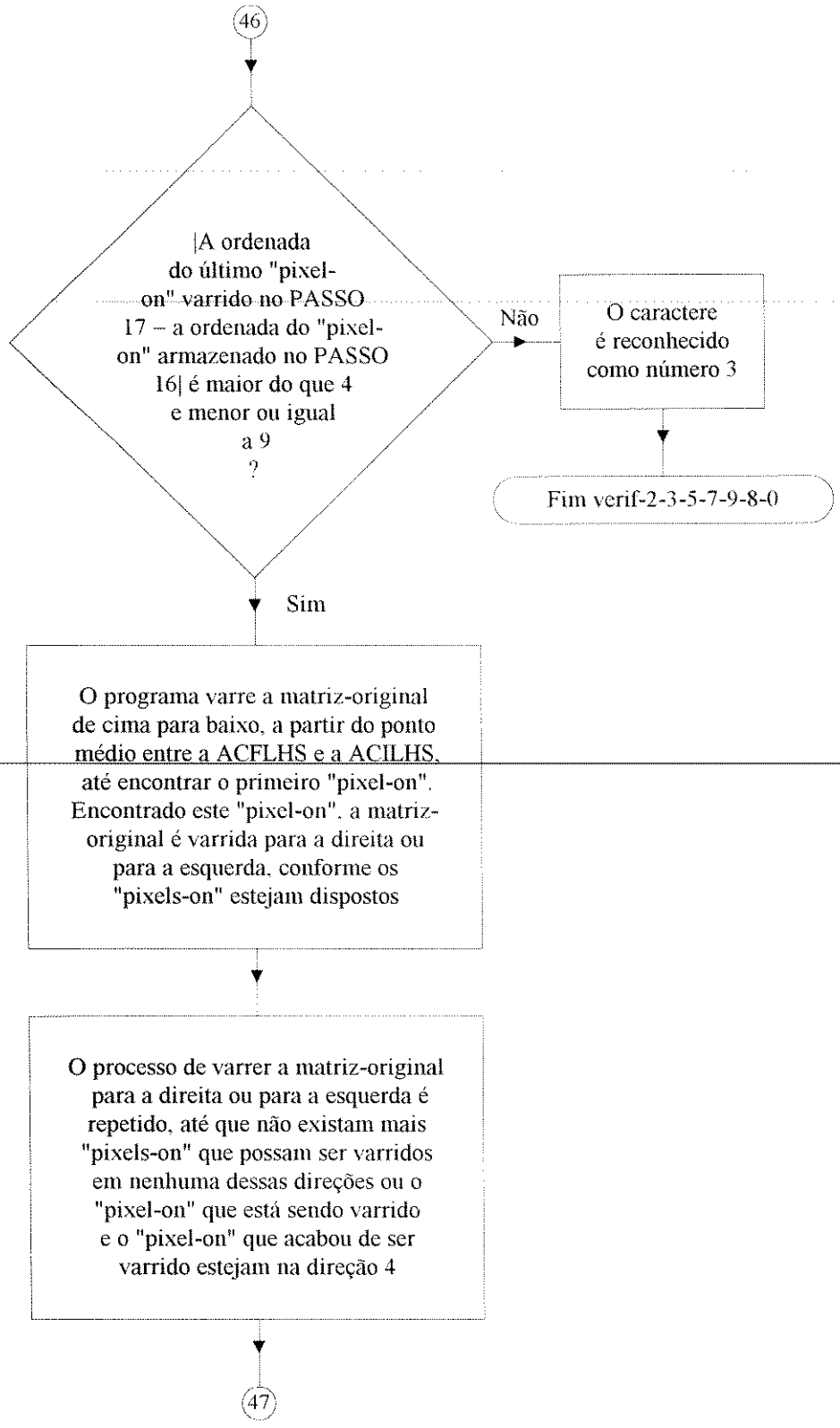


Figura 4.21(c1) - Vigésima nona parte do fluxograma da função verif-2-3-5-7-9-8-0.



LHS - linha horizontal superior
ACILHS - abscissa da coordenada inicial da LHS
ACFLHS - abscissa da coordenada final da LHS

Figura 4.21(d1) - Trigésima parte do fluxograma da função verif-2-3-5-7-9-8-0.

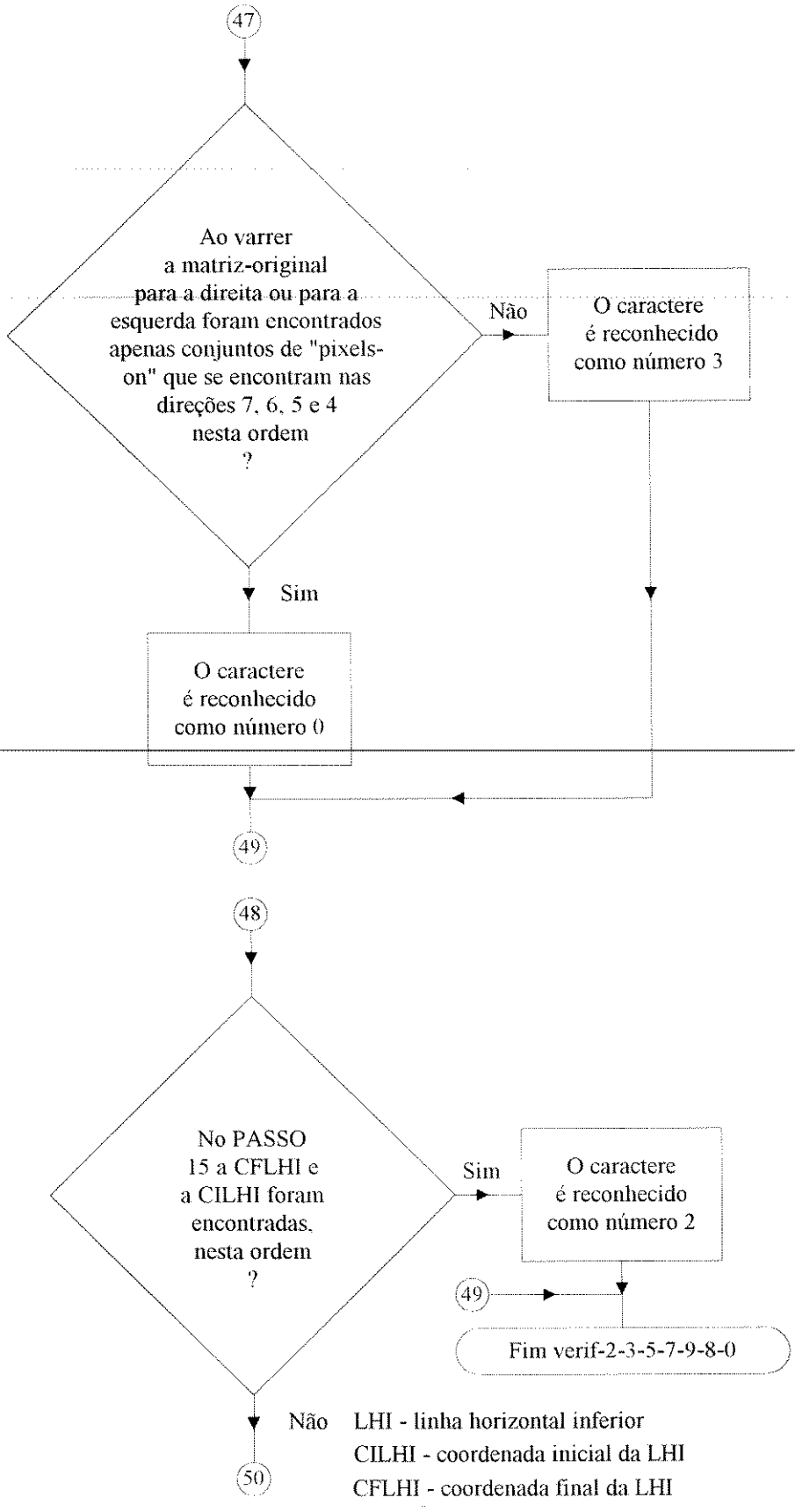


Figura 4.21(e1) - Trigésima primeira parte do fluxograma da função verif-2-3-5-7-9-8-0.

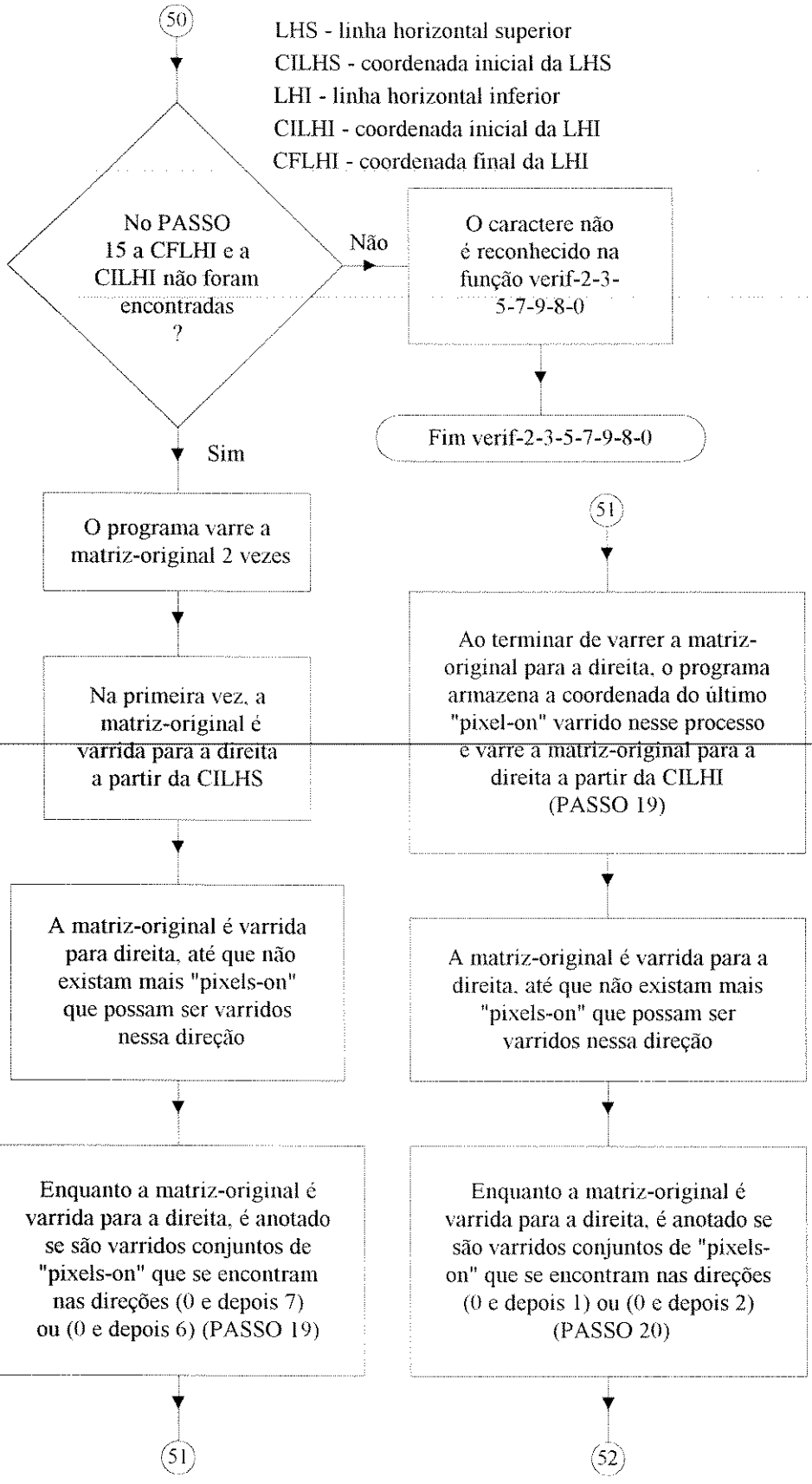


Figura 4.21(f1) - Trigésima segunda parte do fluxograma da função verif-2-3-5-7-9-8-0.

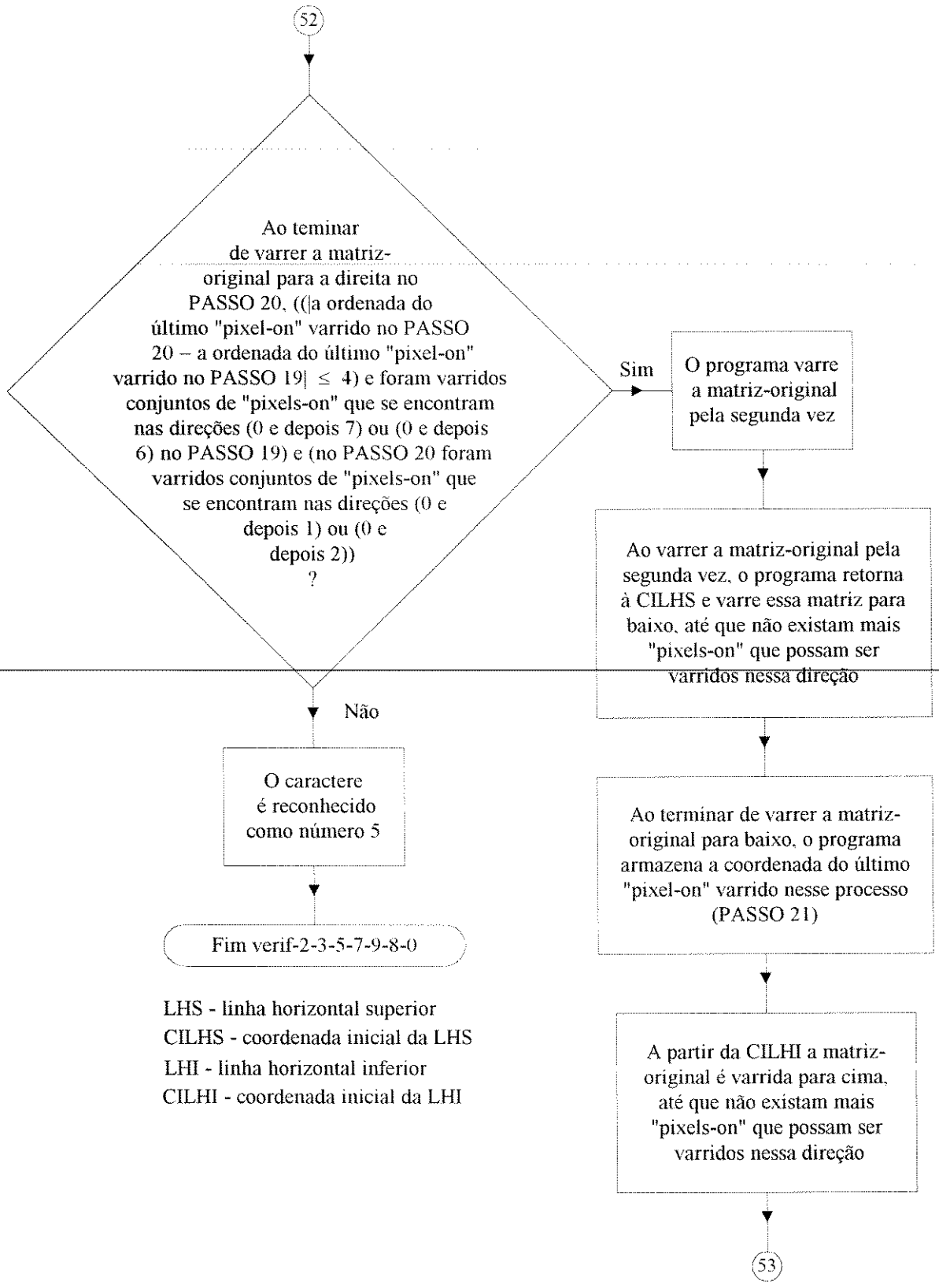


Figura 4.21(g1) - Trigésima terceira parte do fluxograma da função verif-2-3-5-7-9-8-0.

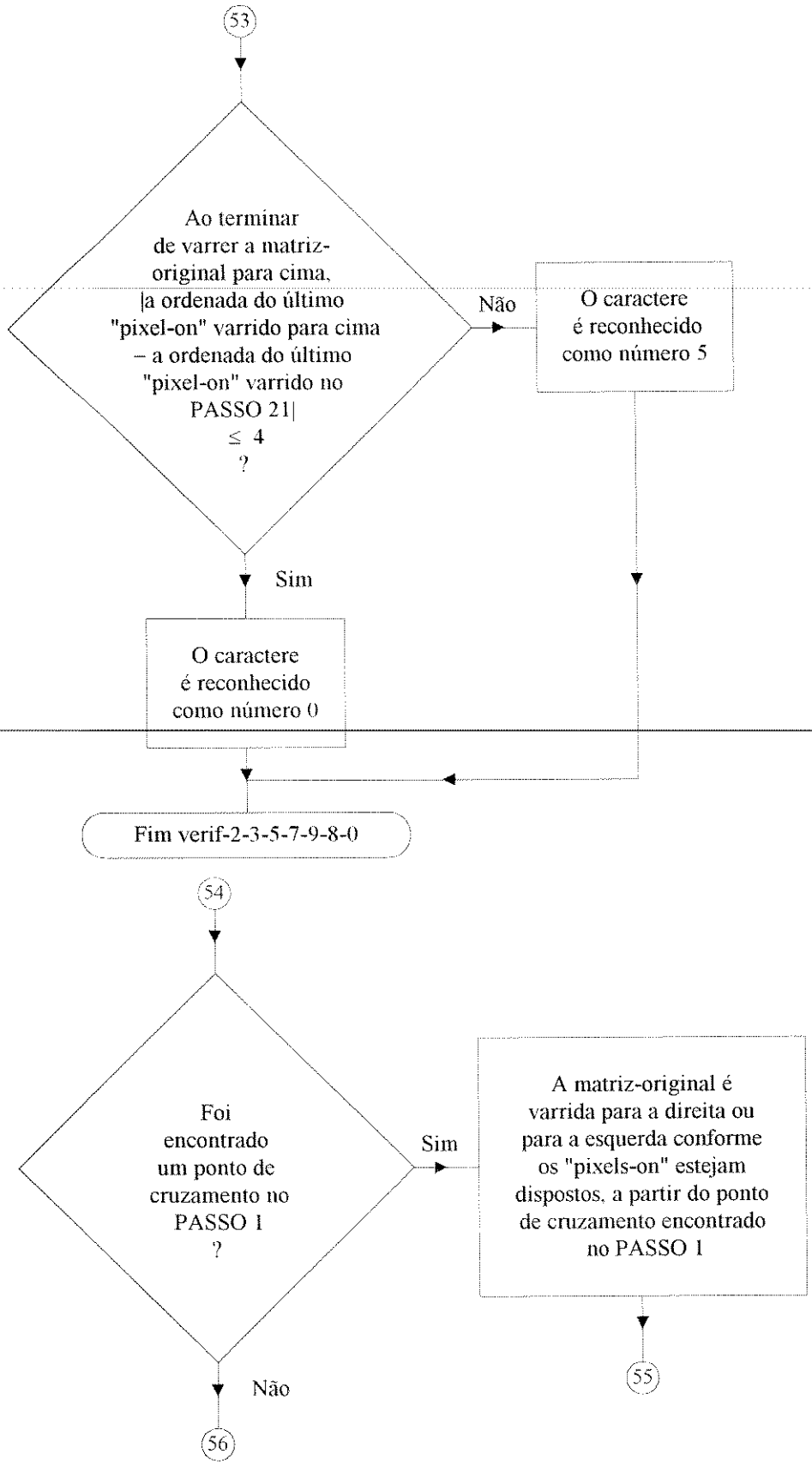


Figura 4.21(h1) - Trigésima quarta parte do fluxograma da função verif-2-3-5-7-9-8-0.

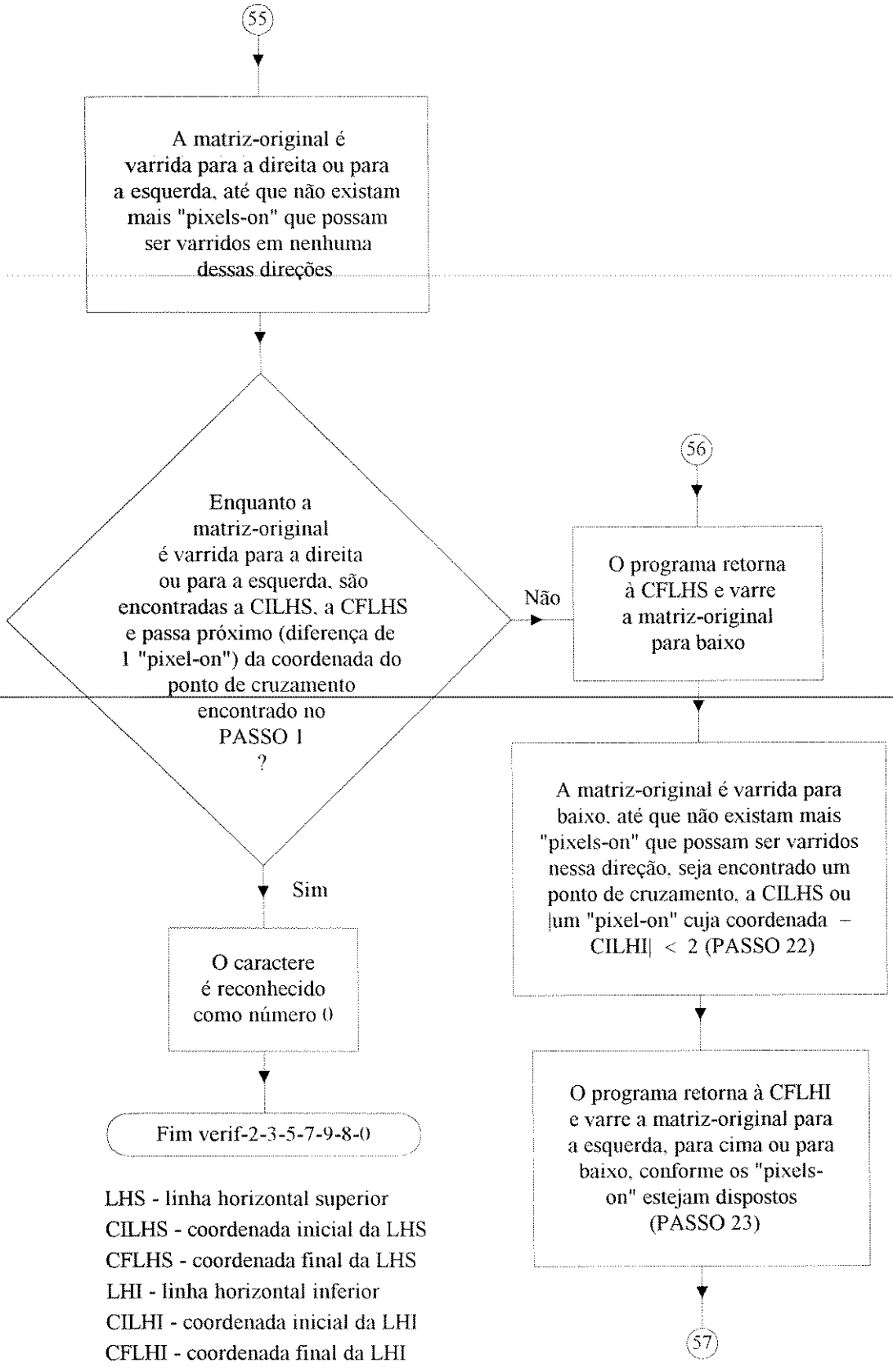


Figura 4.21(i1) - Trigésima quinta parte do fluxograma da função verif-2-3-5-7-9-8-0.

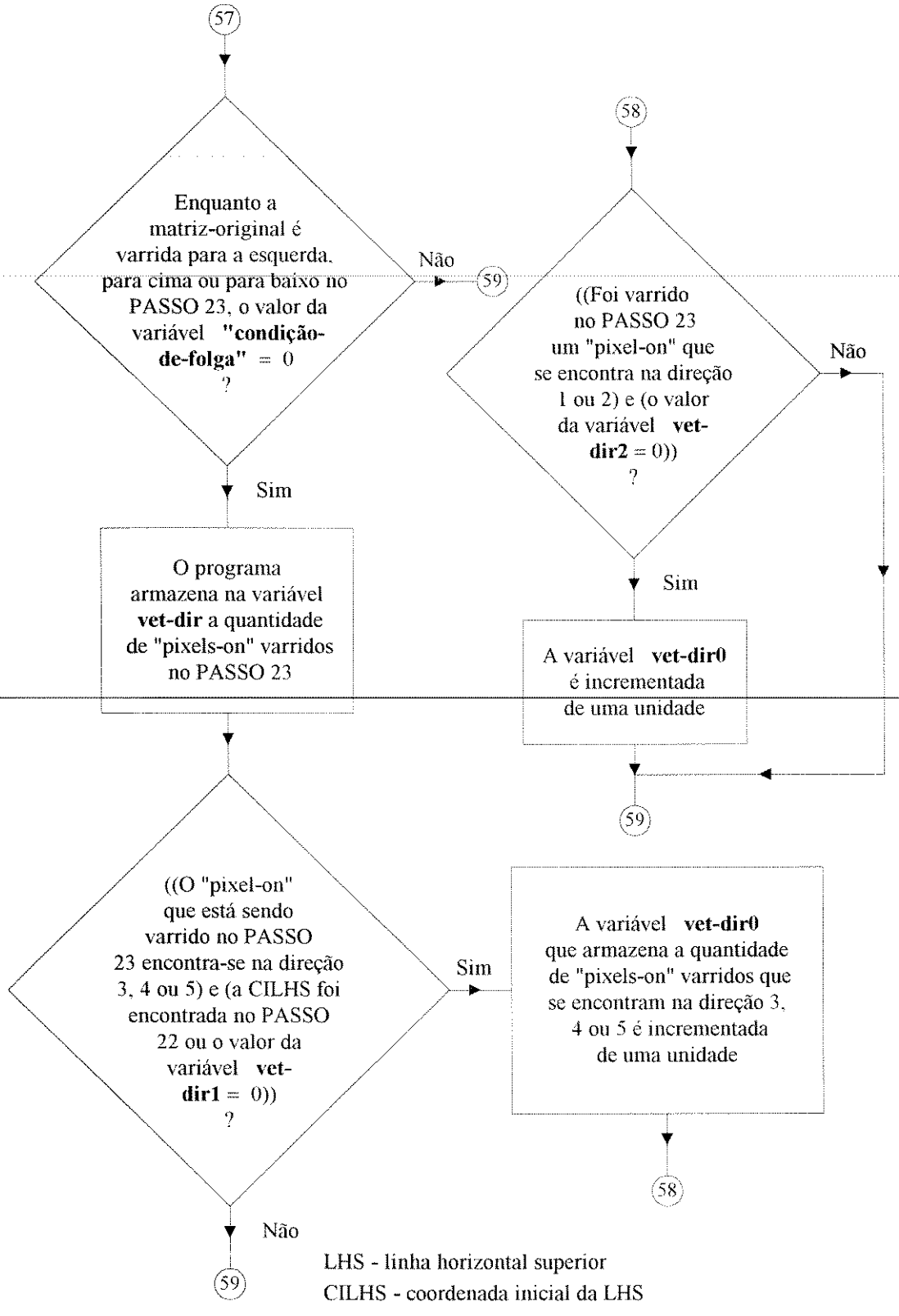


Figura 4.21(j1) - Trigésima sexta parte do fluxograma da função verif-2-3-5-7-9-8-0.

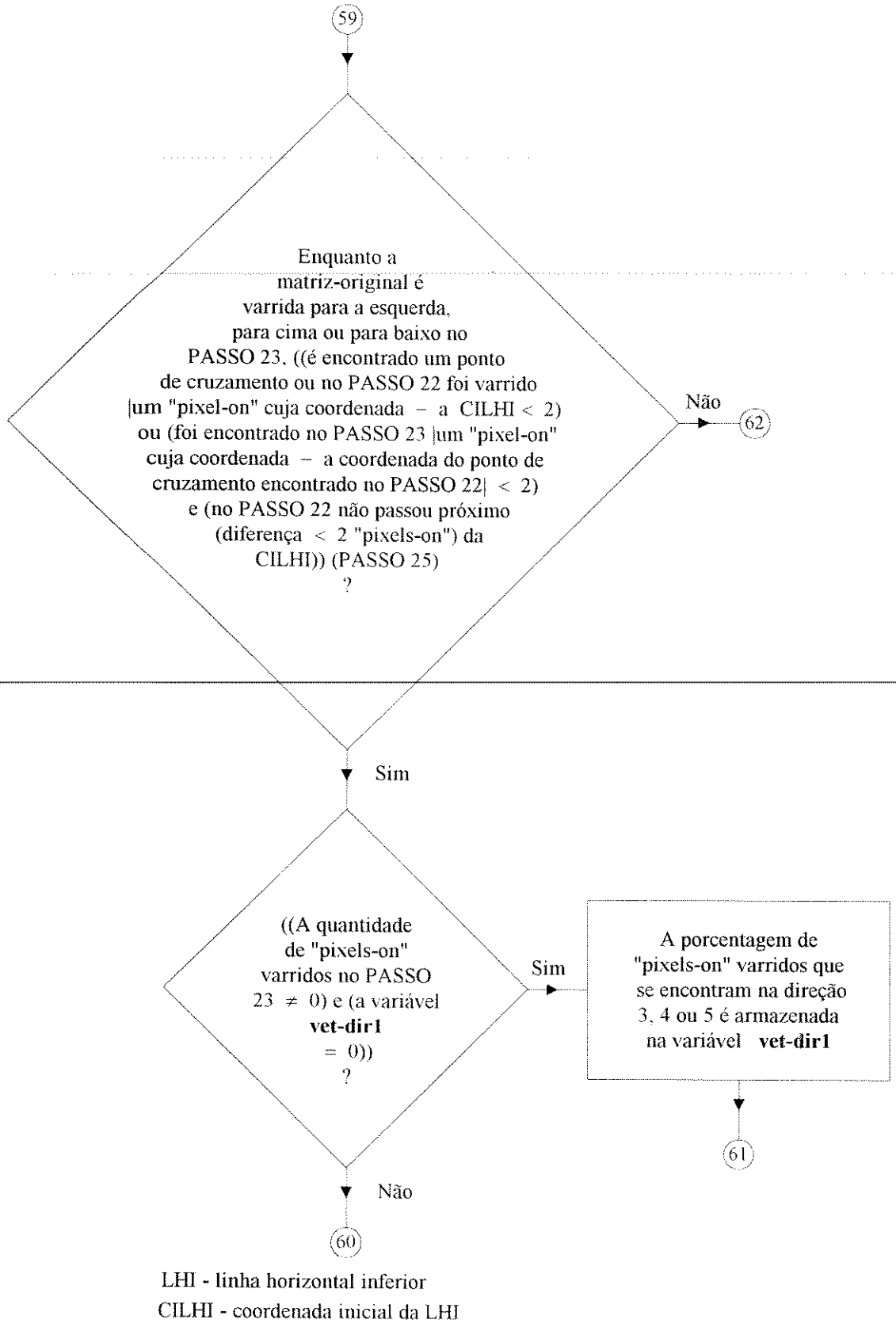


Figura 4.21(k1) - Trigésima sétima parte do fluxograma da função verif-2-3-5-7-9-8-0.

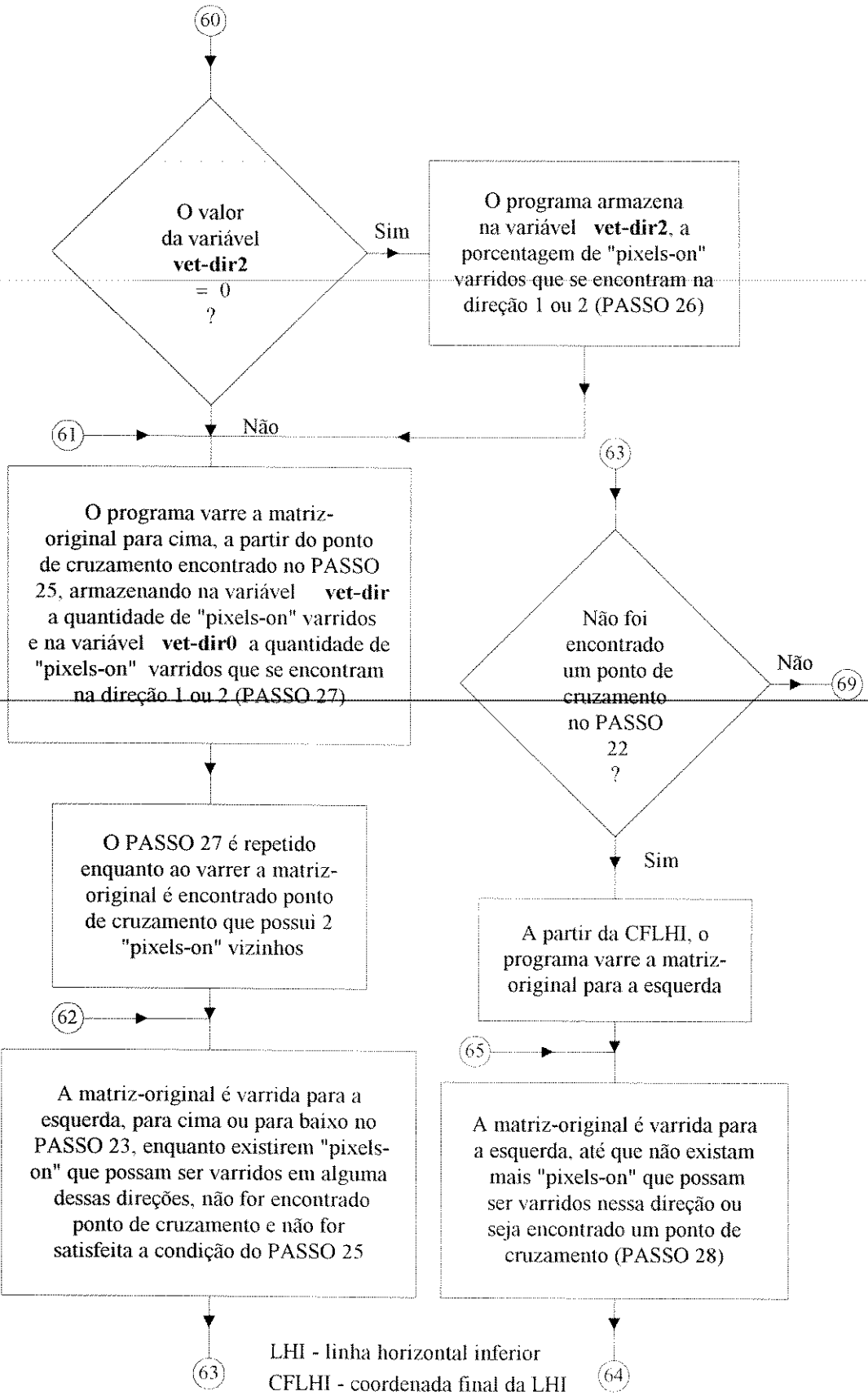
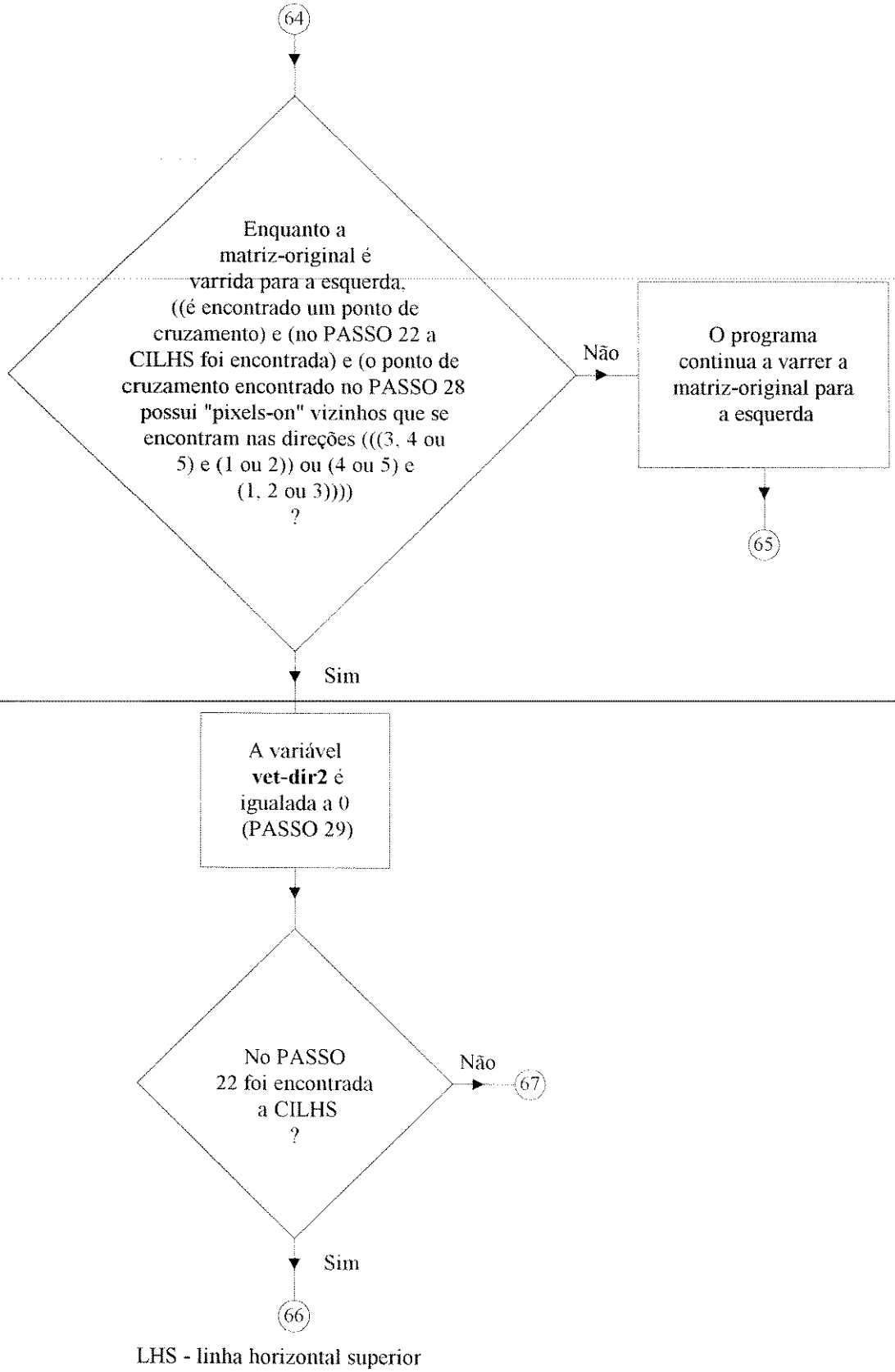


Figura 4.21(11) - Trigésima oitava parte do fluxograma da função verif-2-3-5-7-9-8-0.



LHS - linha horizontal superior
CILHS - coordenada inicial da LHS

Figura 4.21(m1) - Trigésima nona parte do fluxograma da função verif-2-3-5-7-9-8-0.

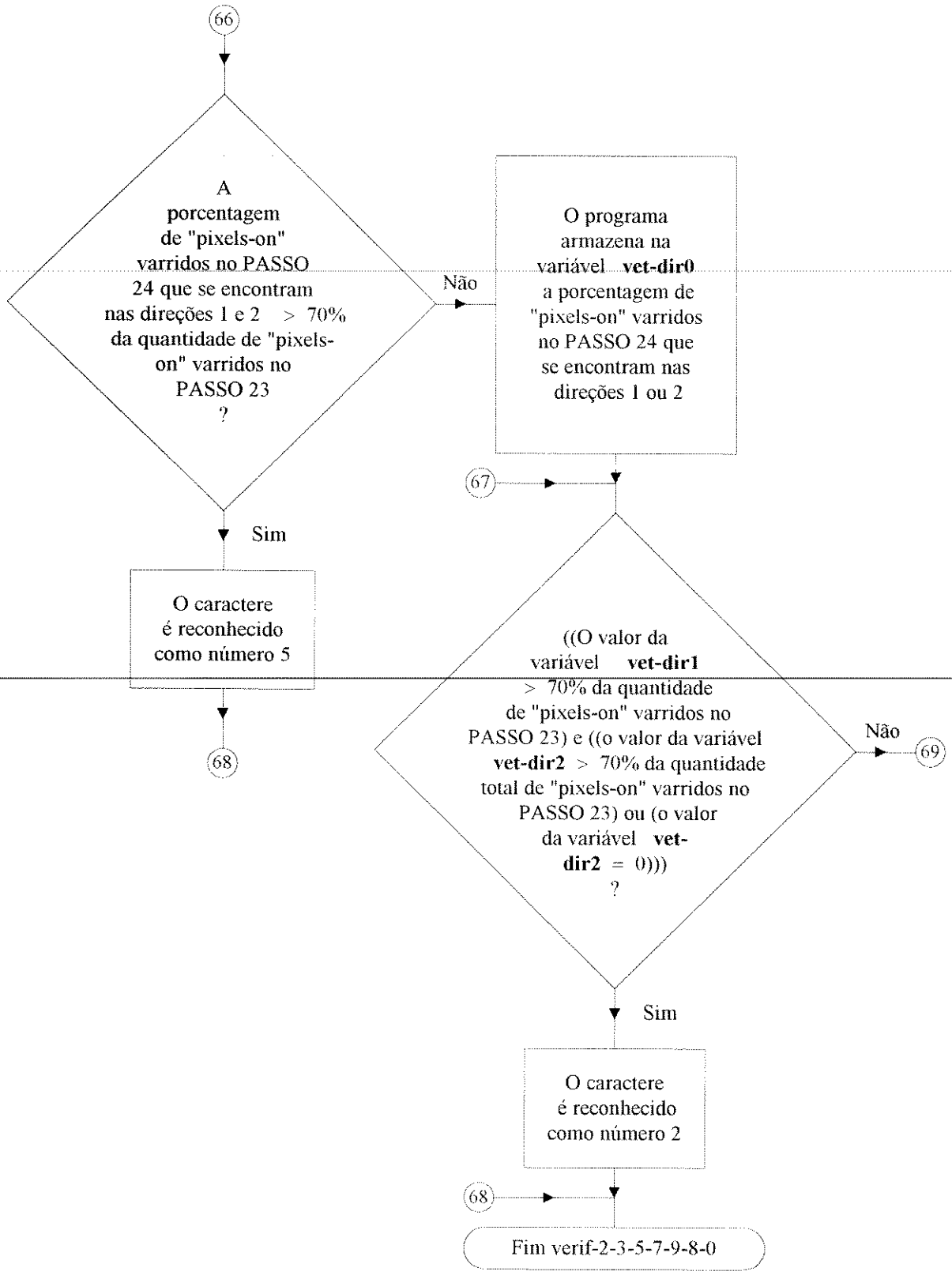


Figura 4.21(n1) - Quadragésima parte do fluxograma da função verif-2-3-5-7-9-8-0.

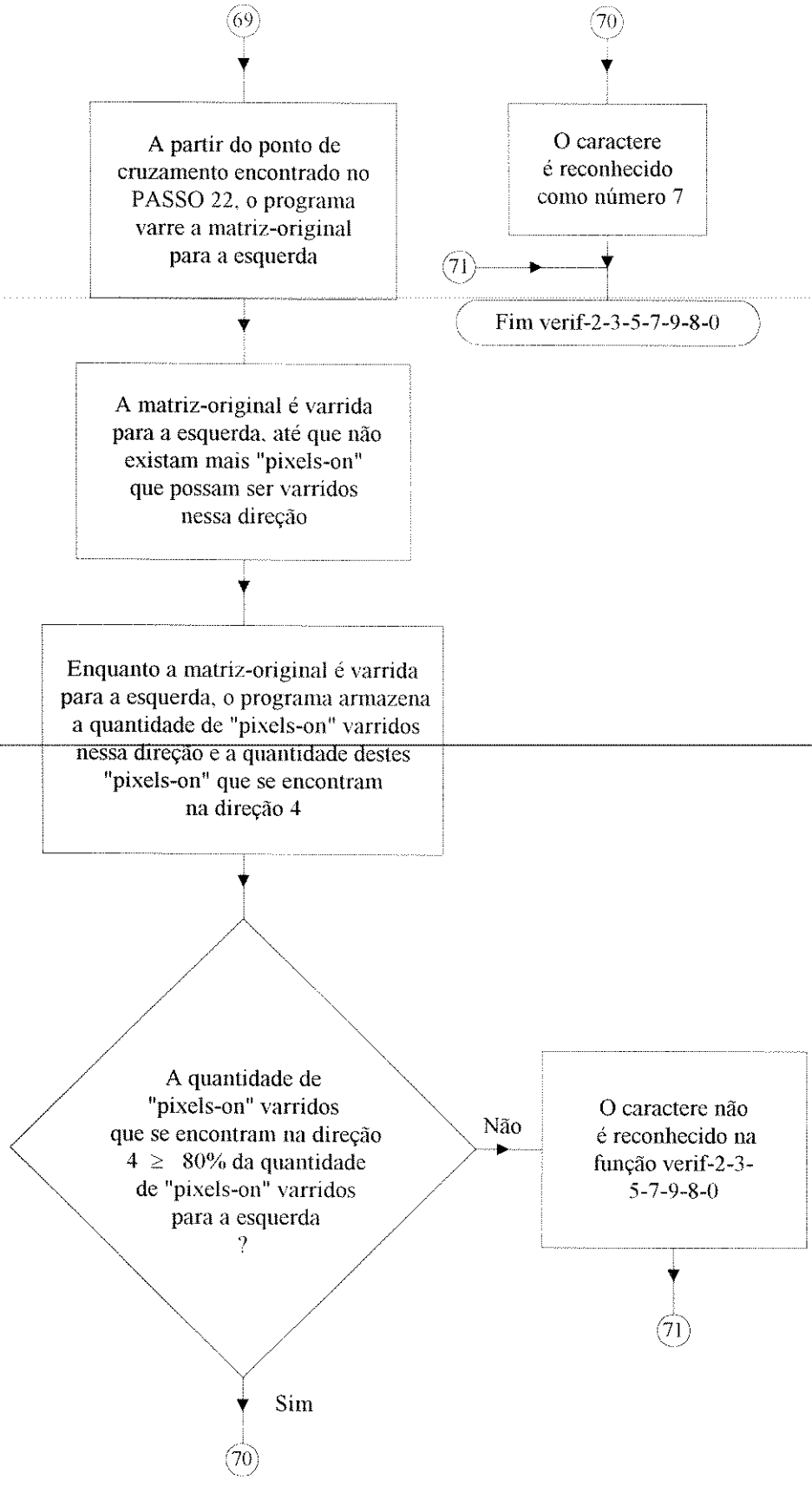


Figura 4.21(o1) - Quadragésima primeira parte do fluxograma da função verif-2-3-5-7-9-8-0.

Fluxograma da função verif-7-9-4-1-2-0-6-8-3

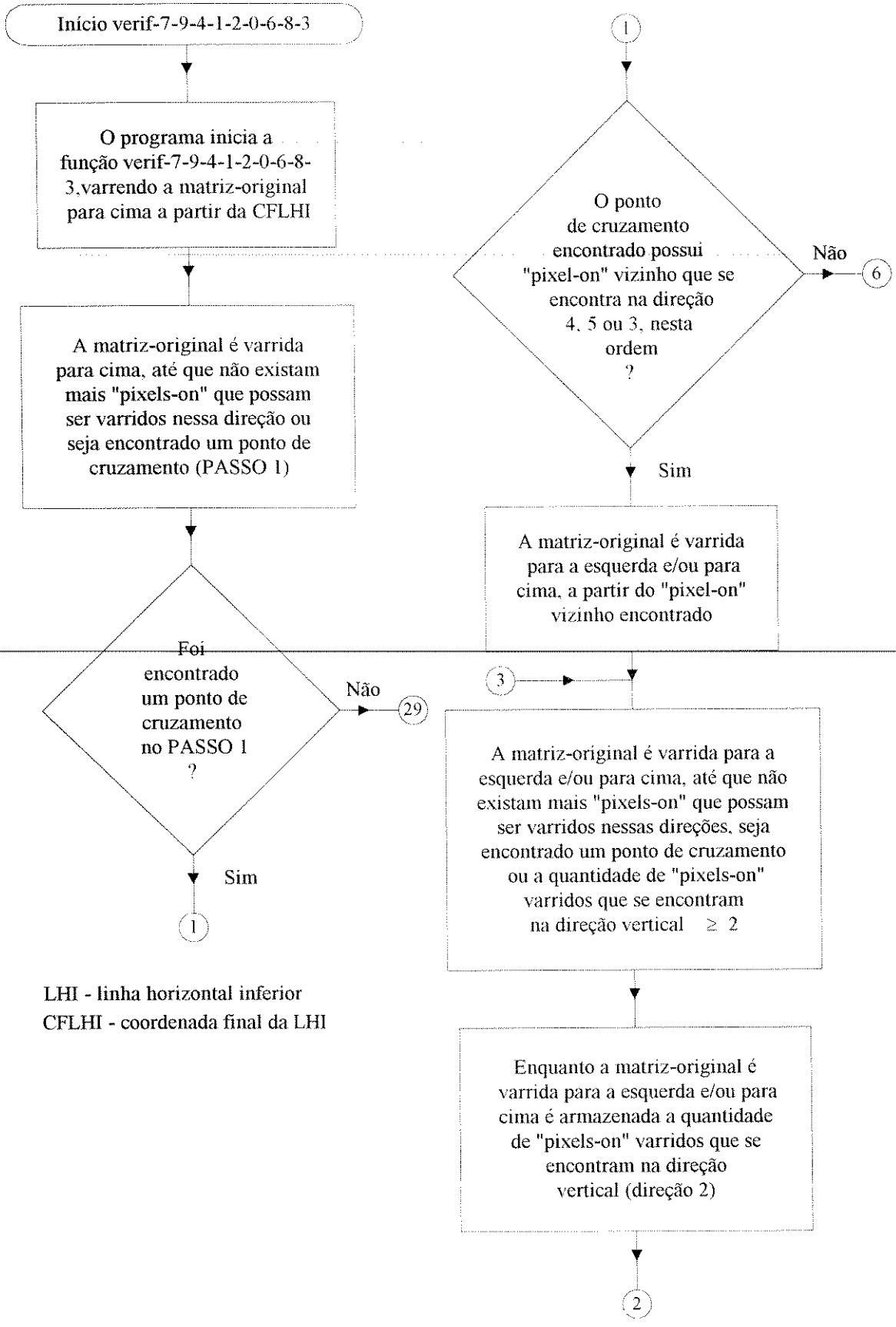


Figura 4.23(a) - Primeira parte do fluxograma da função verif-7-9-4-1-2-0-6-8-3.

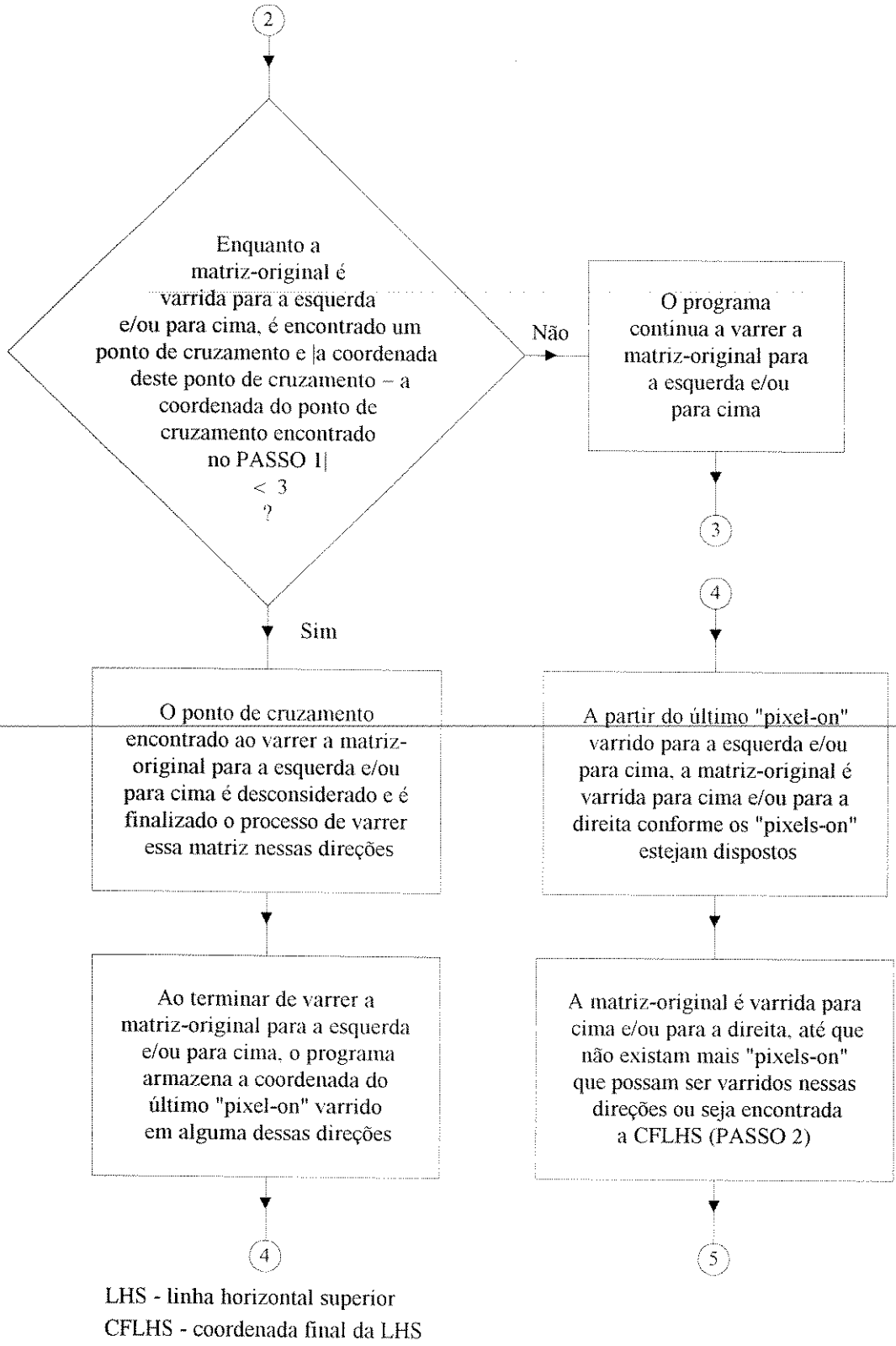


Figura 4.23(b) - Segunda parte do fluxograma da função verif-7-9-4-1-2-0-6-8-3.

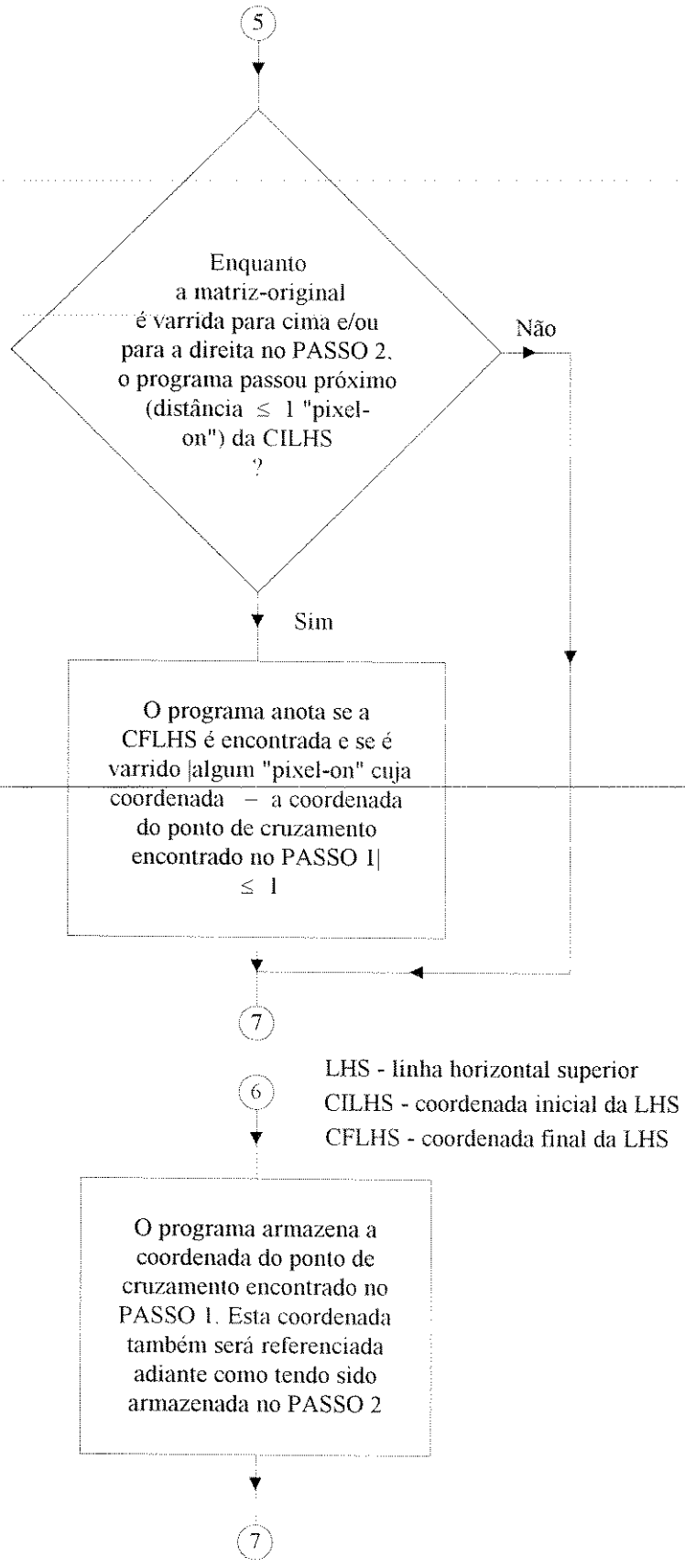


Figura 4.23(c) - Terceira parte do fluxograma da função verif-7-9-4-1-2-0-6-8-3.

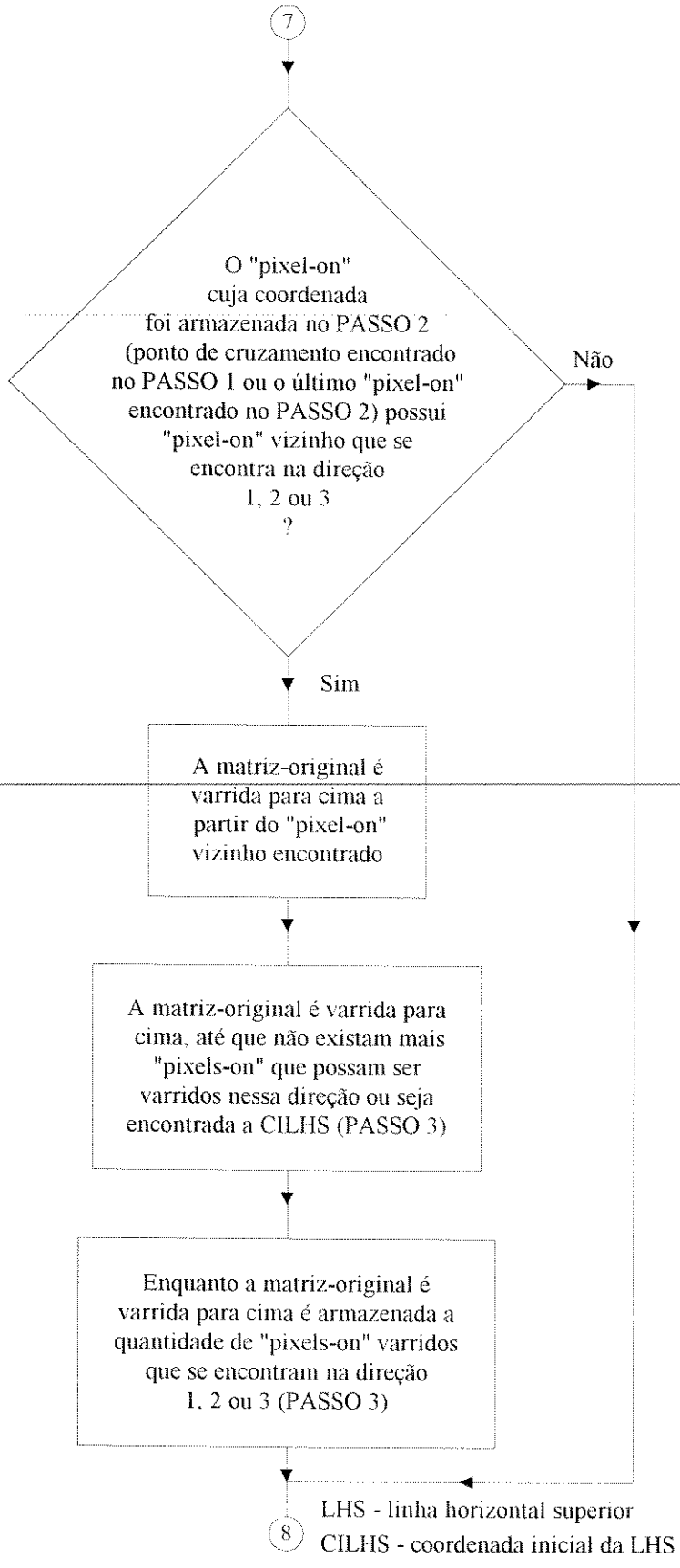


Figura 4.23(d) - Quarta parte do fluxograma da função verif-7-9-4-1-2-0-6-8-3.

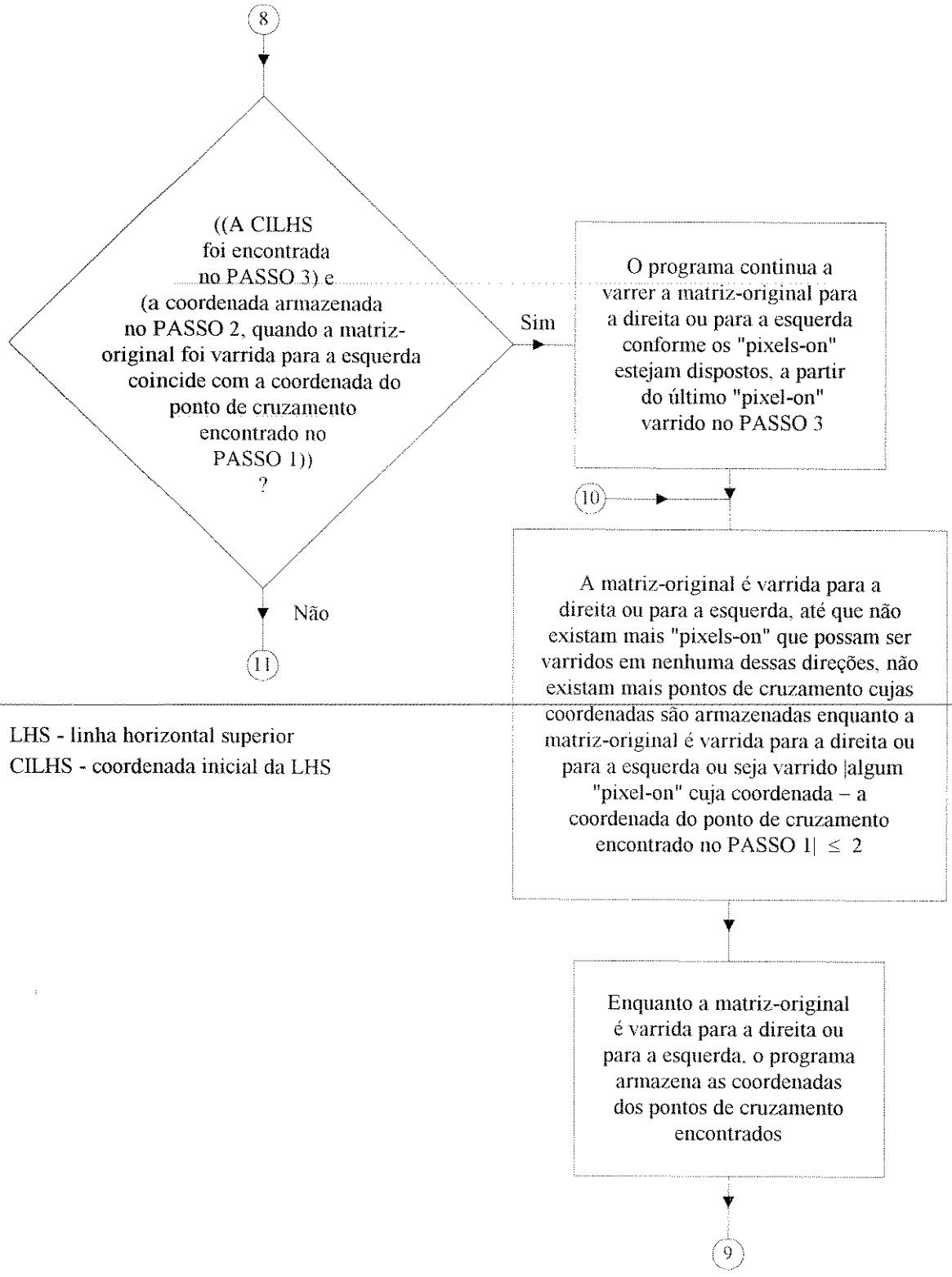


Figura 4.23(e) - Quinta parte do fluxograma da função verif-7-9-4-1-2-0-6-8-3.

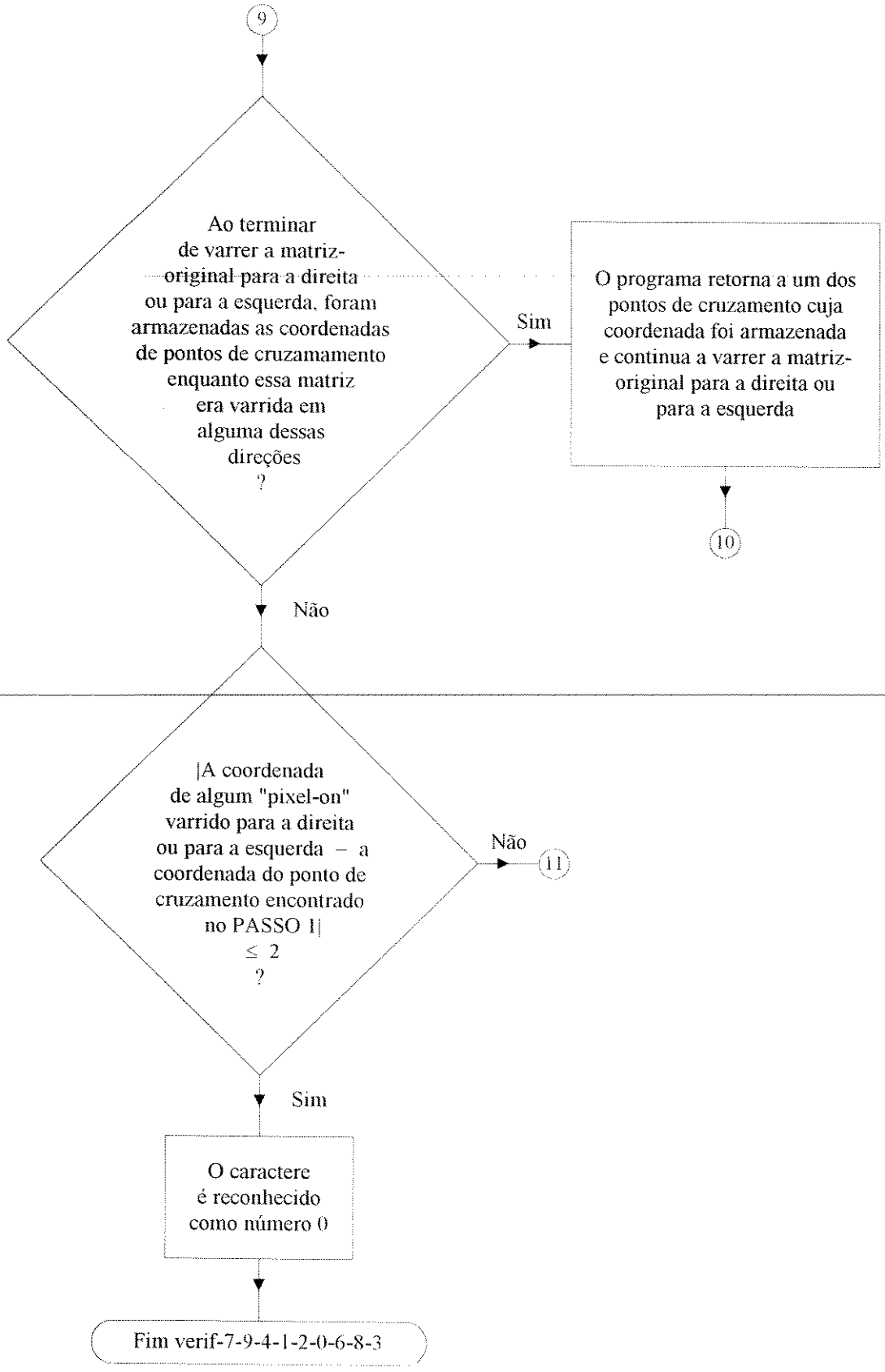


Figura 4.23(f) - Sexta parte do fluxograma da função verif-7-9-4-1-2-0-6-8-3.

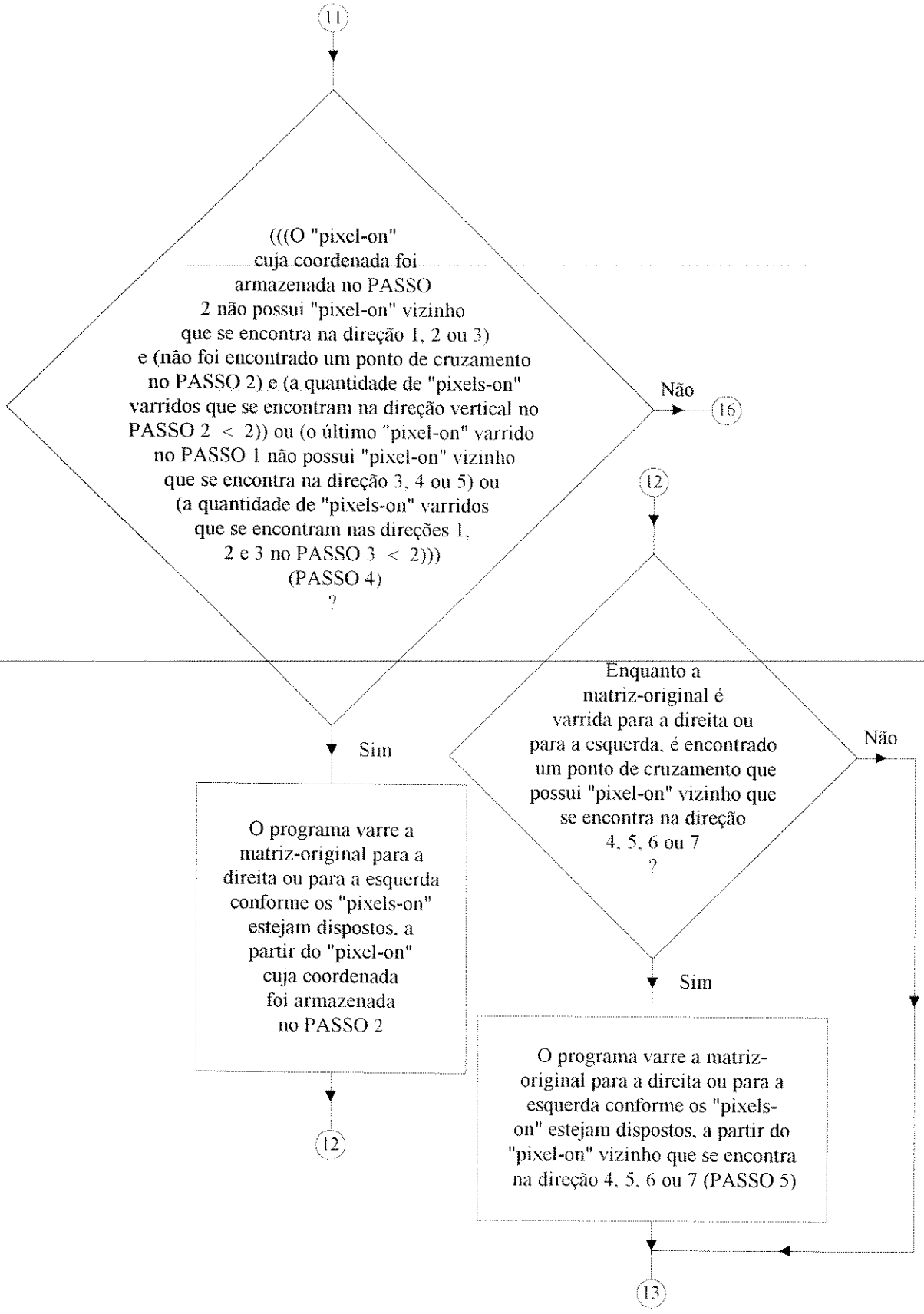


Figura 4.23(g) - Sétima parte do fluxograma da função verif-7-9-4-1-2-0-6-8-3.

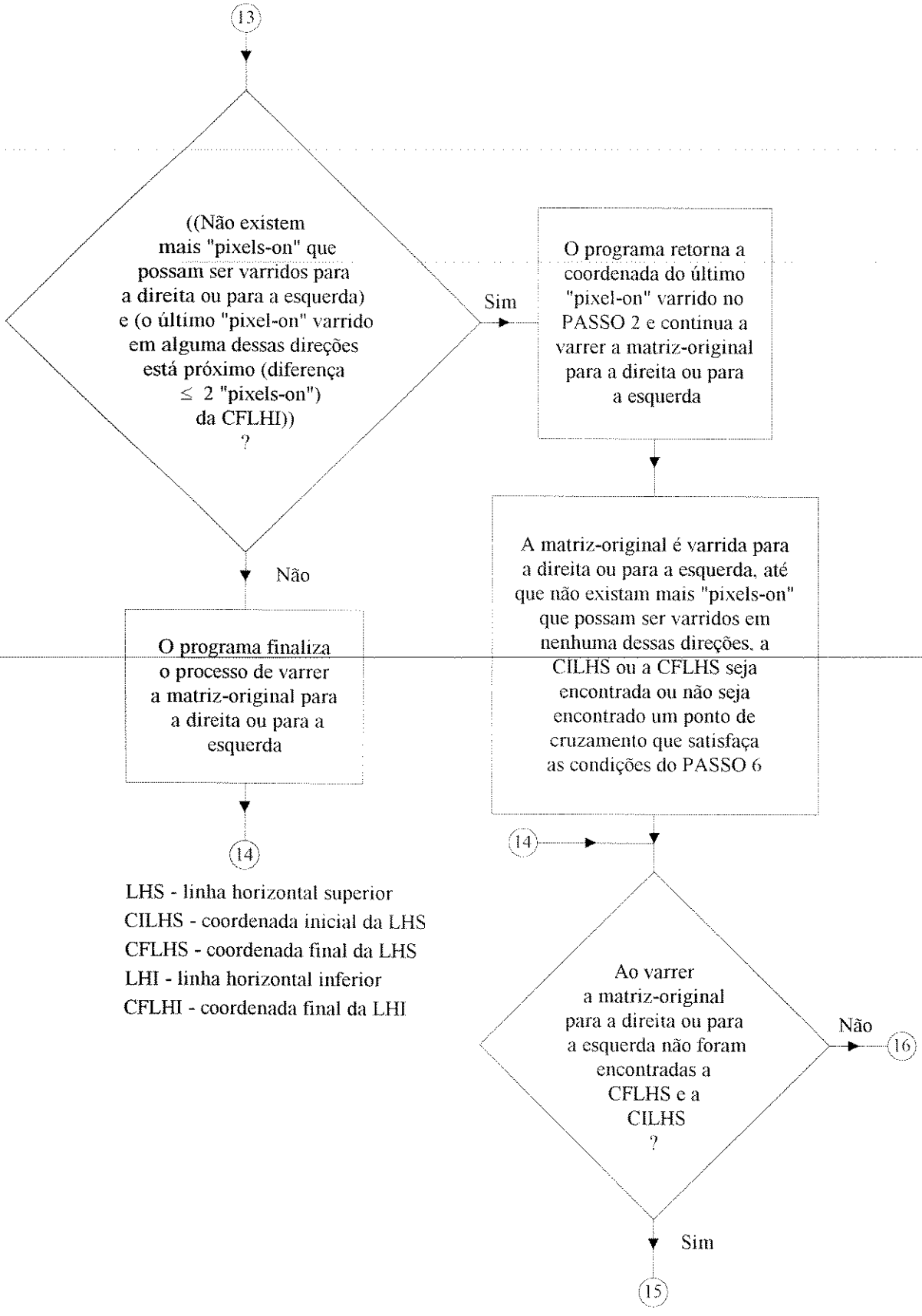


Figura 4.23(h) - Oitava parte do fluxograma da função verif-7-9-4-1-2-0-6-8-3.

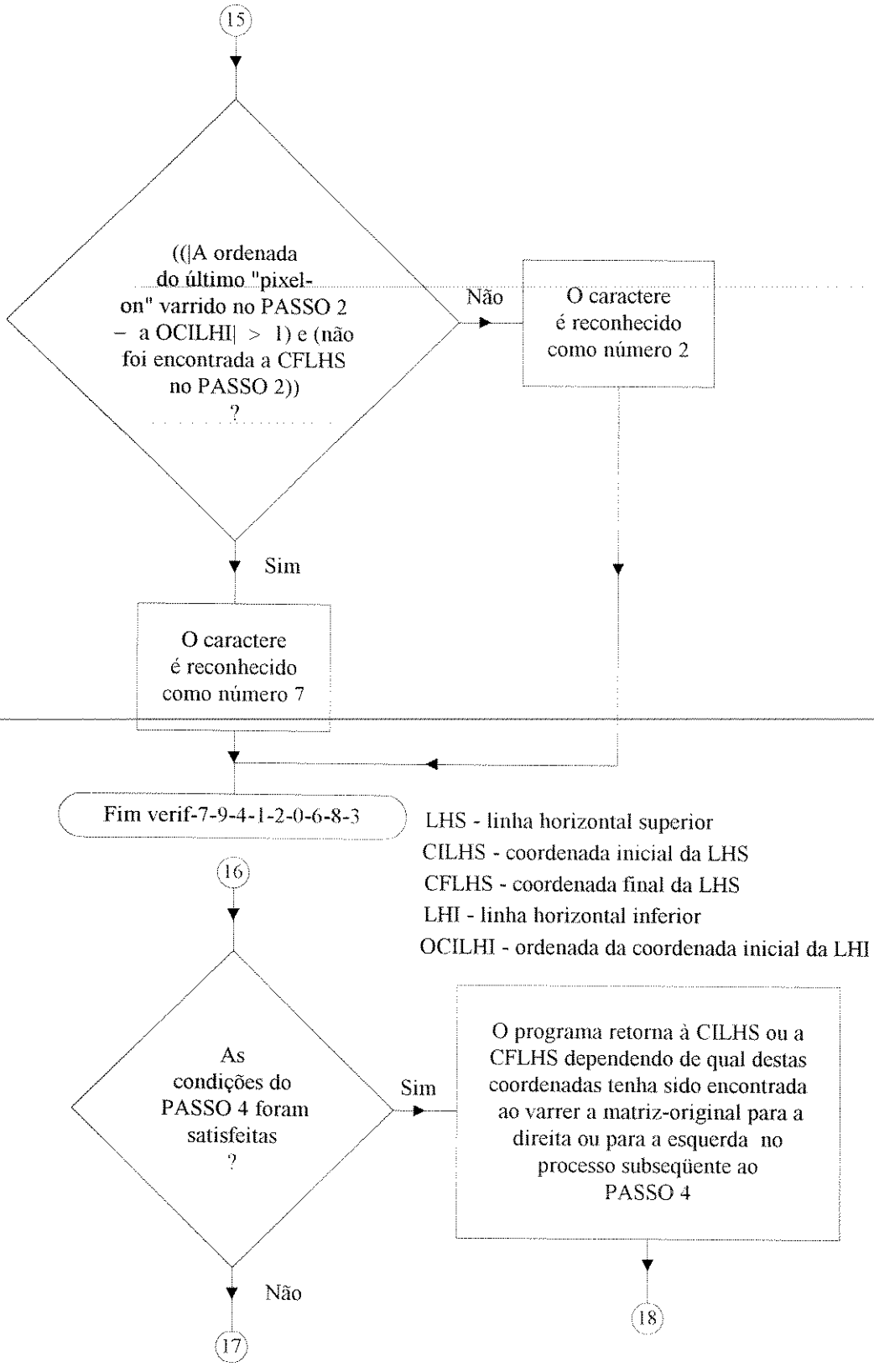


Figura 4.23(i) - Nona parte do fluxograma da função verif-7-9-4-1-2-0-6-8-3.

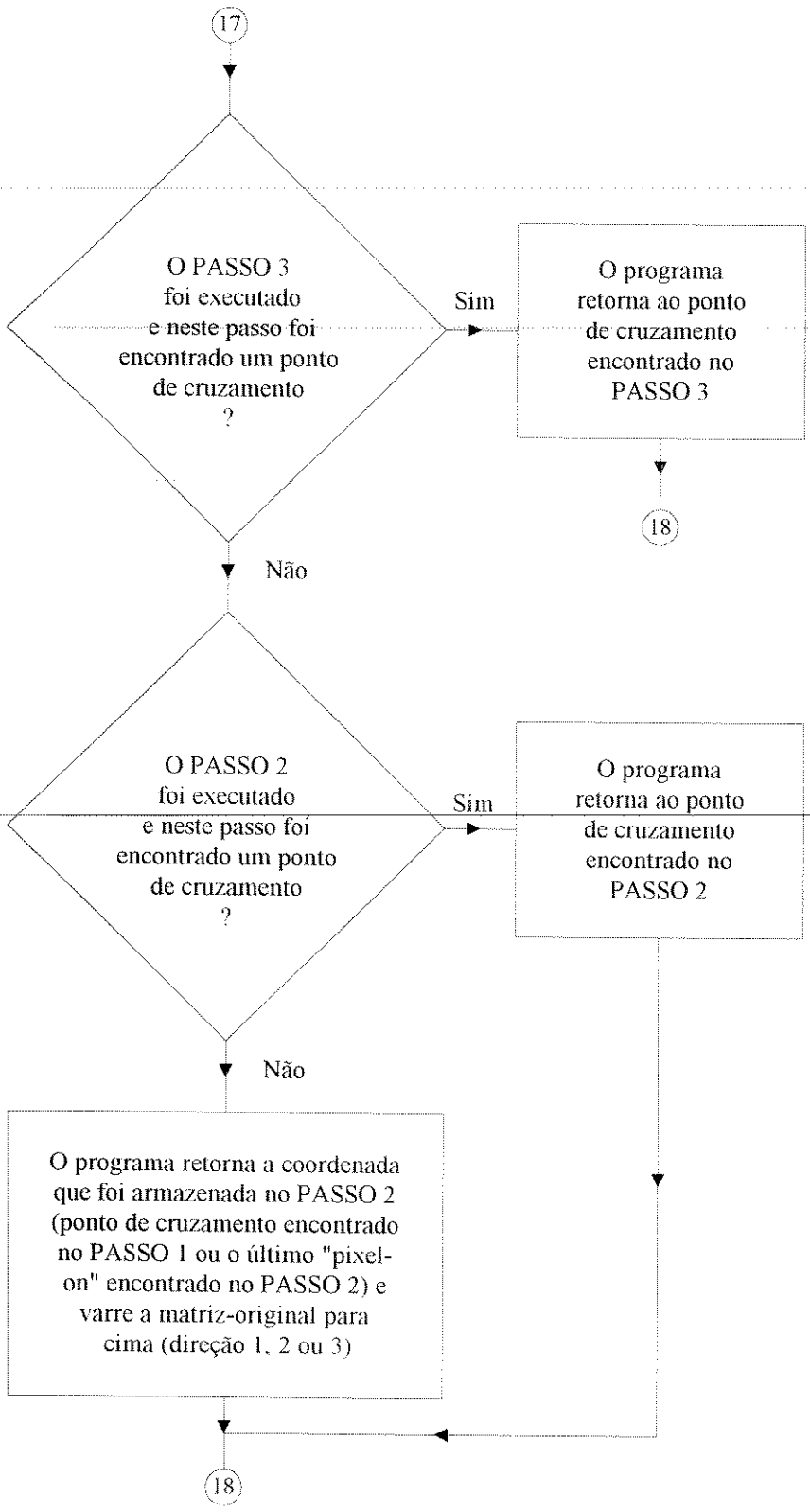
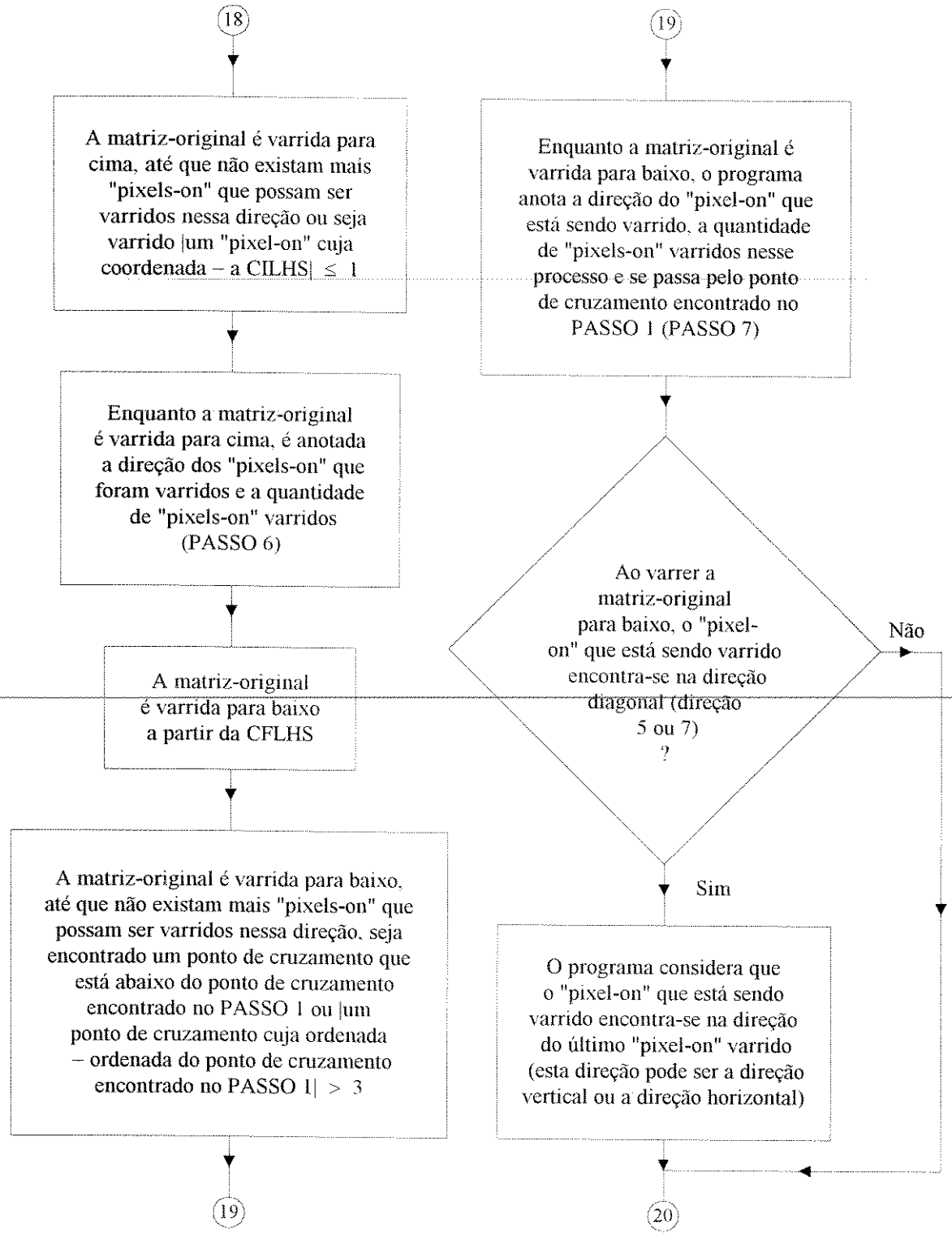


Figura 4.23(j) - Décima parte do fluxograma da função verif-7-9-4-1-2-0-6-8-3.



LHS - linha horizontal superior
CILHS - coordenada inicial da LHS
CFLHS - coordenada final da LHS

Figura 4.23(k) - Décima primeira parte do fluxograma da função verif-7-9-4-1-2-0-6-8-3.

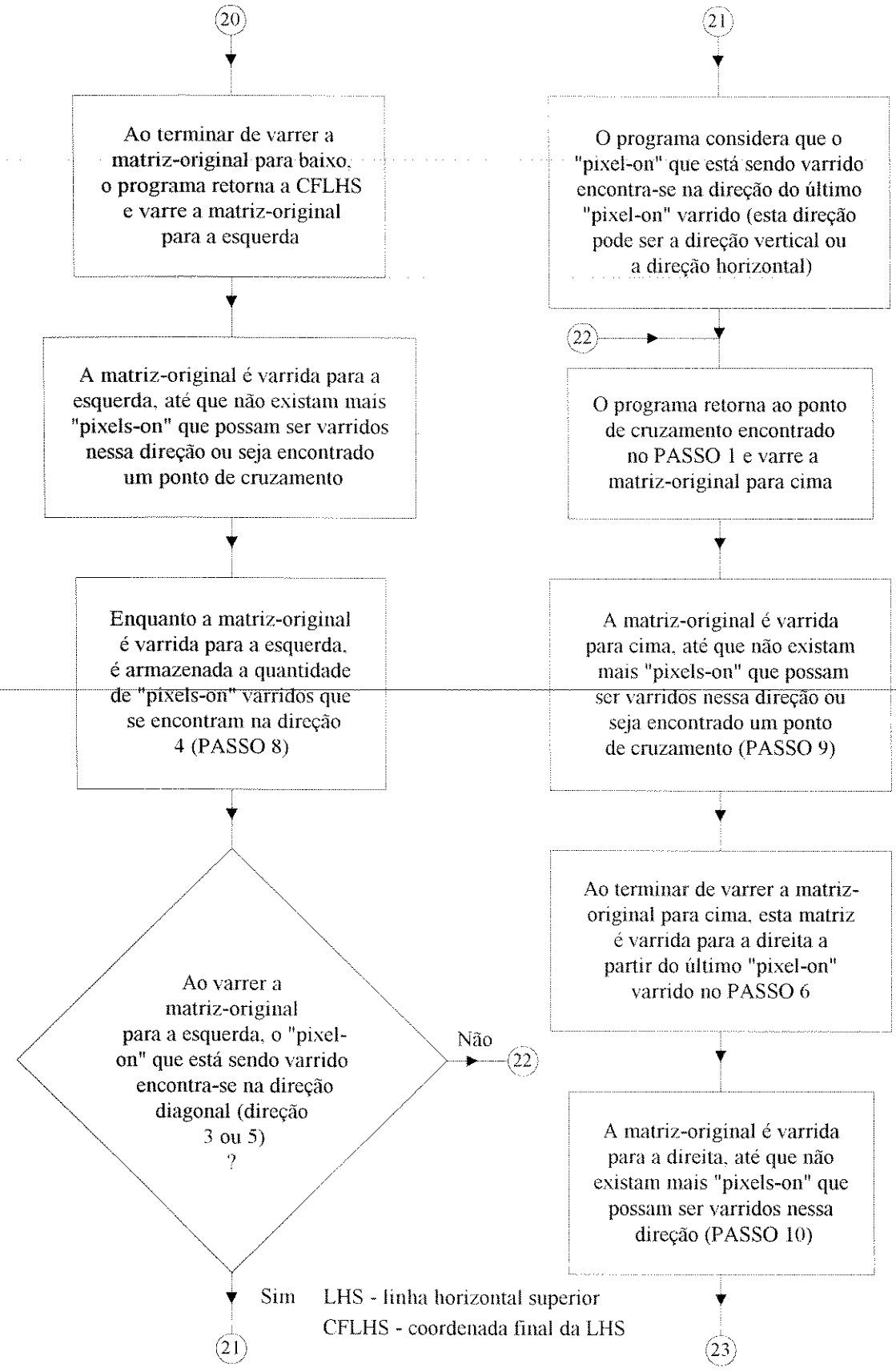


Figura 4.23(l) - Décima segunda parte do fluxograma da função verif-7-9-4-1-2-0-6-8-3.

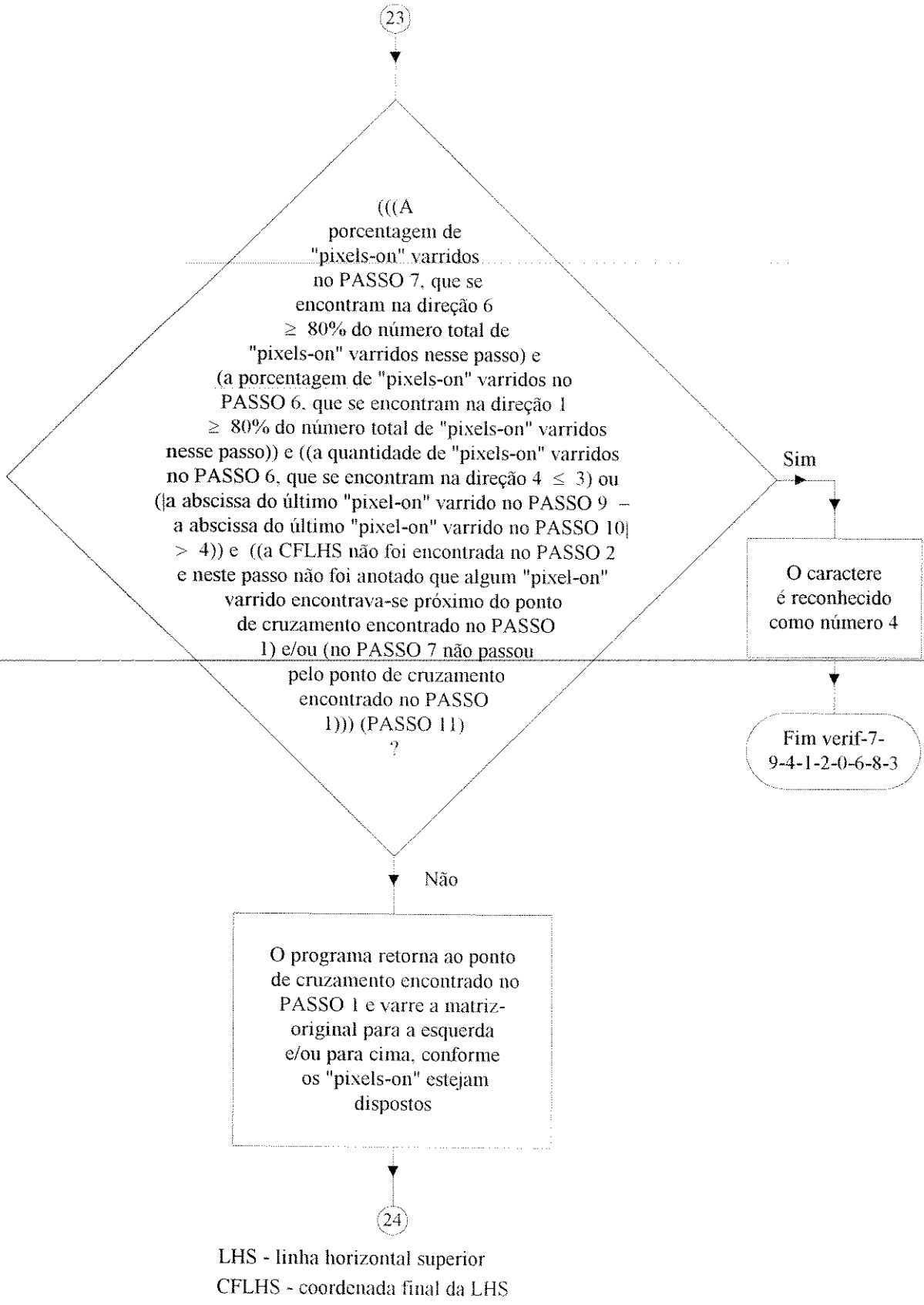


Figura 4.23(m) - Décima terceira parte do fluxograma da função verif-7-9-4-1-2-0-6-8-3.

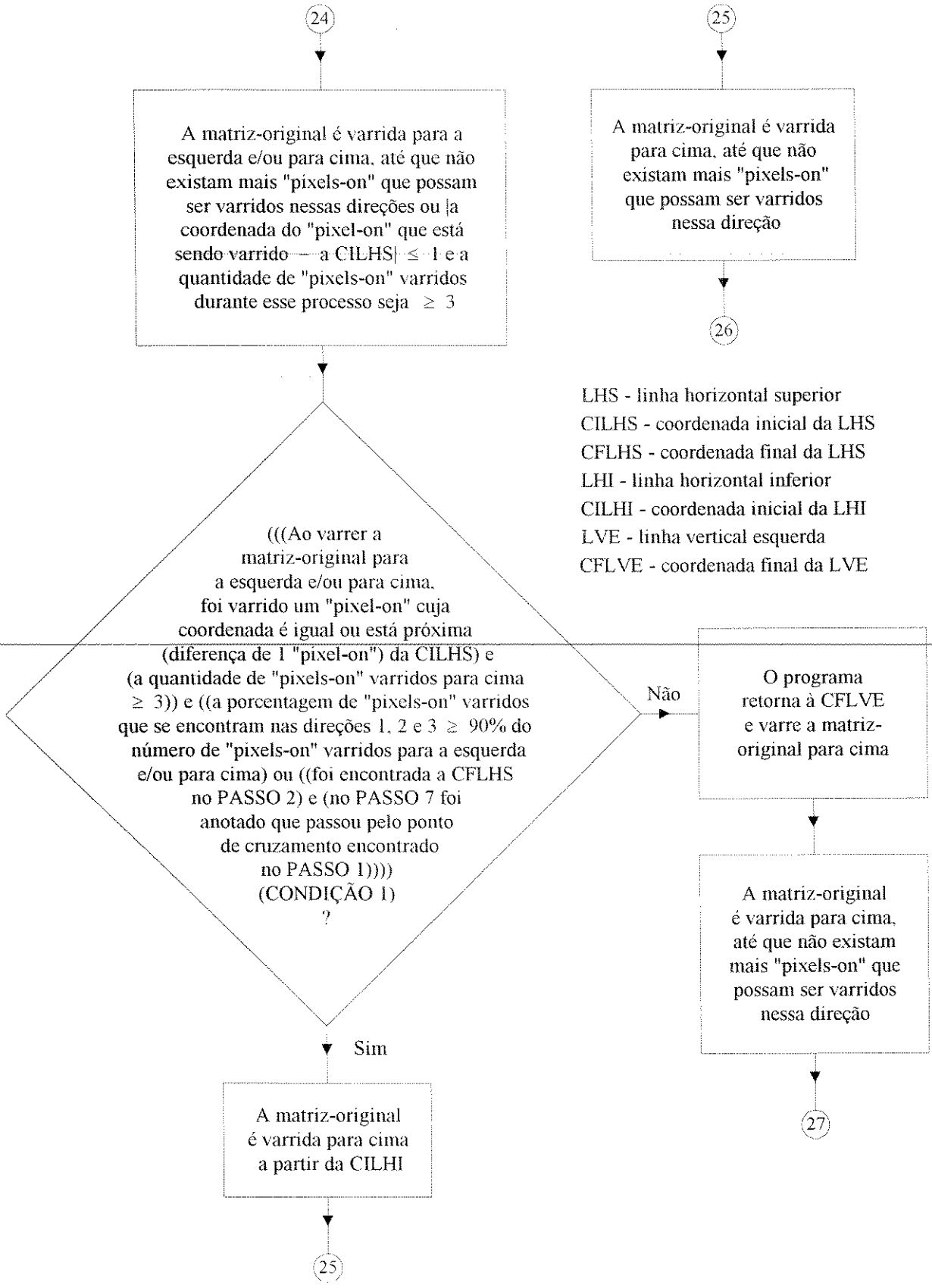
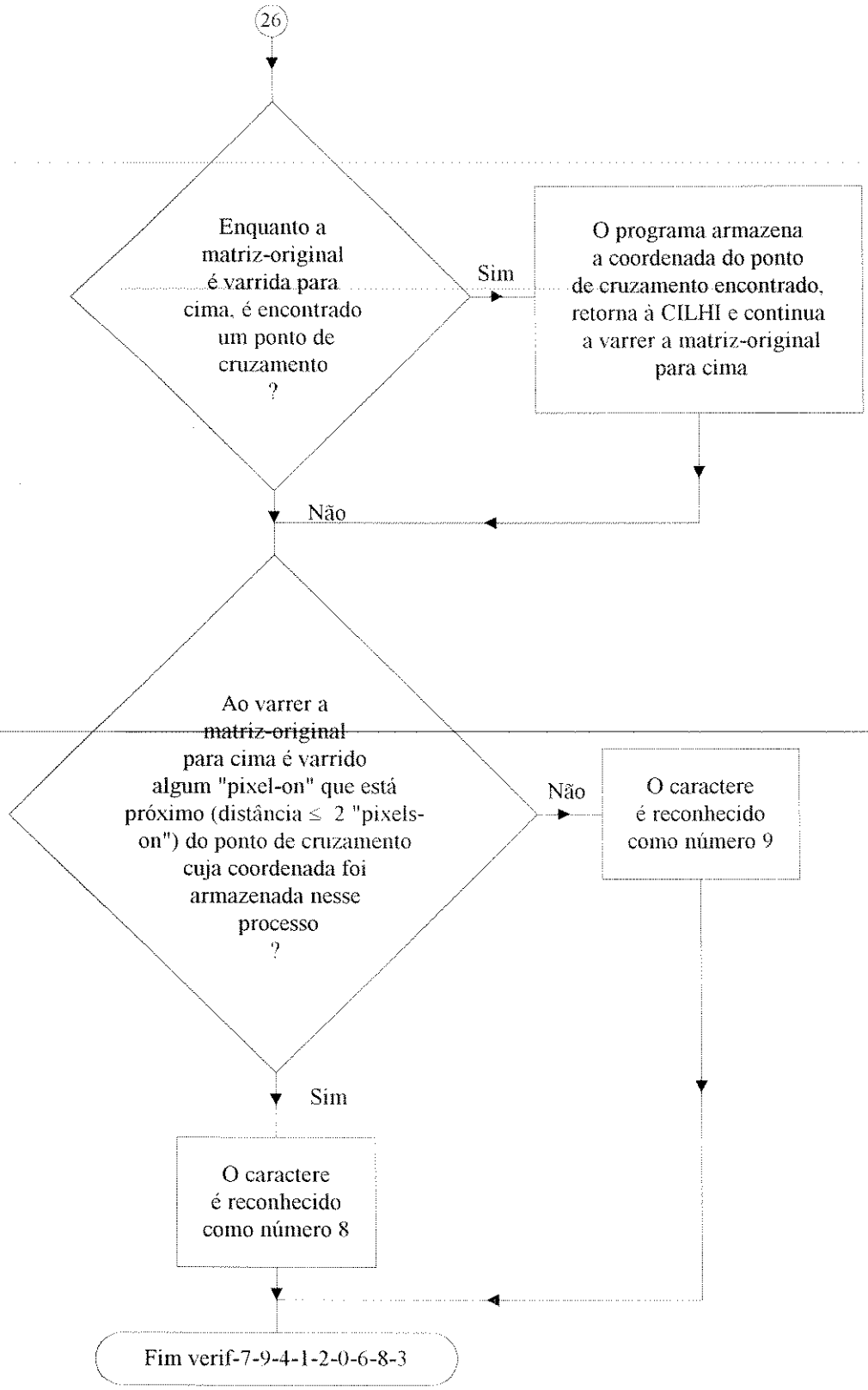


Figura 4.23(n) - Décima quarta parte do fluxograma da função verif-7-9-4-1-2-0-6-8-3.



LHI - linha horizontal inferior
CILHI - coordenada inicial da LHI

Figura 4.23(o) - Décima quinta parte do fluxograma da função verif-7-9-4-1-2-0-6-8-3.

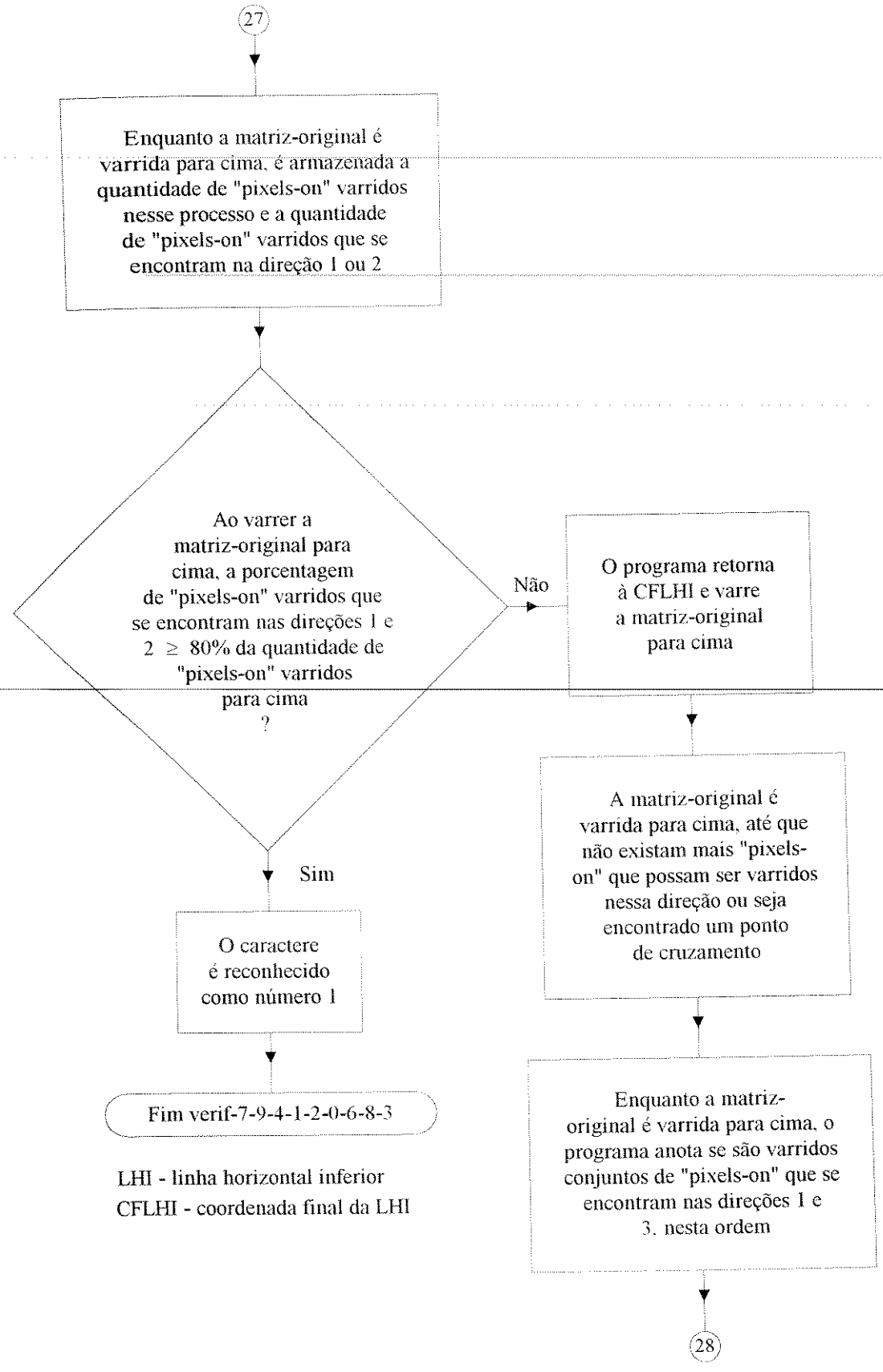


Figura 4.23(p) - Décima sexta parte do fluxograma da função verif-7-9-4-1-2-0-6-8-3.

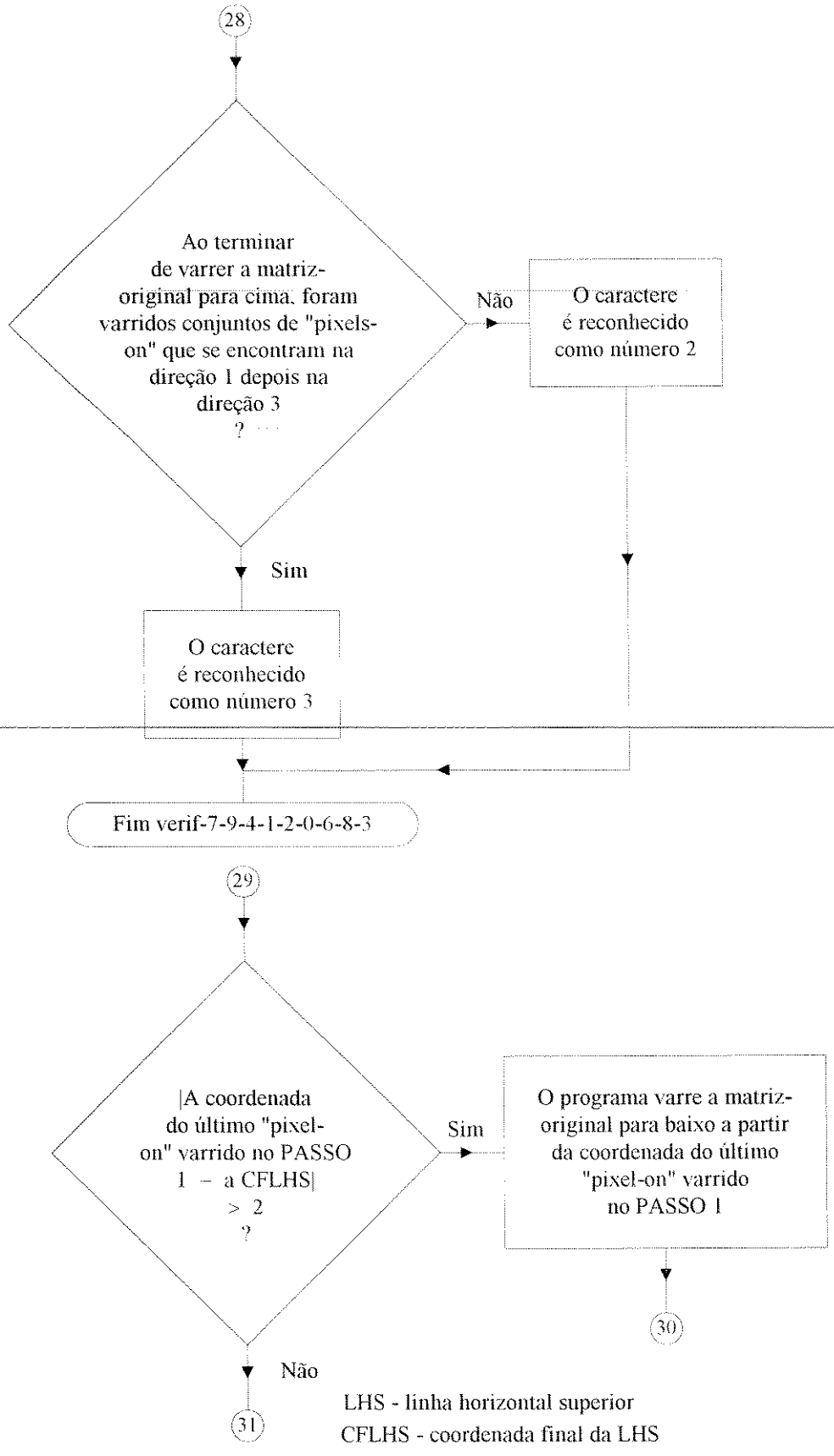


Figura 4.23(q) - Décima sétima parte do fluxograma da função verif-7-9-4-1-2-0-6-8-3.

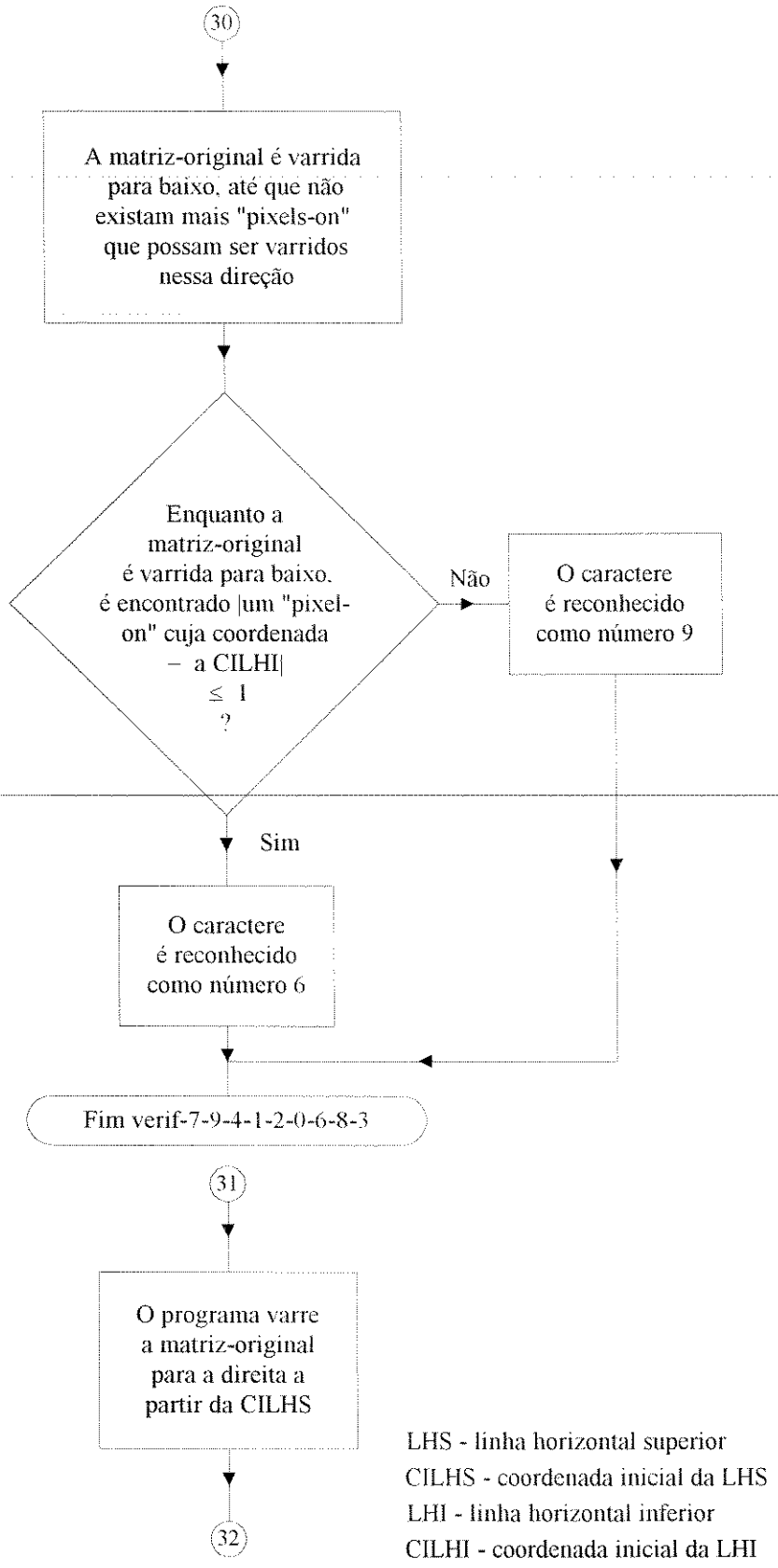


Figura 4.23(r) - Décima oitava parte do fluxograma da função verif-7-9-4-1-2-0-6-8-3.

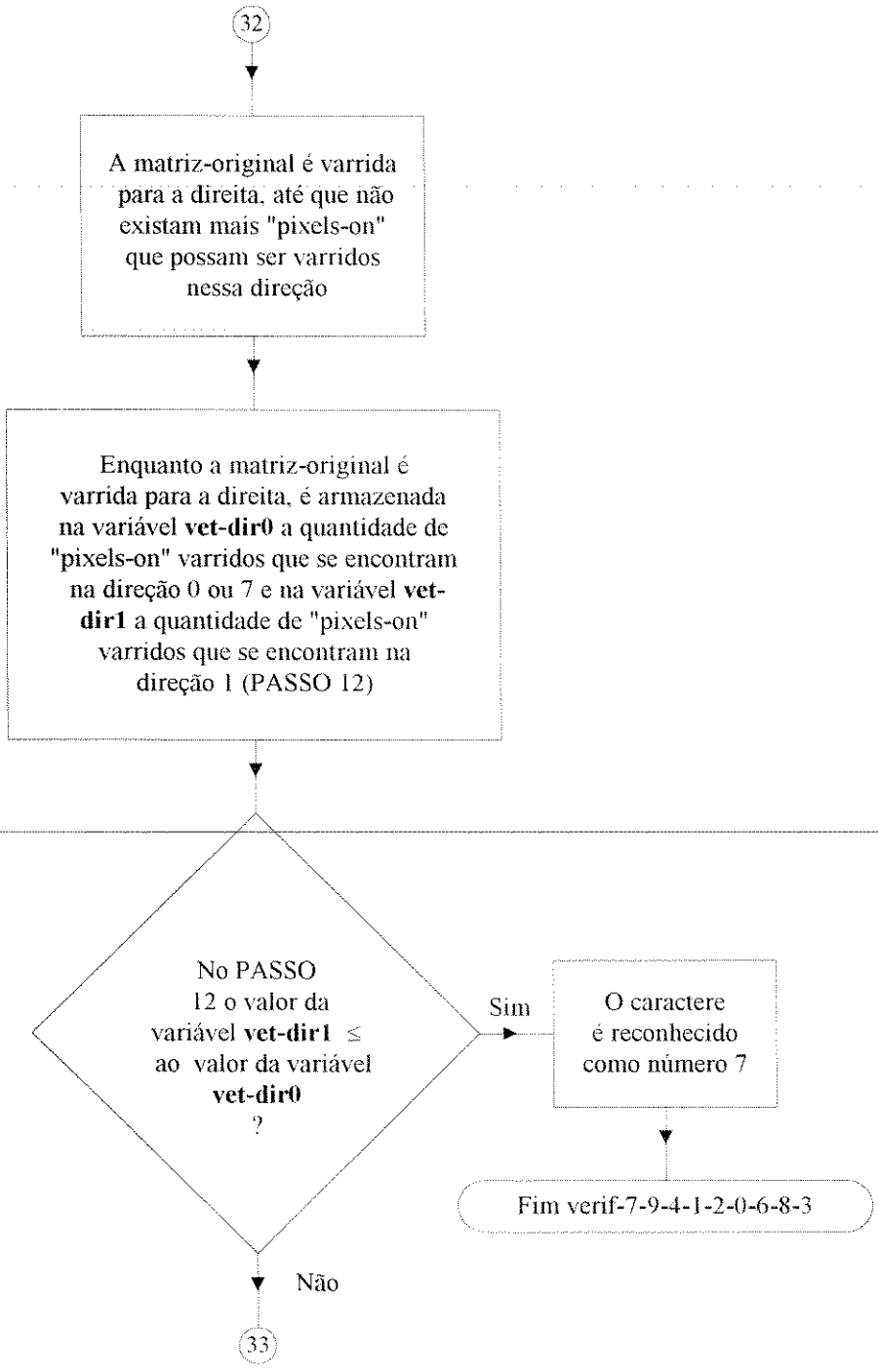


Figura 4.23(s) - Décima nona parte do fluxograma da função verif-7-9-4-1-2-0-6-8-3.

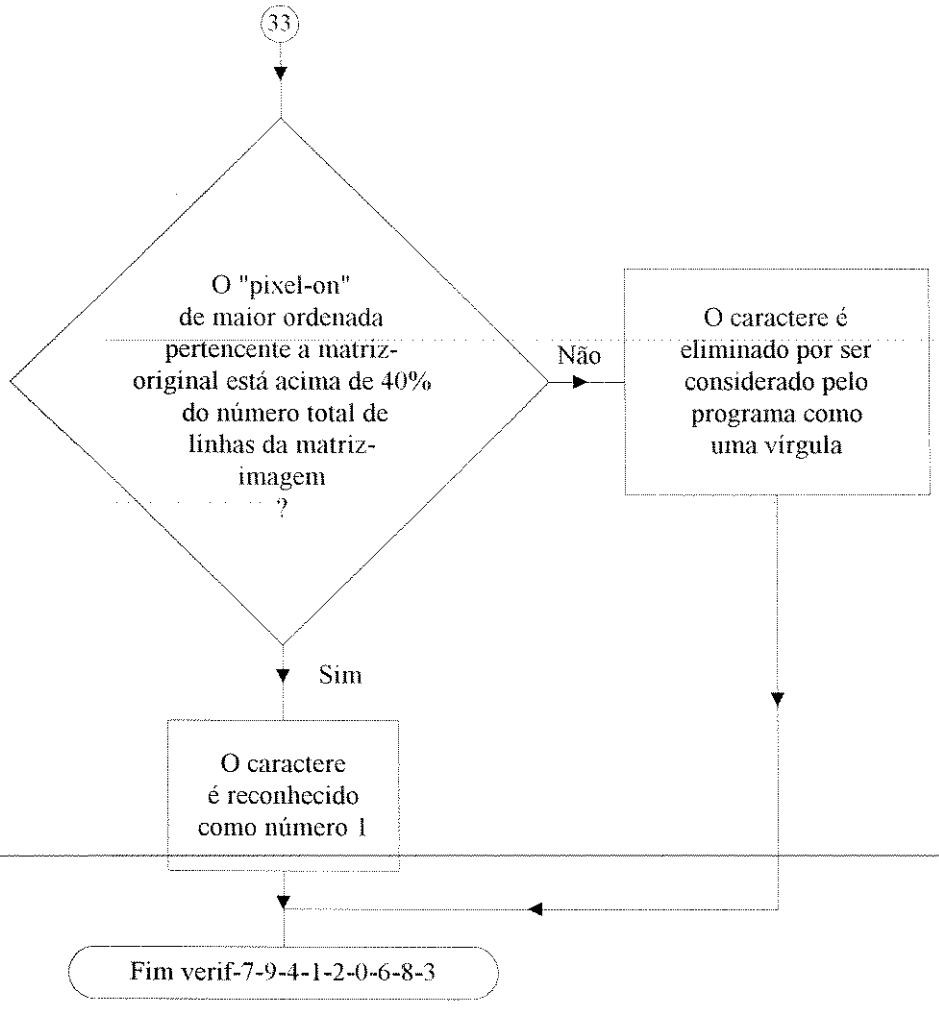


Figura 4.23(t) - Vigésima parte do fluxograma da função verif-7-9-4-1-2-0-6-8-3.

Fluxograma da função verif-1-4-5-6-7

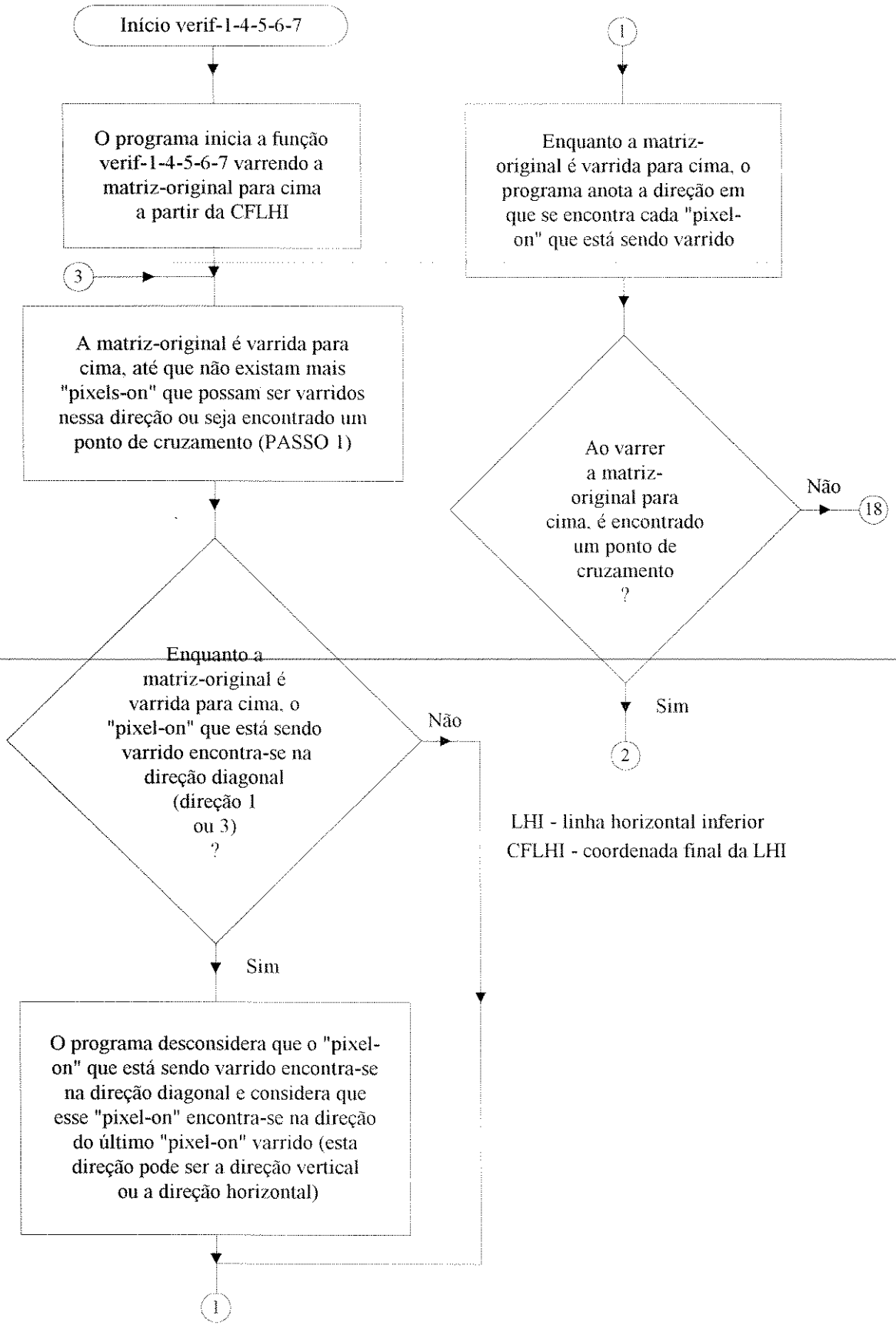


Figura 4.25(a) - Primeira parte do fluxograma da função verif-1-4-5-6-7.

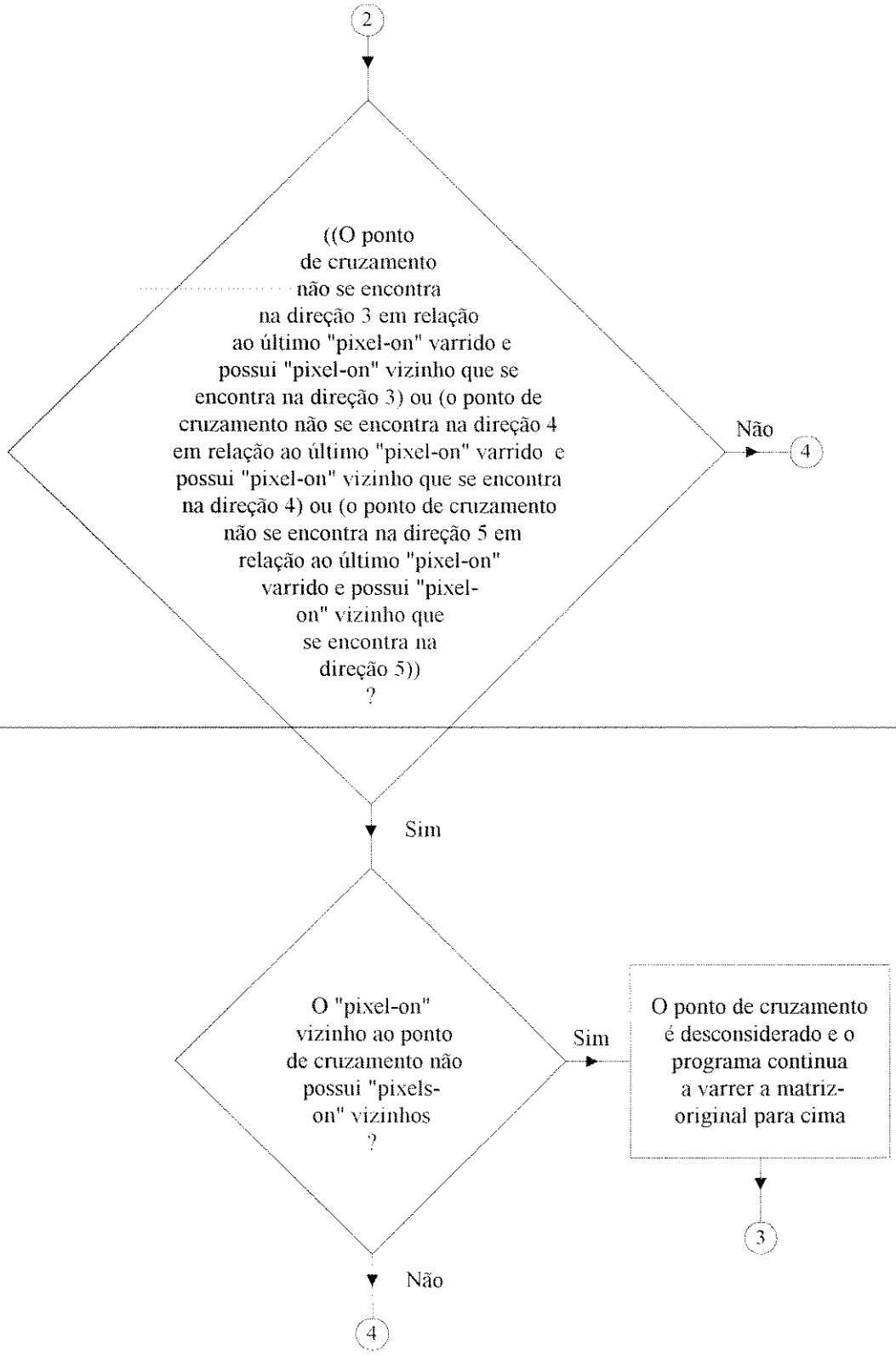


Figura 4.25(b) - Segunda parte do fluxograma da função verif-1-4-5-6-7.

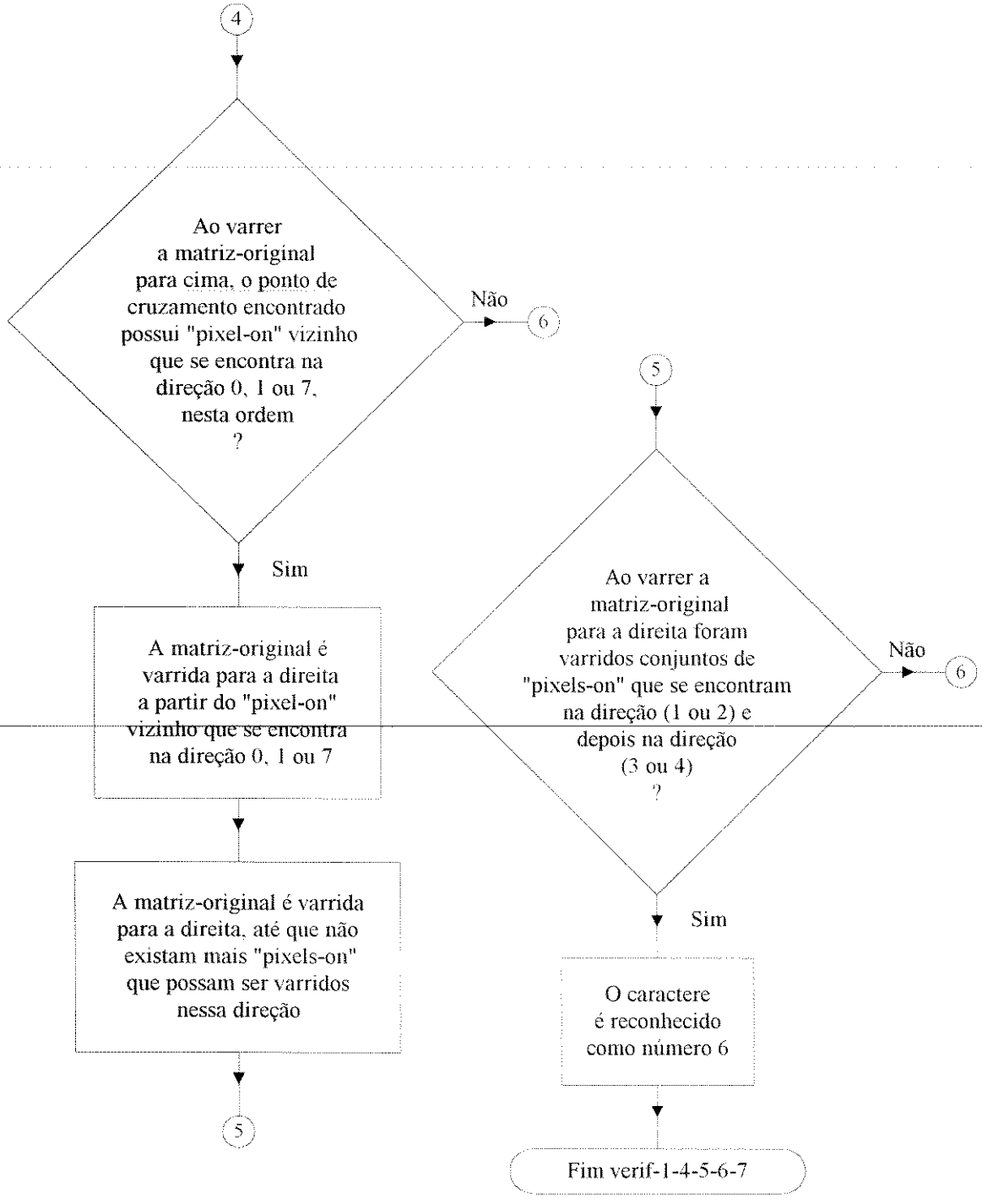


Figura 4.25(c) - Terceira parte do fluxograma da função verif-1-4-5-6-7.

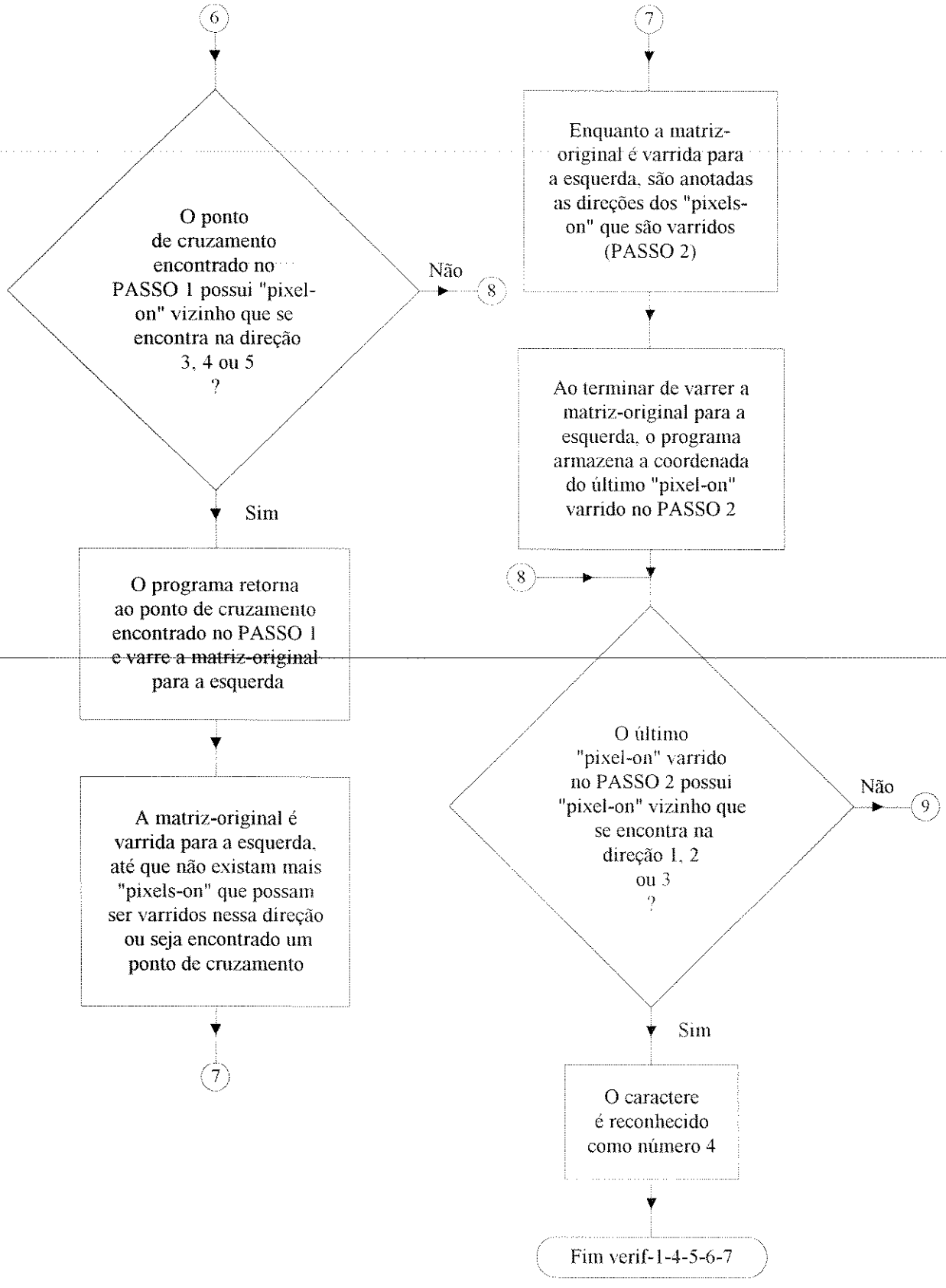


Figura 4.25(d) - Quarta parte do fluxograma da função verif-1-4-5-6-7.

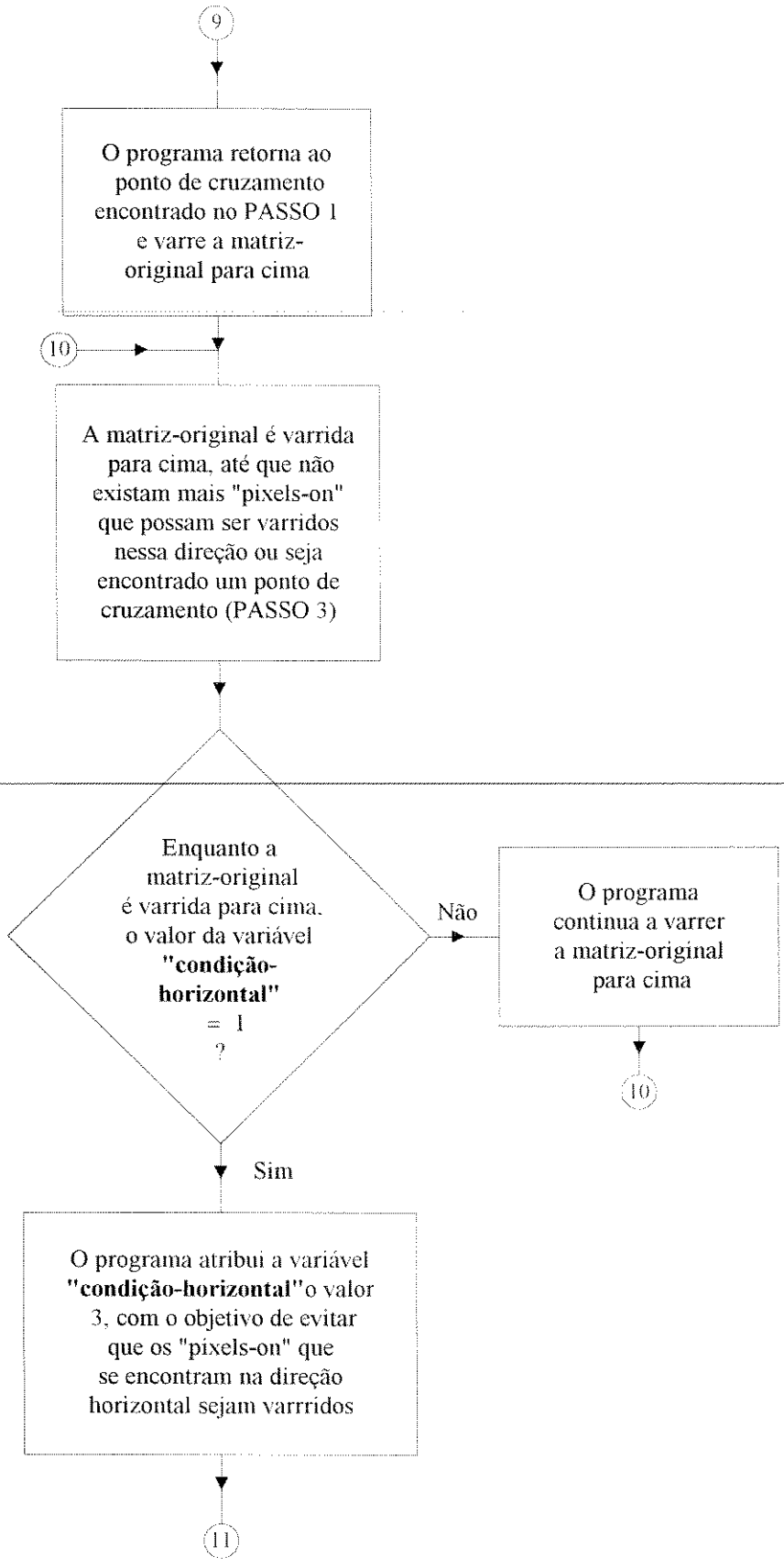


Figura 4.25(e) - Quinta parte do fluxograma da função verif-1-4-5-6-7.

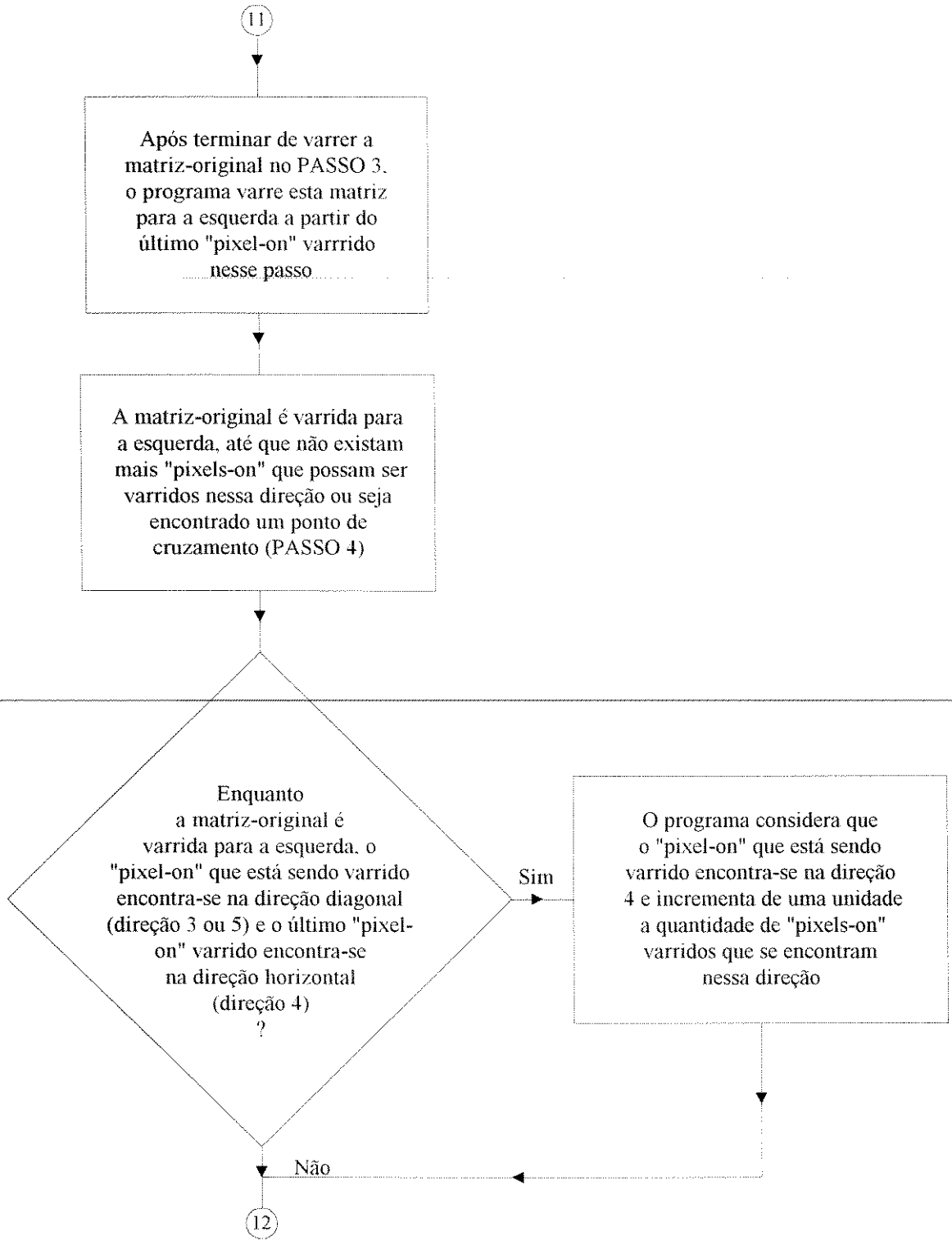


Figura 4.25(f) - Sexta parte do fluxograma da função verif-1-4-5-6-7.

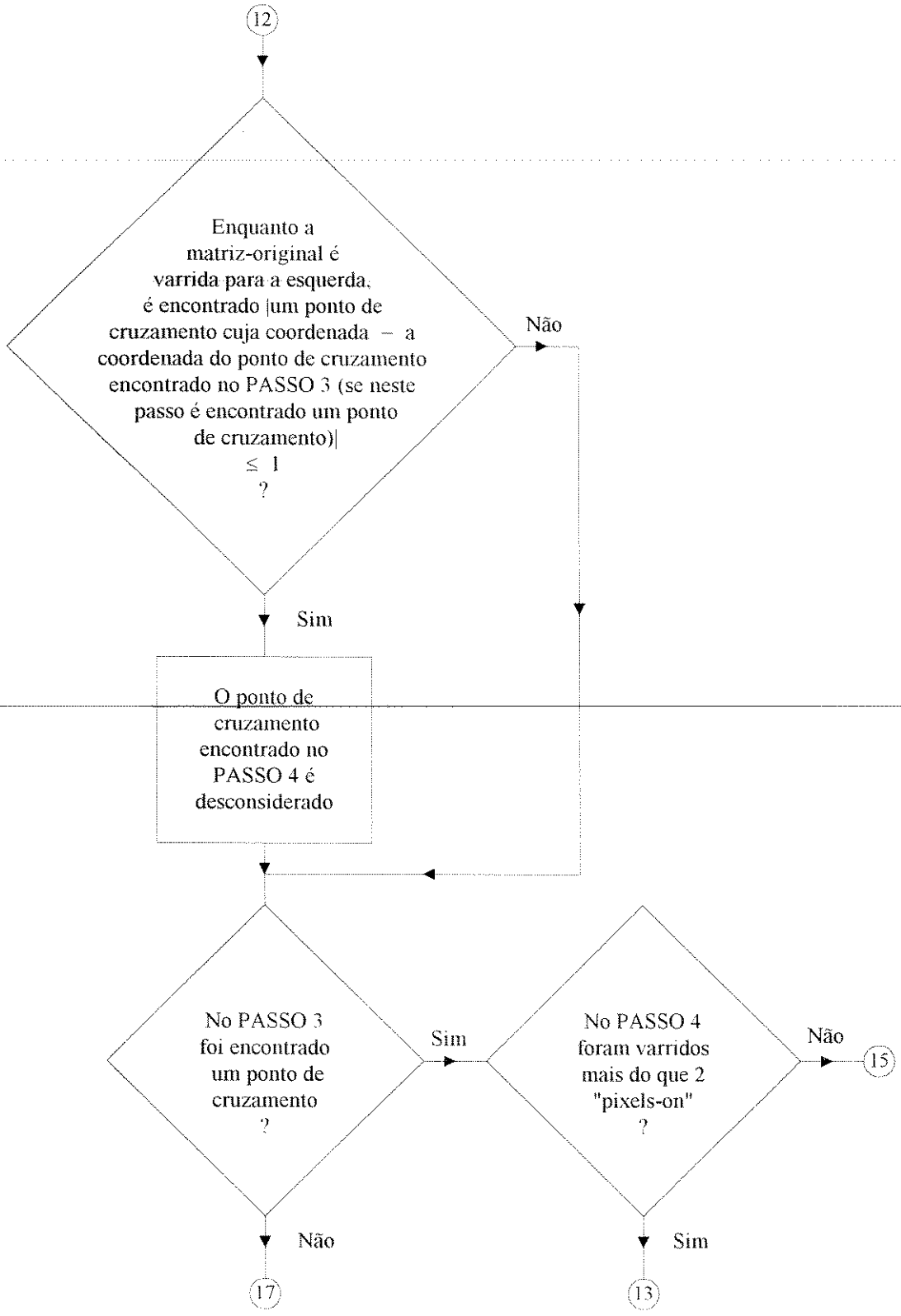


Figura 4.25(g) - Sétima parte do fluxograma da função verif-1-4-5-6-7.

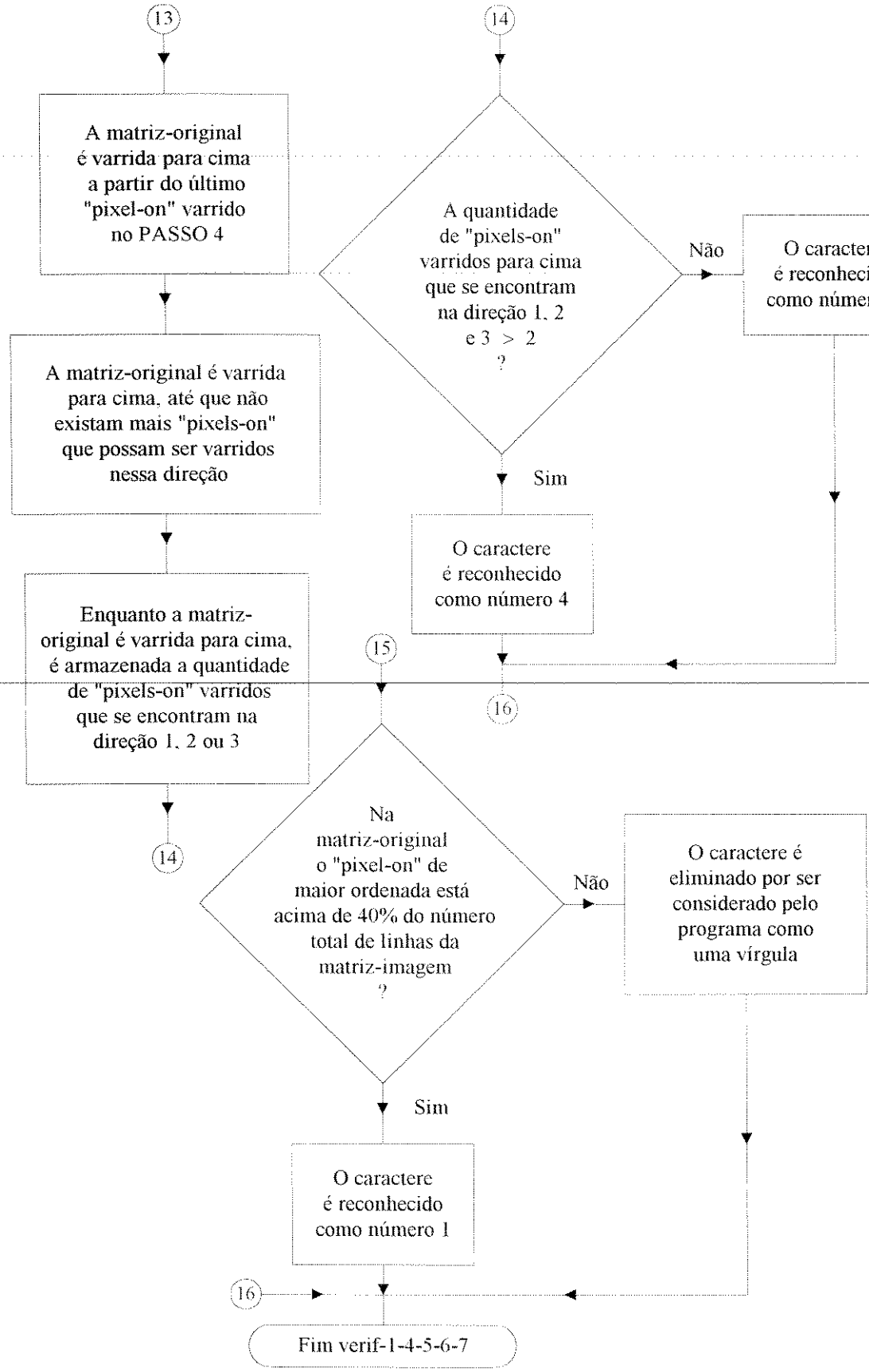


Figura 4.25(h) - Oitava parte do fluxograma da função verif-1-4-5-6-7.

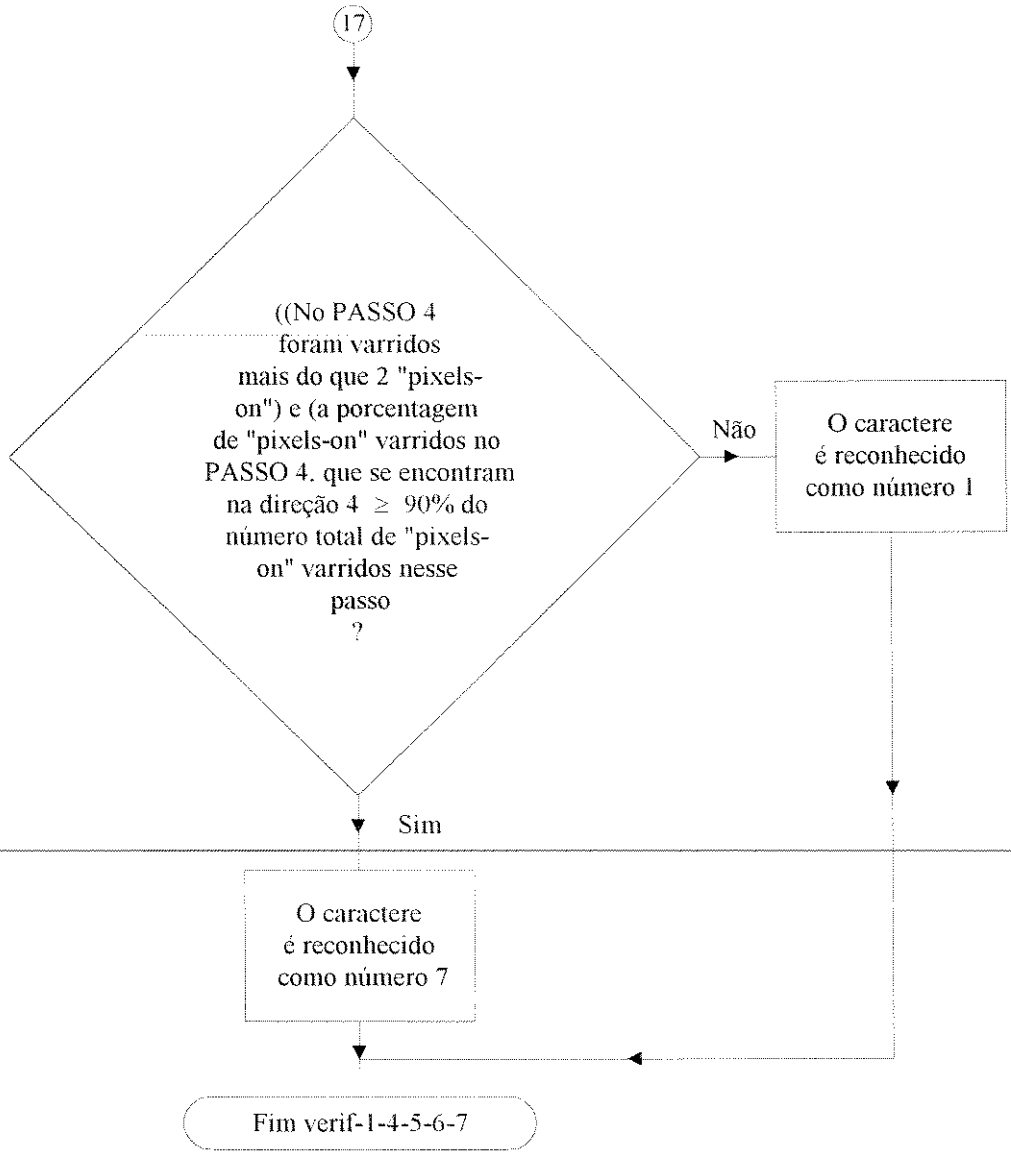


Figura 4.25(i) - Nona parte do fluxograma da função verif-1-4-5-6-7.

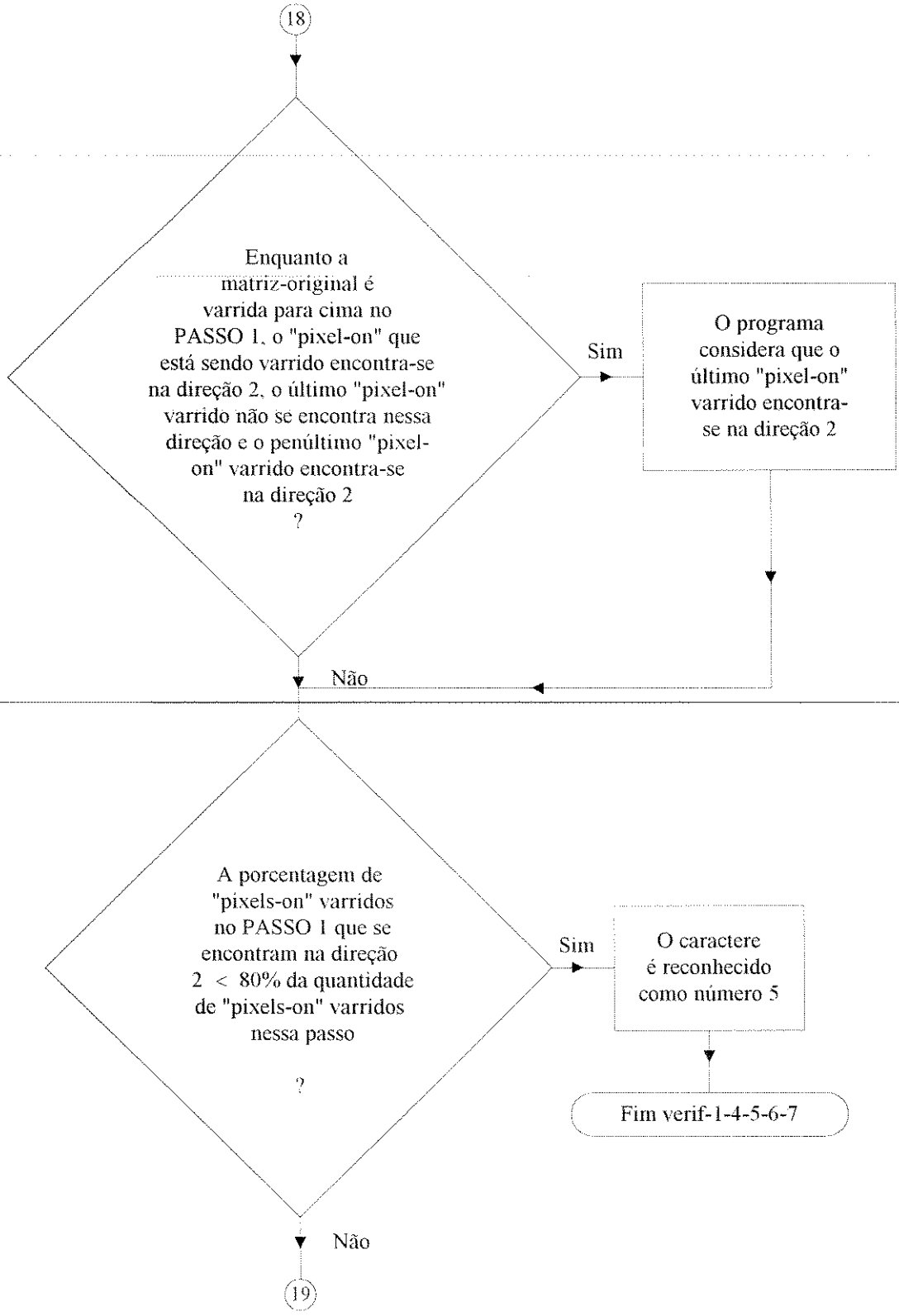


Figura 4.25(j) - Décima parte do fluxograma da função verif-1-4-5-6-7.

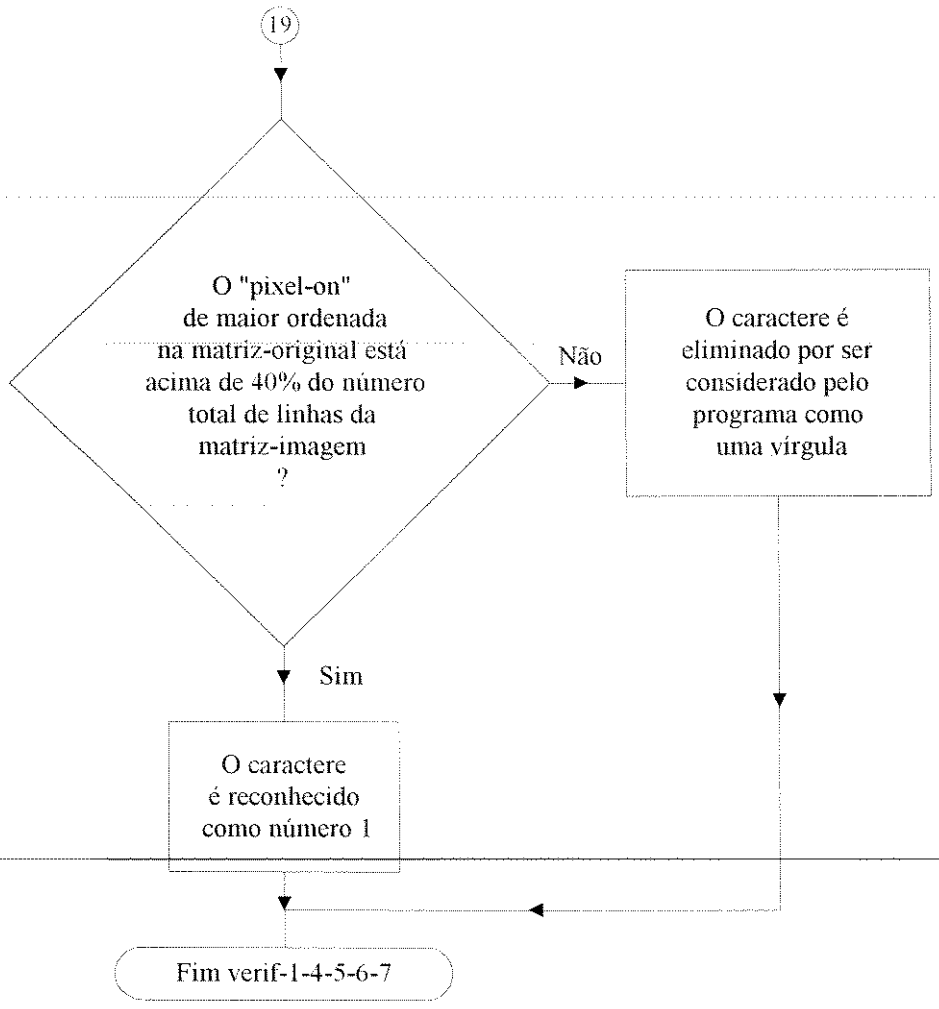


Figura 4.25(k) - Décima primeira parte do fluxograma da função verif-1-4-5-6-7.

FLUXOGRAMAS DO CAPÍTULO 5

Fluxograma da função virg-1

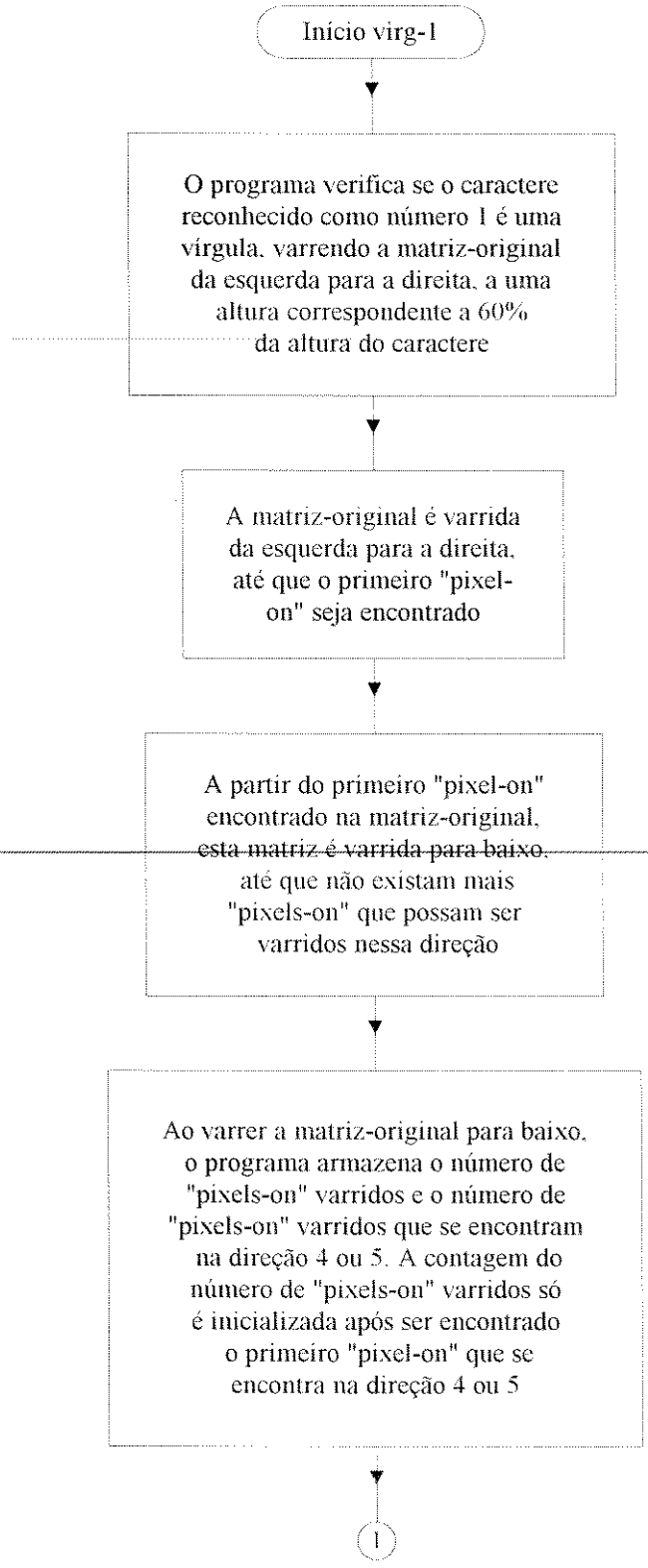


Figura 5.1(a) - Primeira parte do fluxograma da função virg-1.

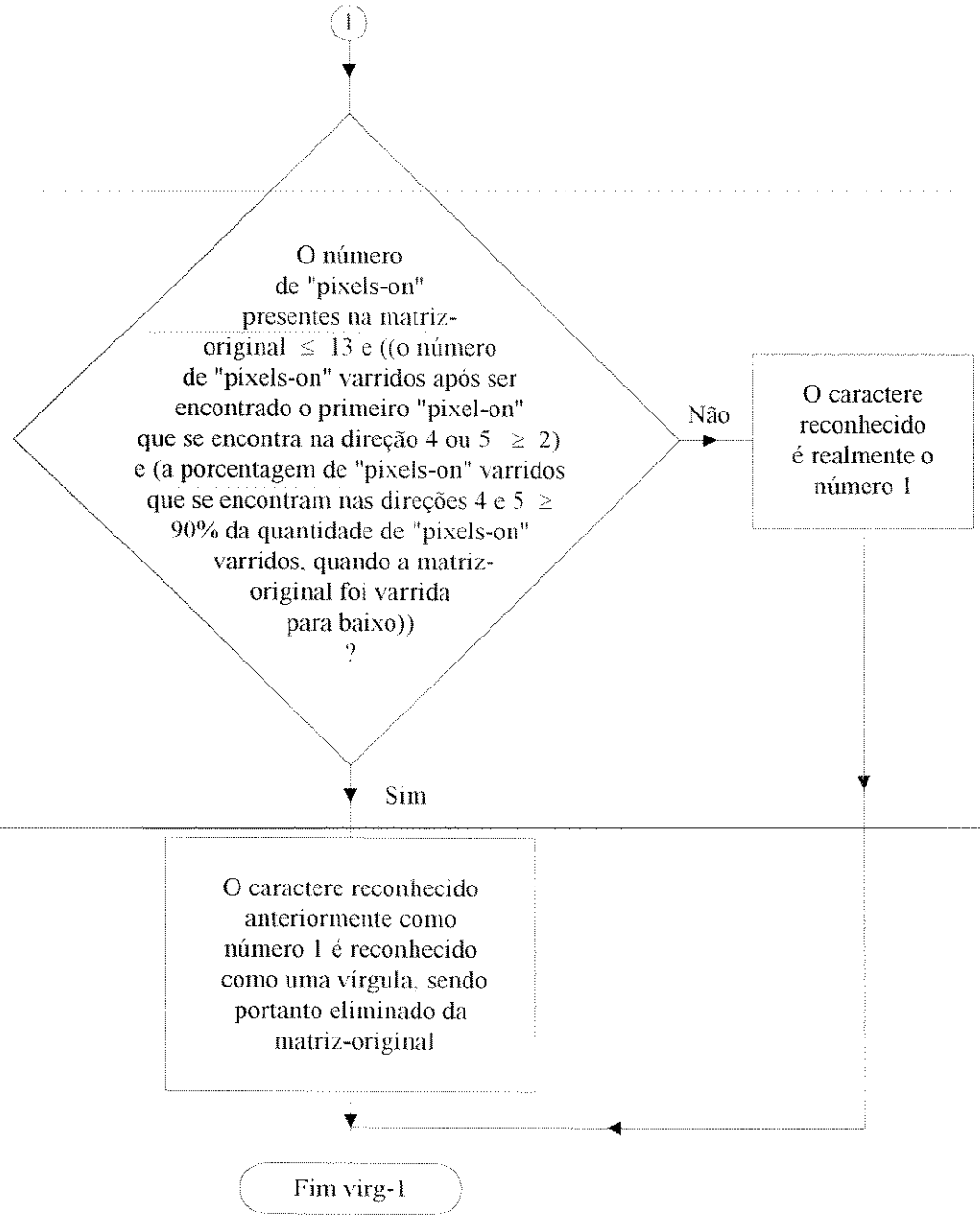


Figura 5.1(b) - Segunda parte do fluxograma da função virg-1.

Fluxograma da função elim-virg

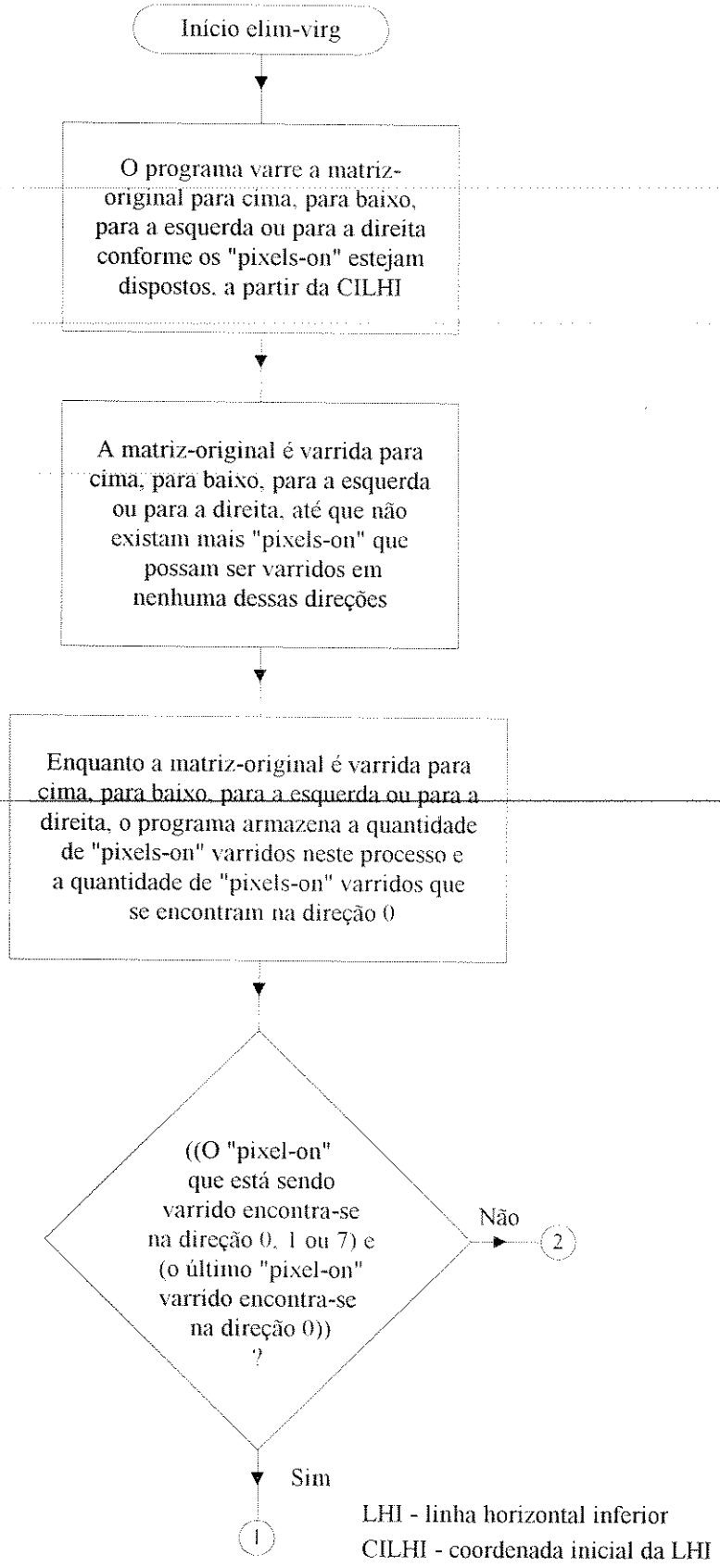


Figura 5.2(a) - Primeira parte do fluxograma da função elim-virg.

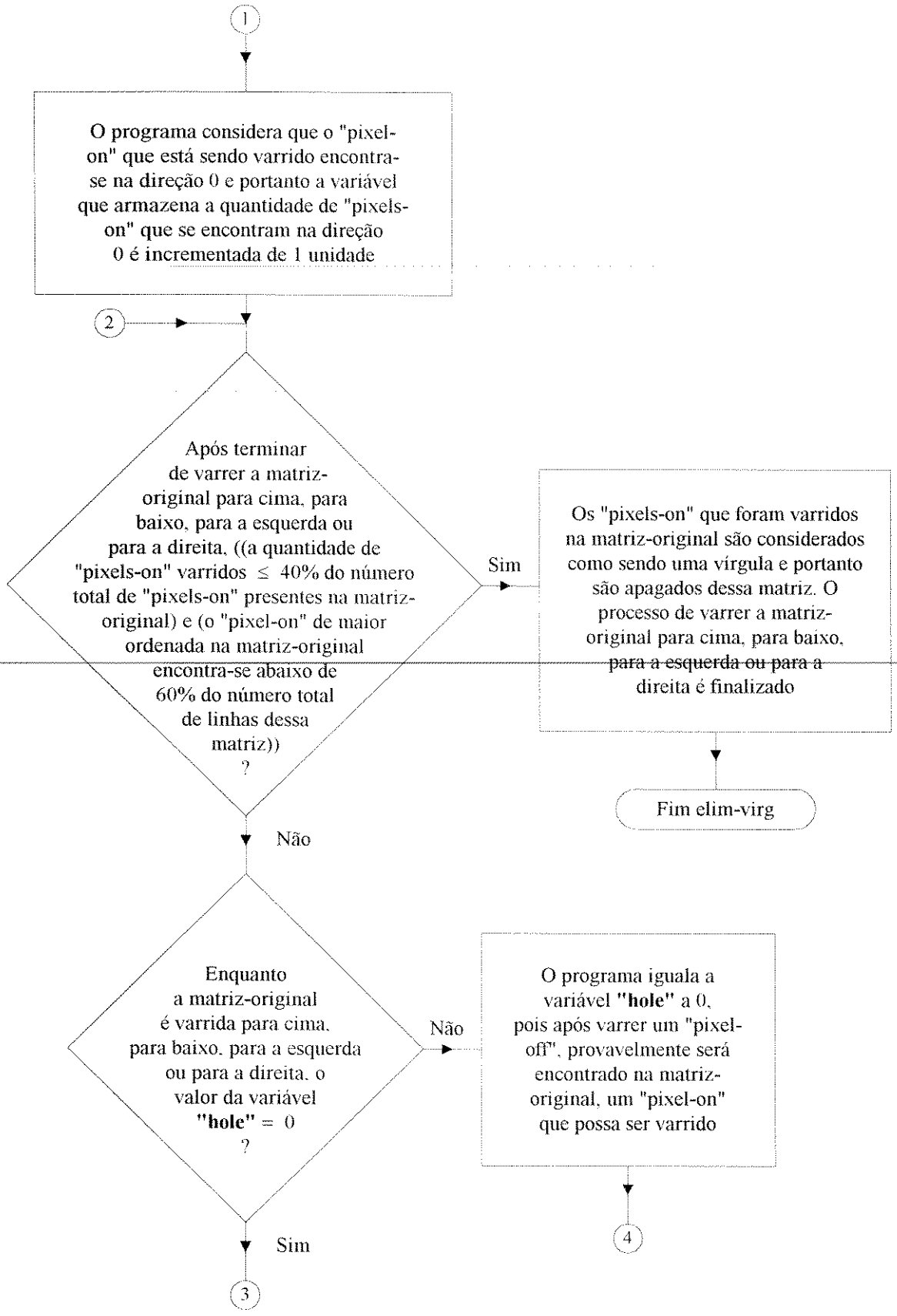


Figura 5.2(b) - Segunda parte do fluxograma da função elim-virg.

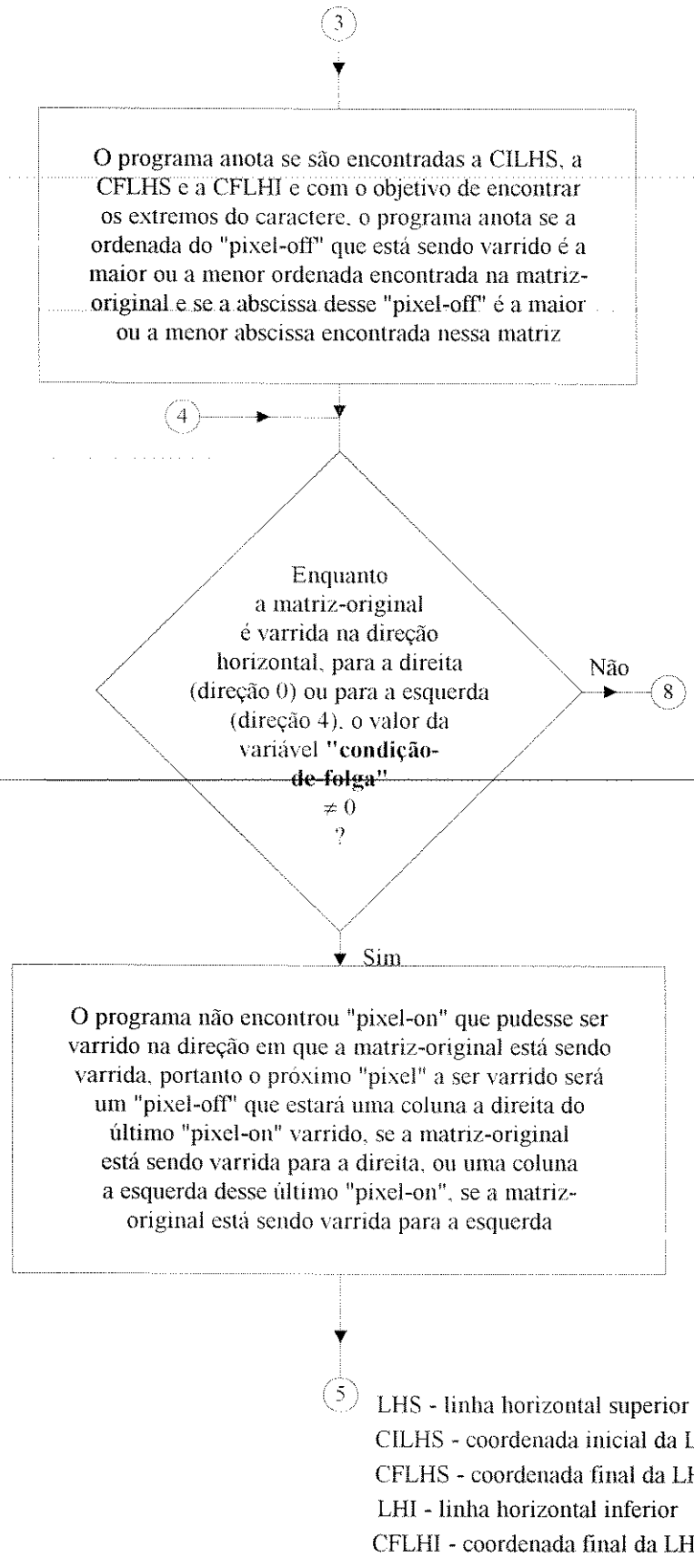


Figura 5.2(c) - Terceira parte do fluxograma da função elim-virg.

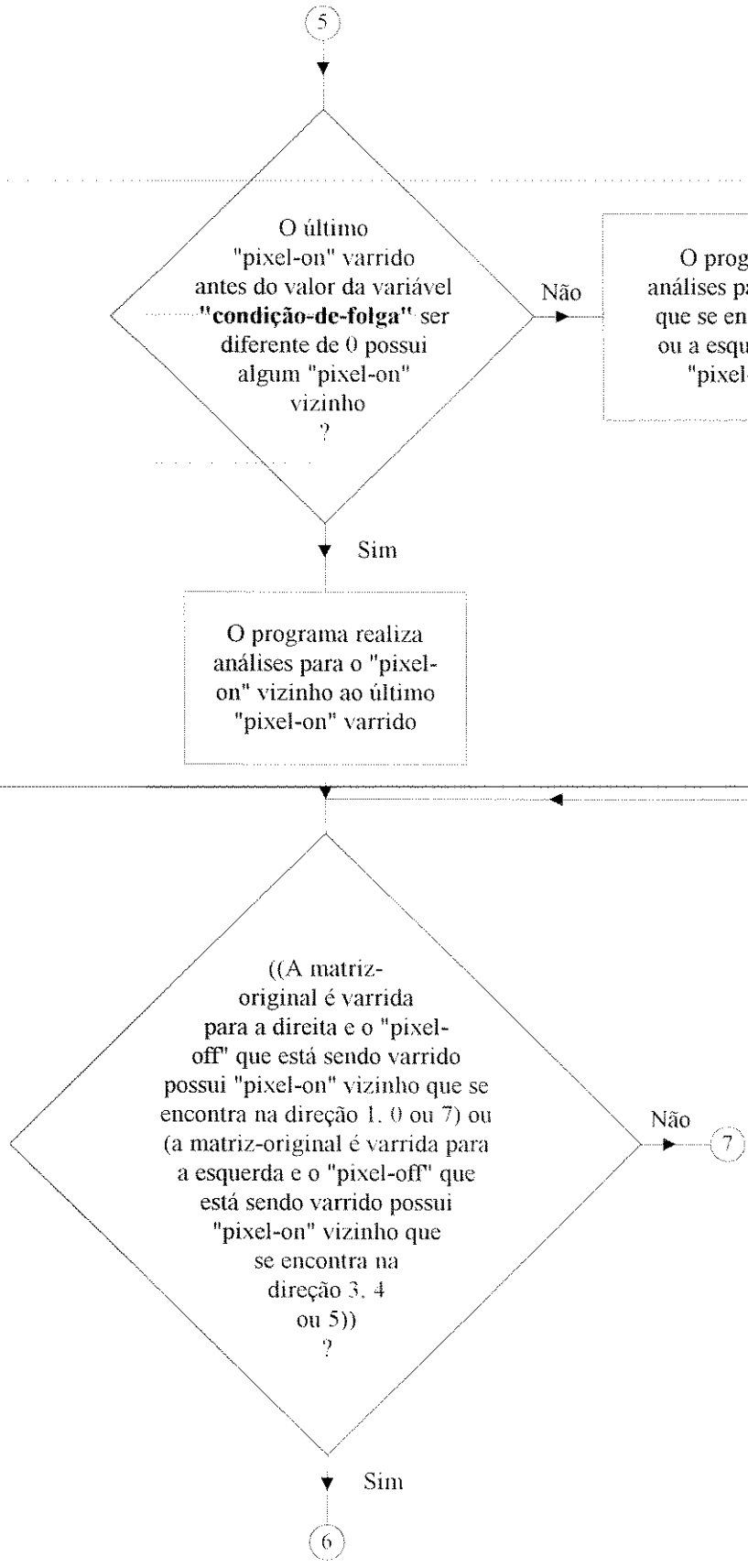


Figura 5.2(d) - Quarta parte do fluxograma da função elim-virg.

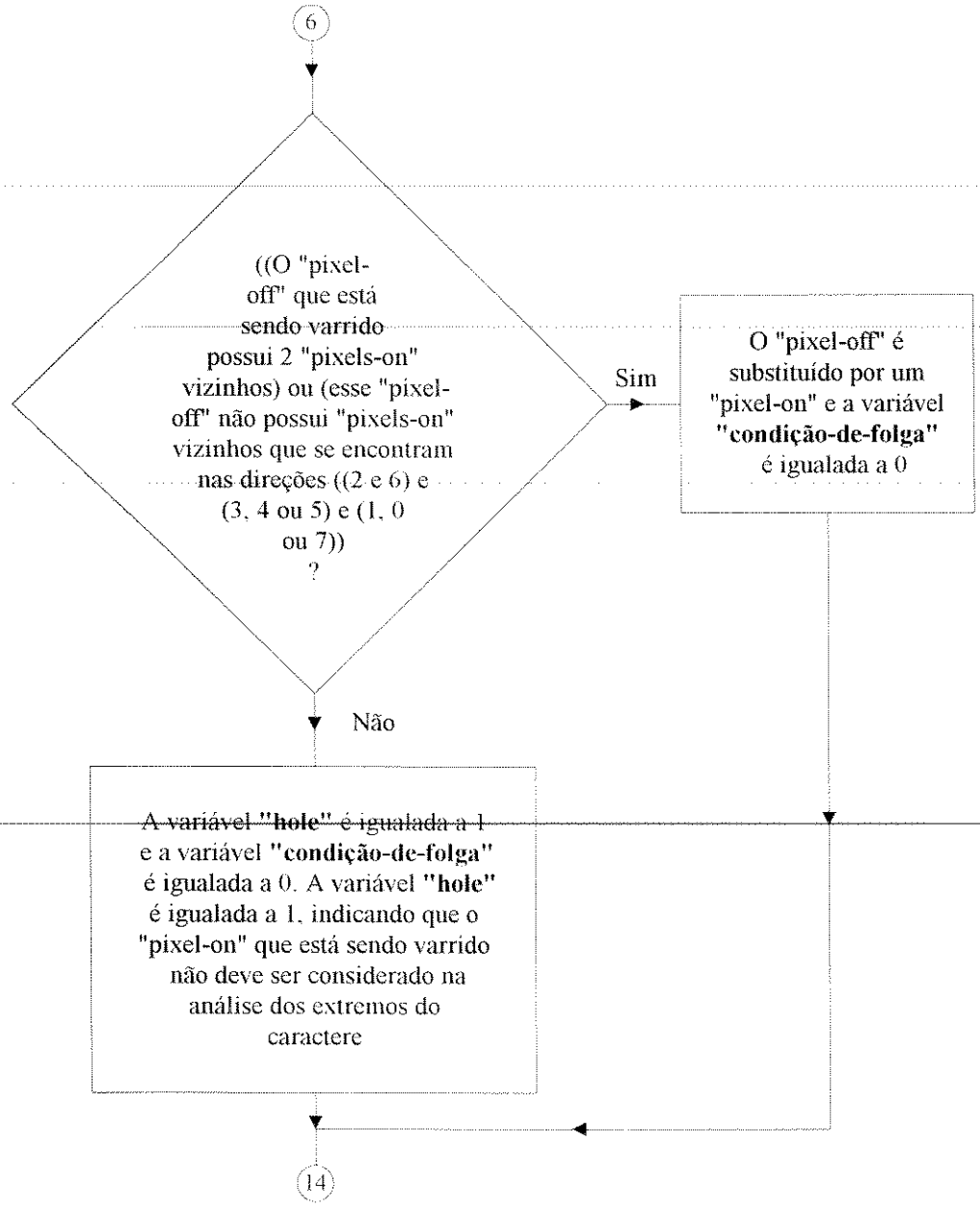


Figura 5.2(e) - Quinta parte do fluxograma da função elim-virg.

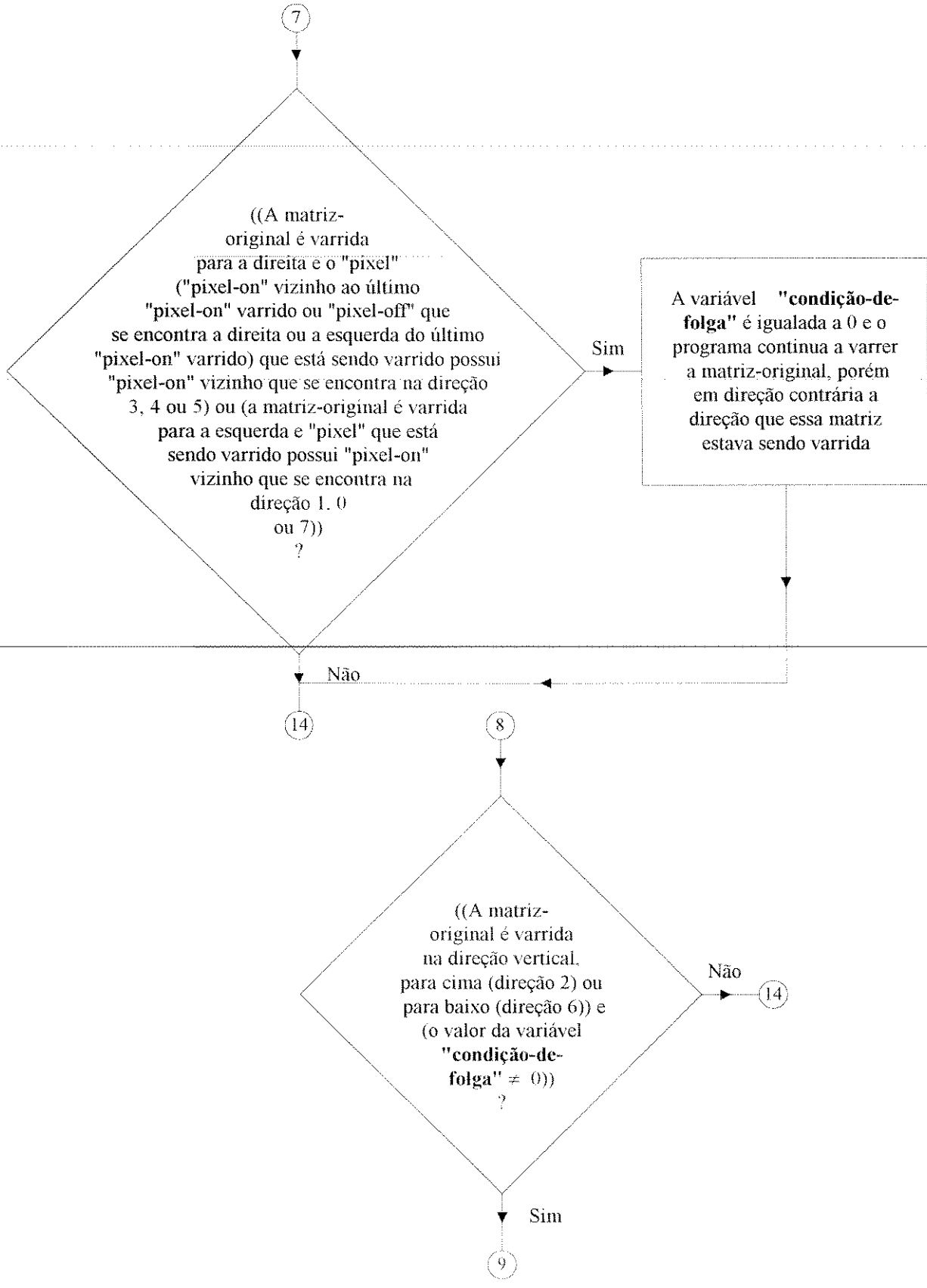


Figura 5.2(f) - Sexta parte do fluxograma da função elim-virg.

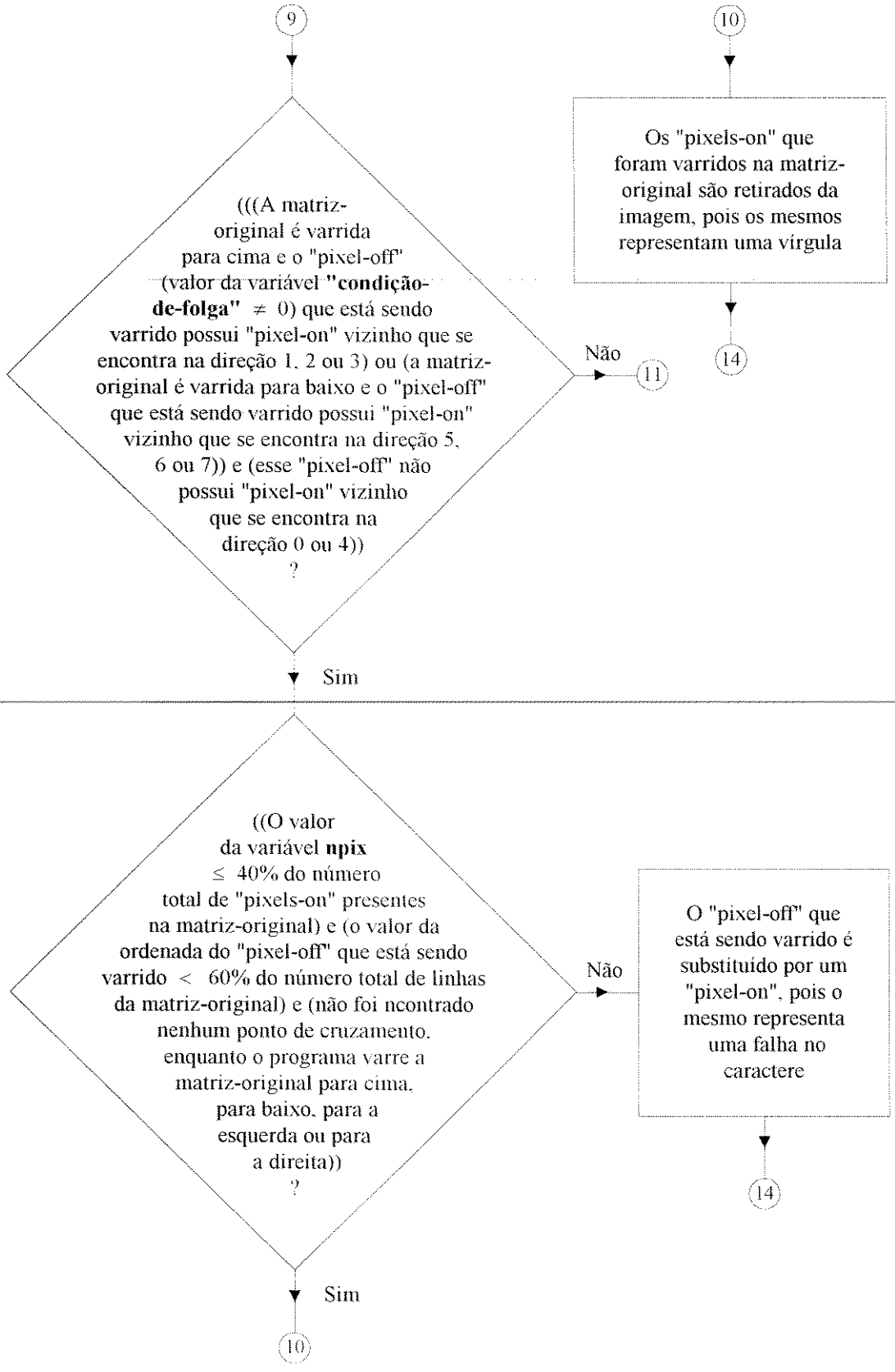


Figura 5.2(g) - Sétima parte do fluxograma da função elim-virg.

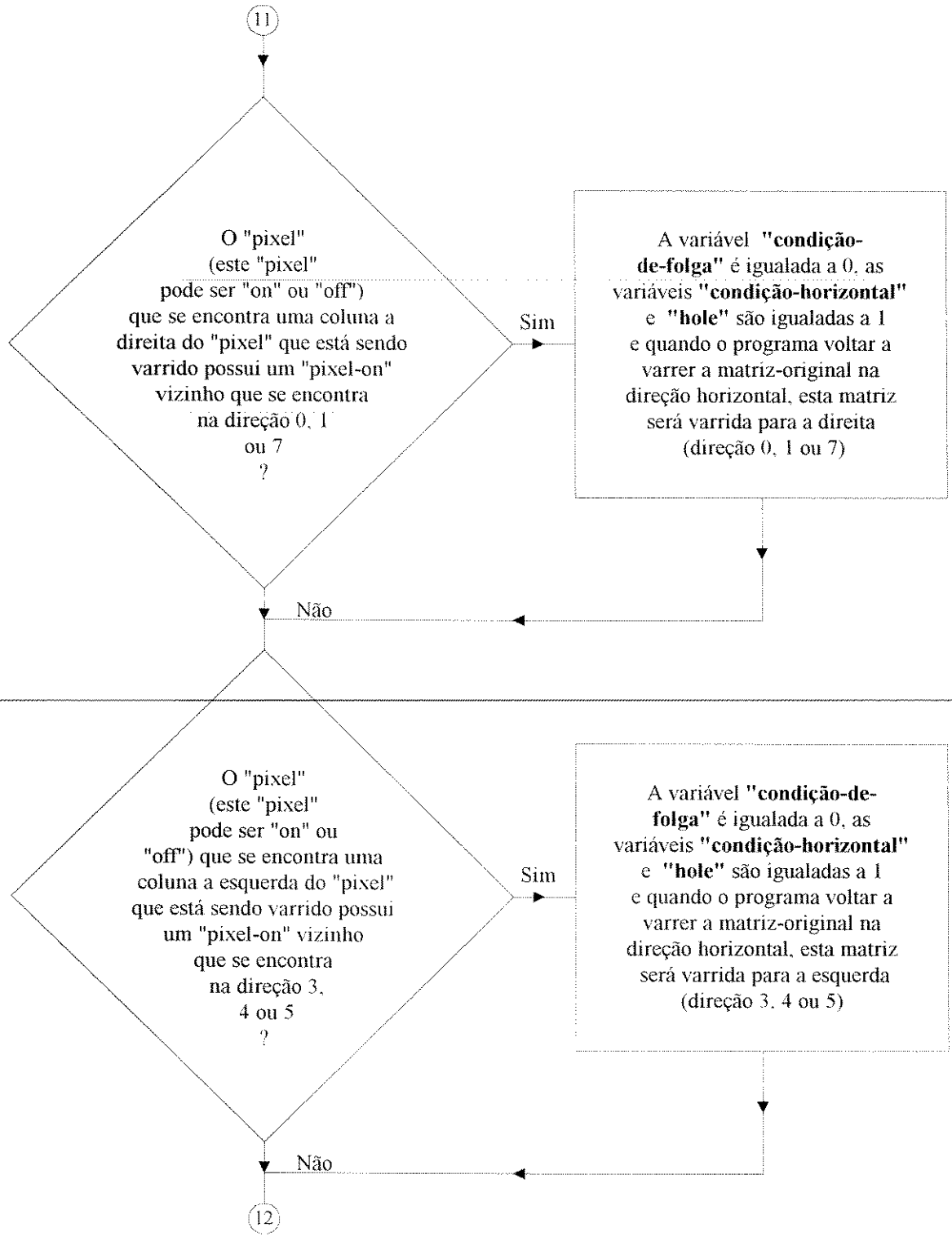


Figura 5.2(h) - Oitava parte do fluxograma da função elim-virg.

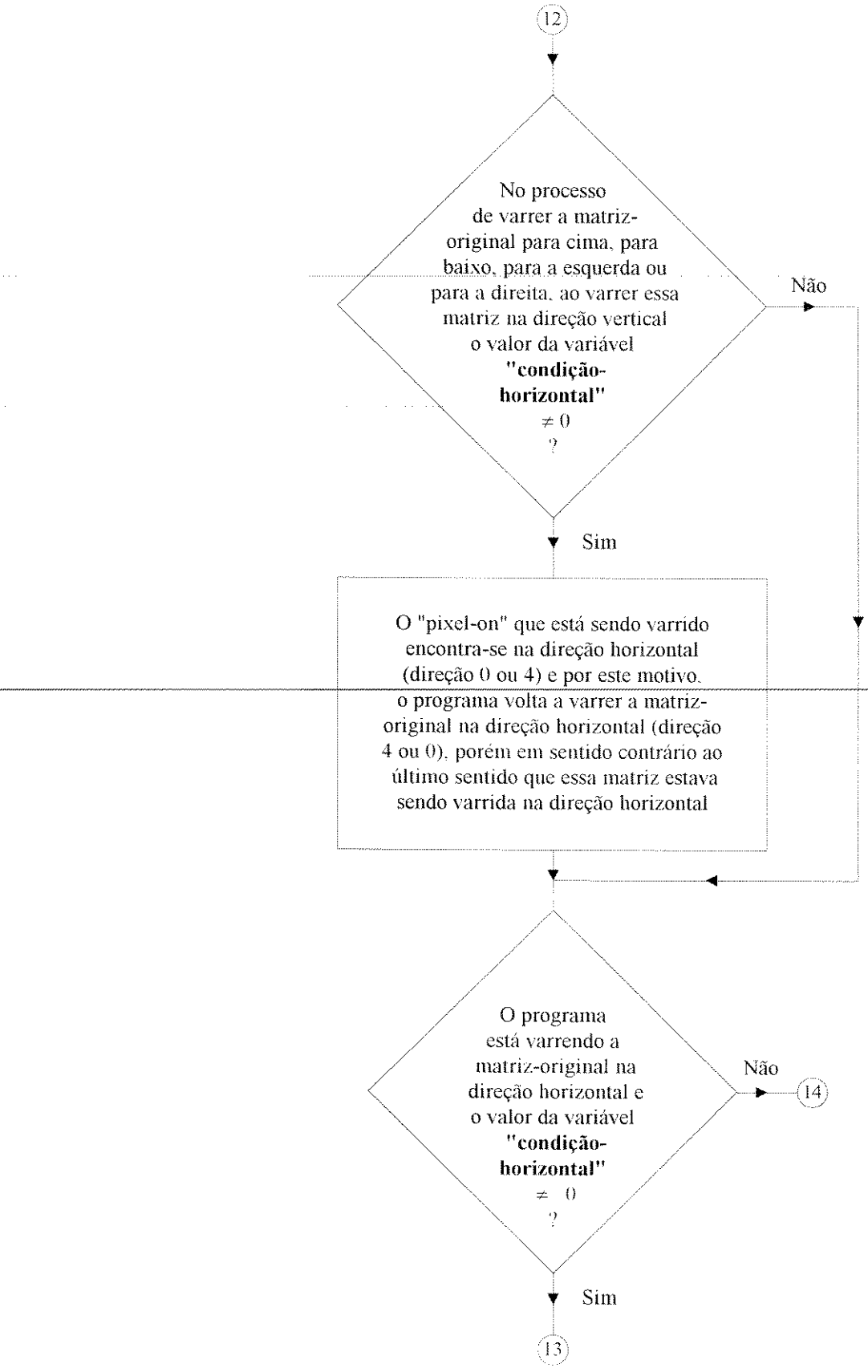


Figura 5.2(i) - Nona parte do fluxograma da função elim-virg.

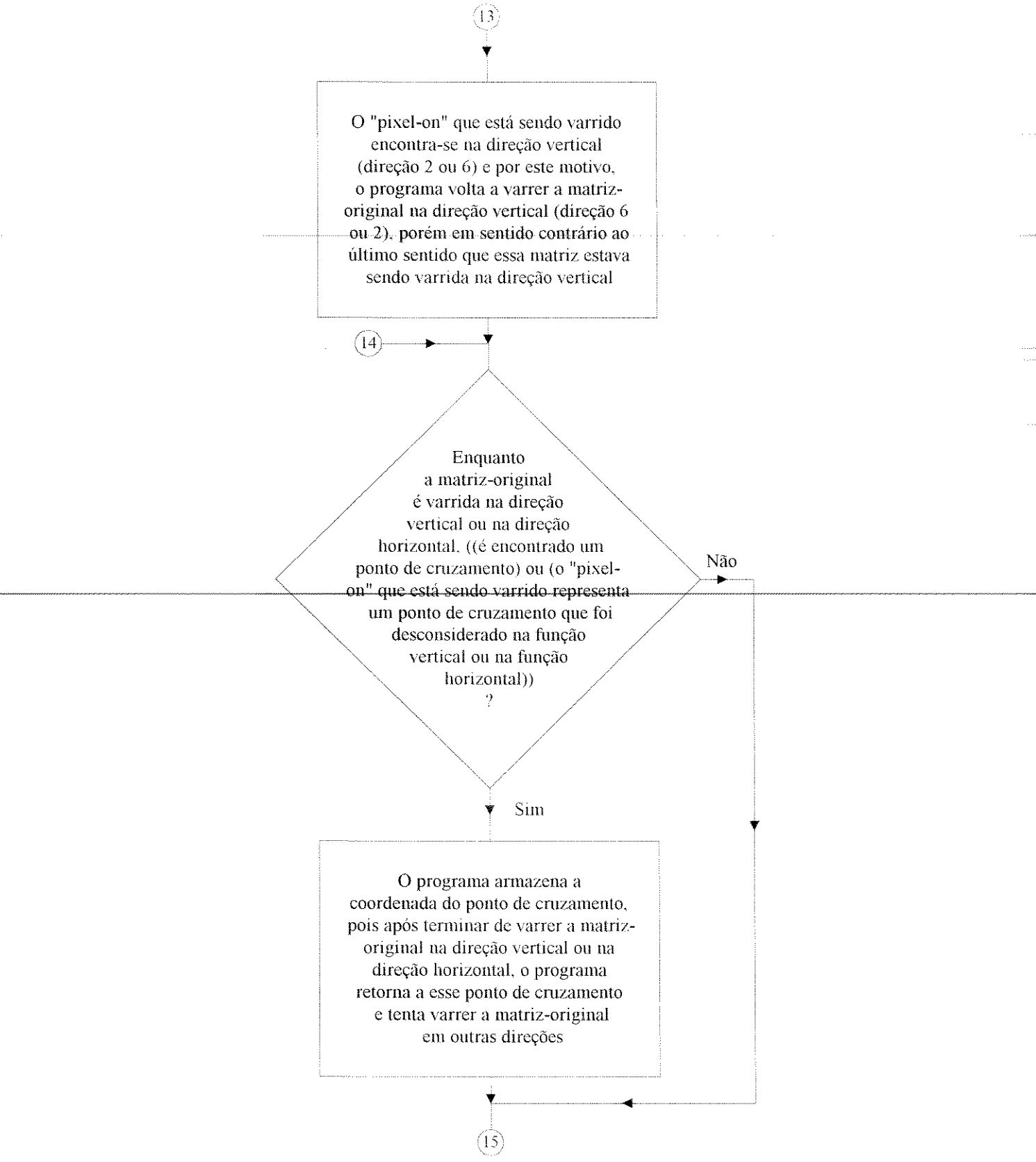


Figura 5.2(j) - Décima parte do fluxograma da função elim-virg.

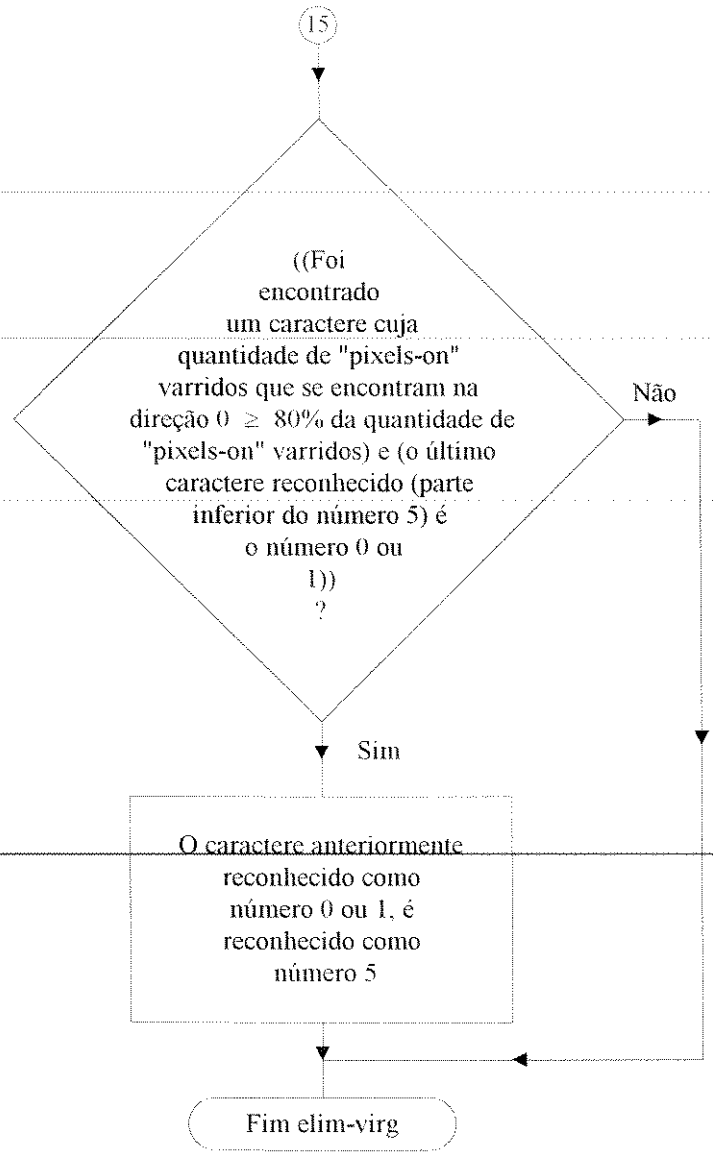


Figura 5.2(k) - Décima primeira parte do fluxograma da função elim-virg.

Fluxograma do processo de eliminação do ponto e/ou ruído

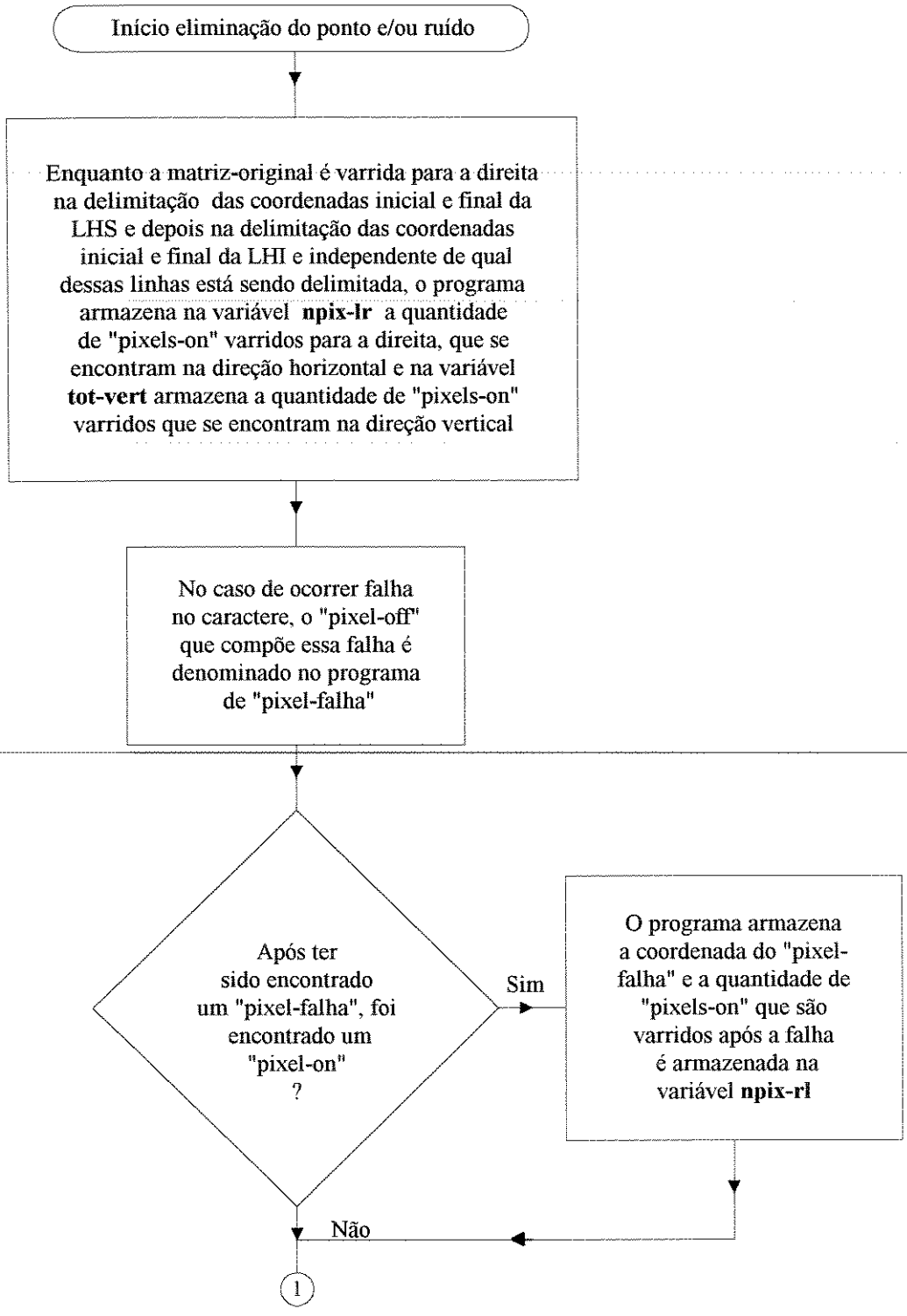


Figura 5.3(a) - Primeira parte do fluxograma do processo de eliminação do ponto e/ou ruído.

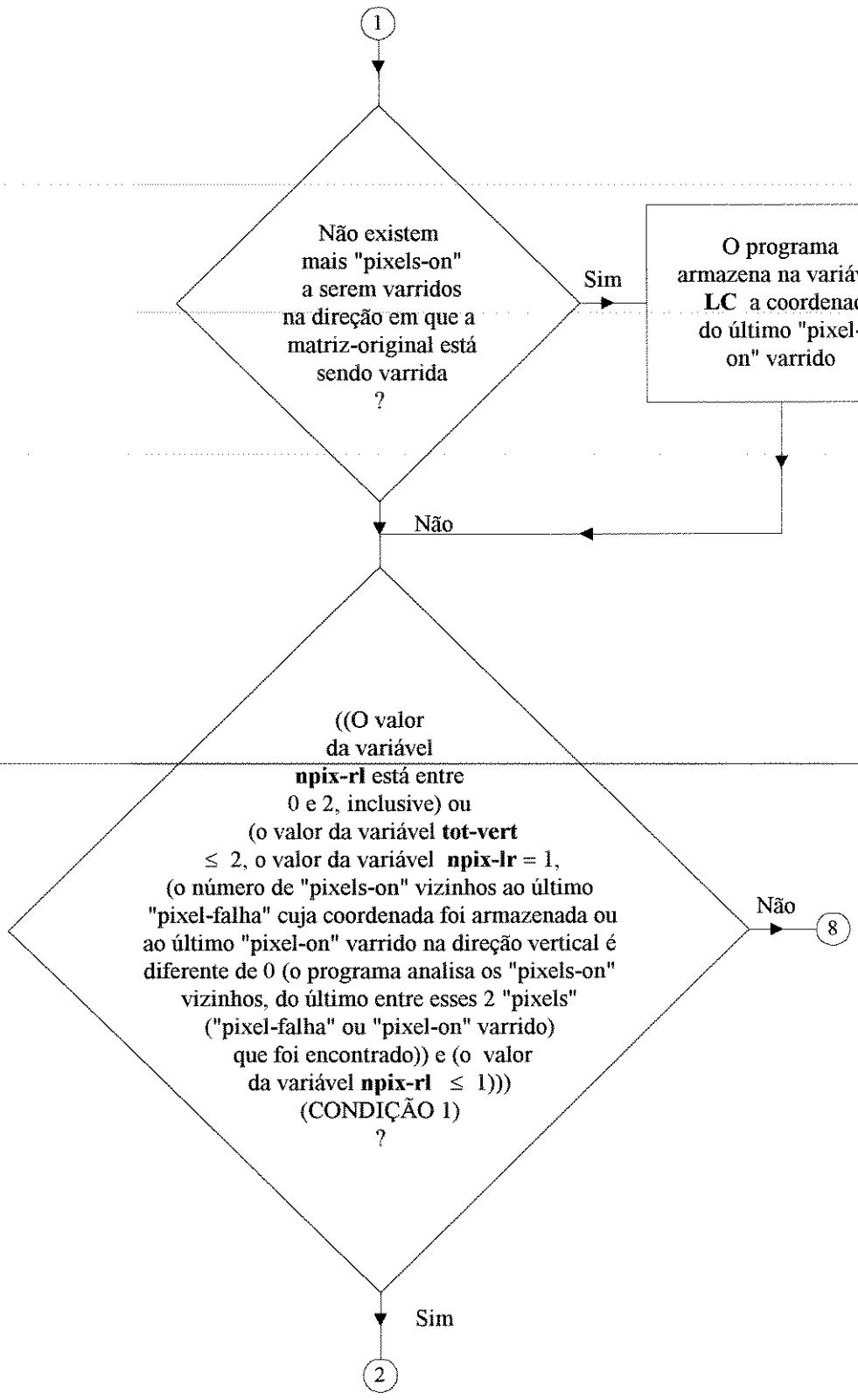


Figura 5.3(b) - Segunda parte do fluxograma do processo de eliminação do ponto e/ou ruído.

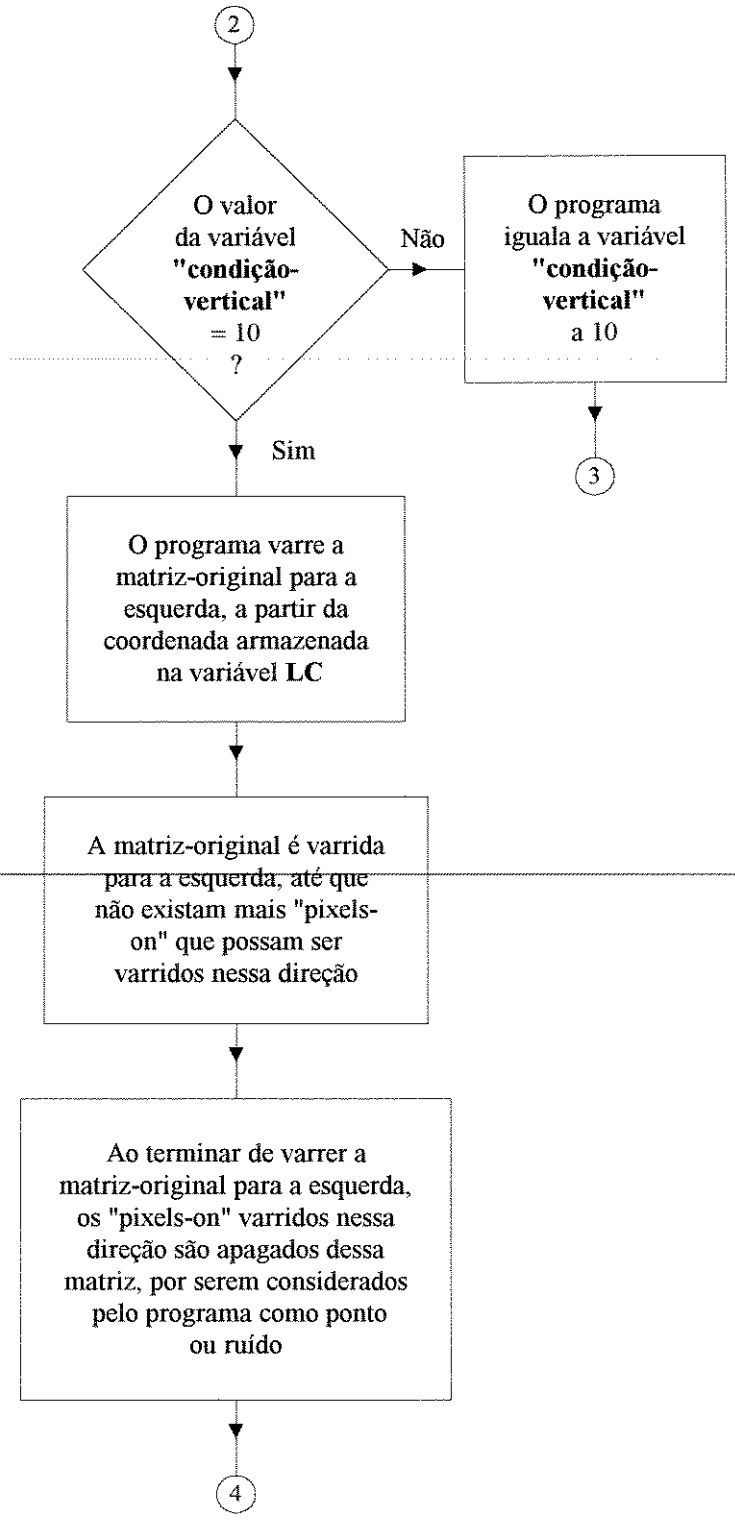


Figura 5.3(c) - Terceira parte do fluxograma do processo de eliminação do ponto e/ou ruído.

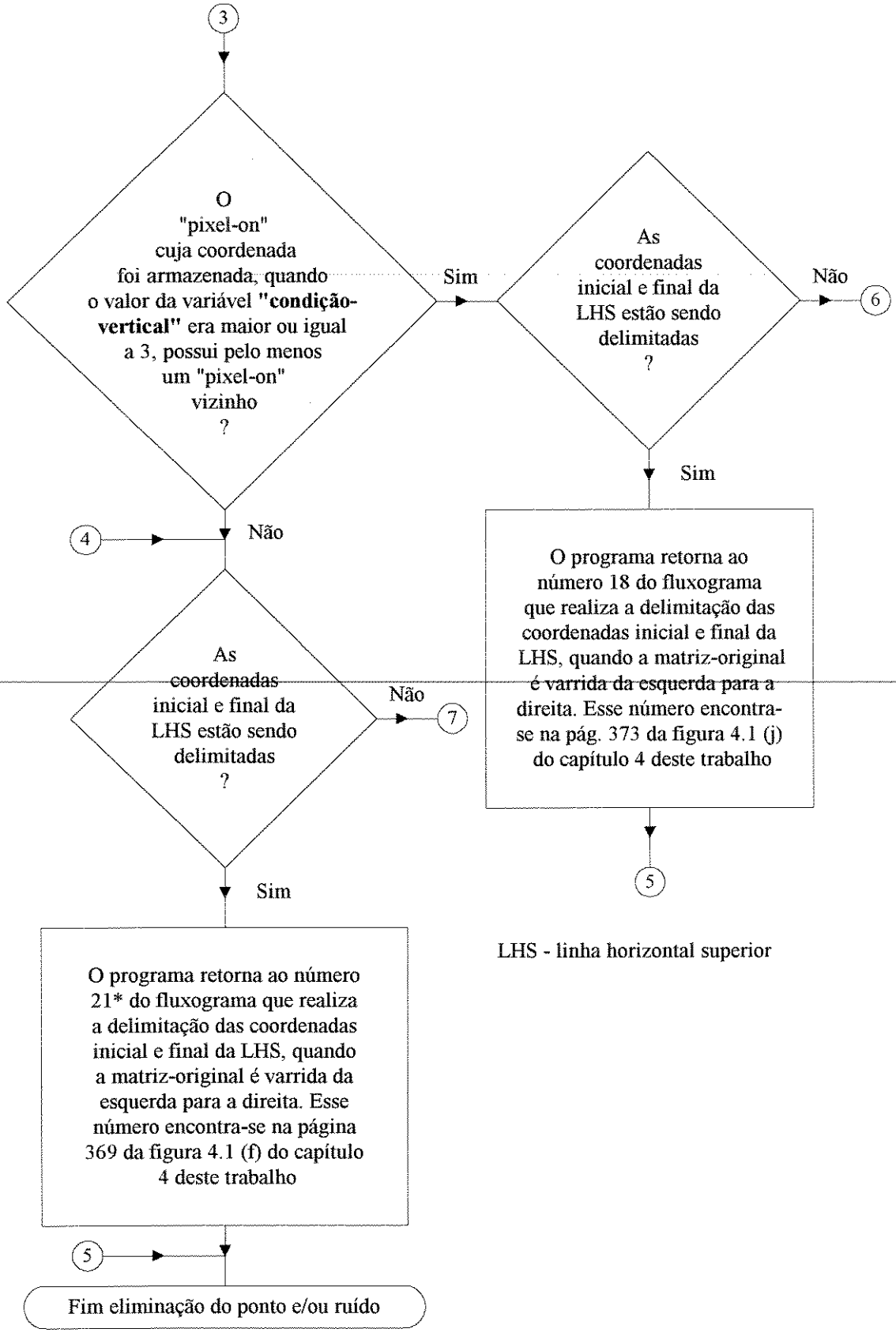


Figura 5.3(d) - Quarta parte do fluxograma do processo de eliminação do ponto e/ou ruído.

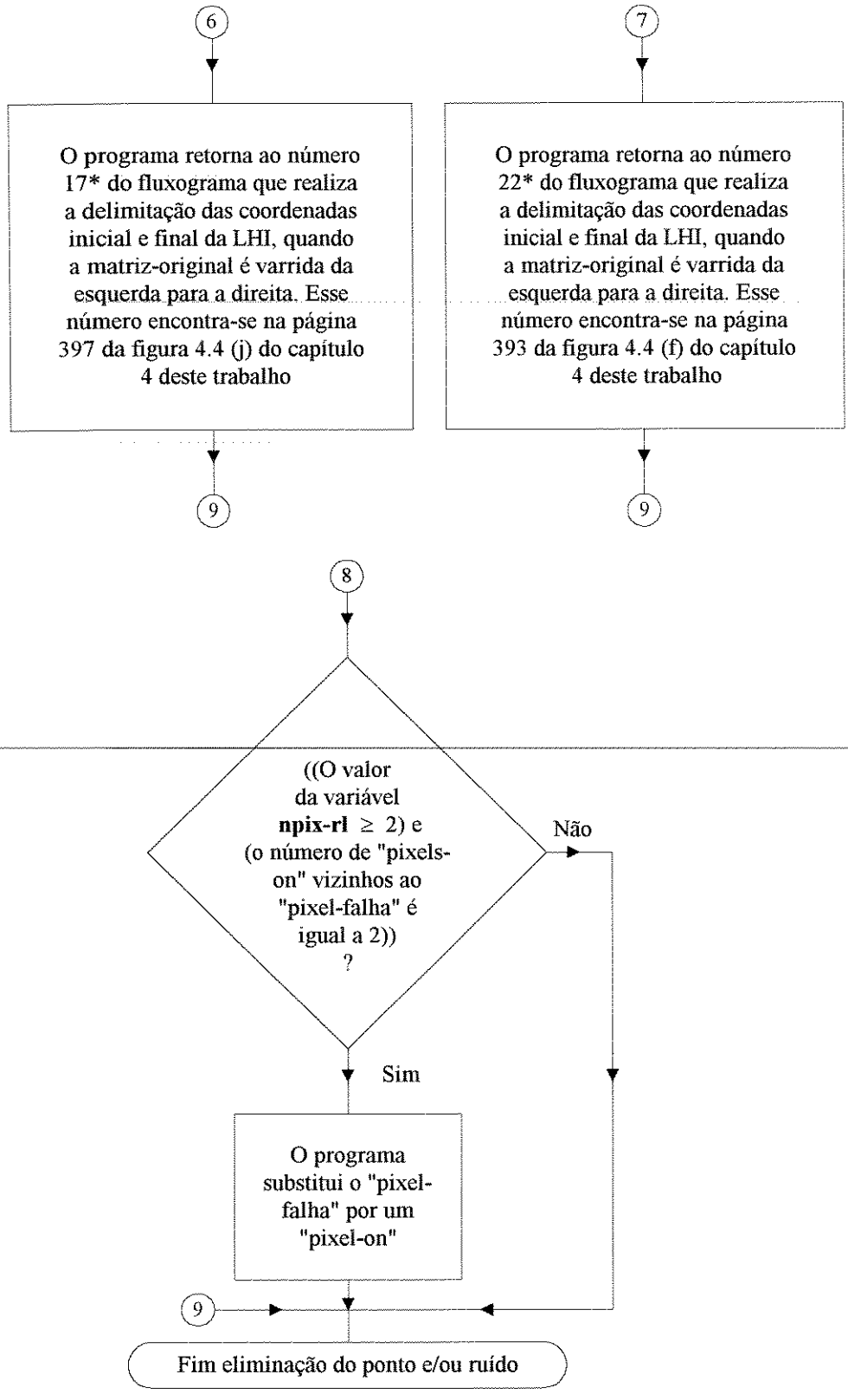


Figura 5.3(e) - Quinta parte do fluxograma do processo de eliminação do ponto e/ou ruído.

Fluxograma de todo o processo de reconhecimento de caracteres manuscritos na área reservada à escrita da quantia numérica de cheques bancários

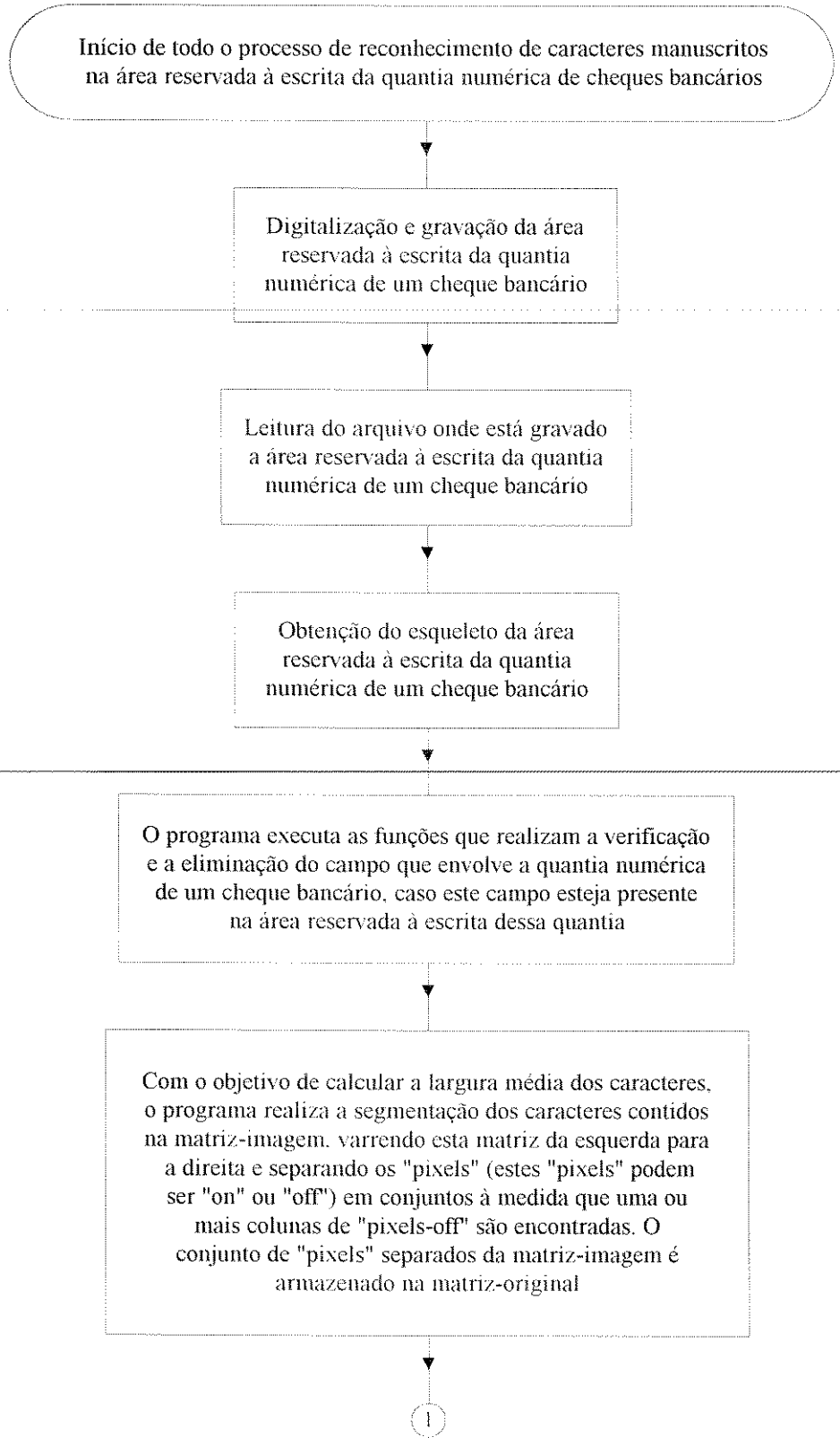


Figura 5.4(a) - Primeira parte do fluxograma de todo o processo de reconhecimento de caracteres manuscritos na área reservada à escrita da quantia numérica de cheques bancários.

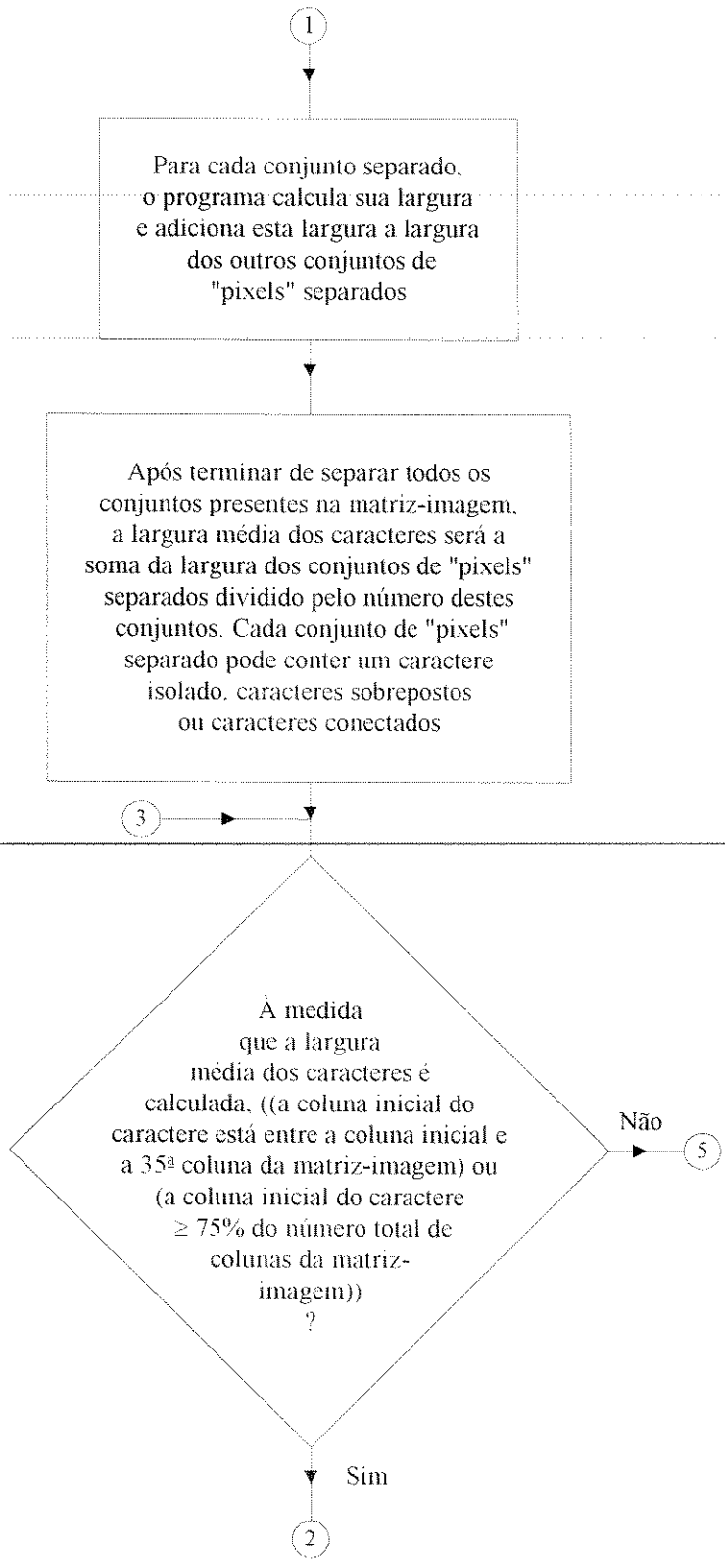


Figura 5.4(b) - Segunda parte do fluxograma de todo o processo de reconhecimento de caracteres manuscritos na área reservada à escrita da quantia numérica de cheques bancários.

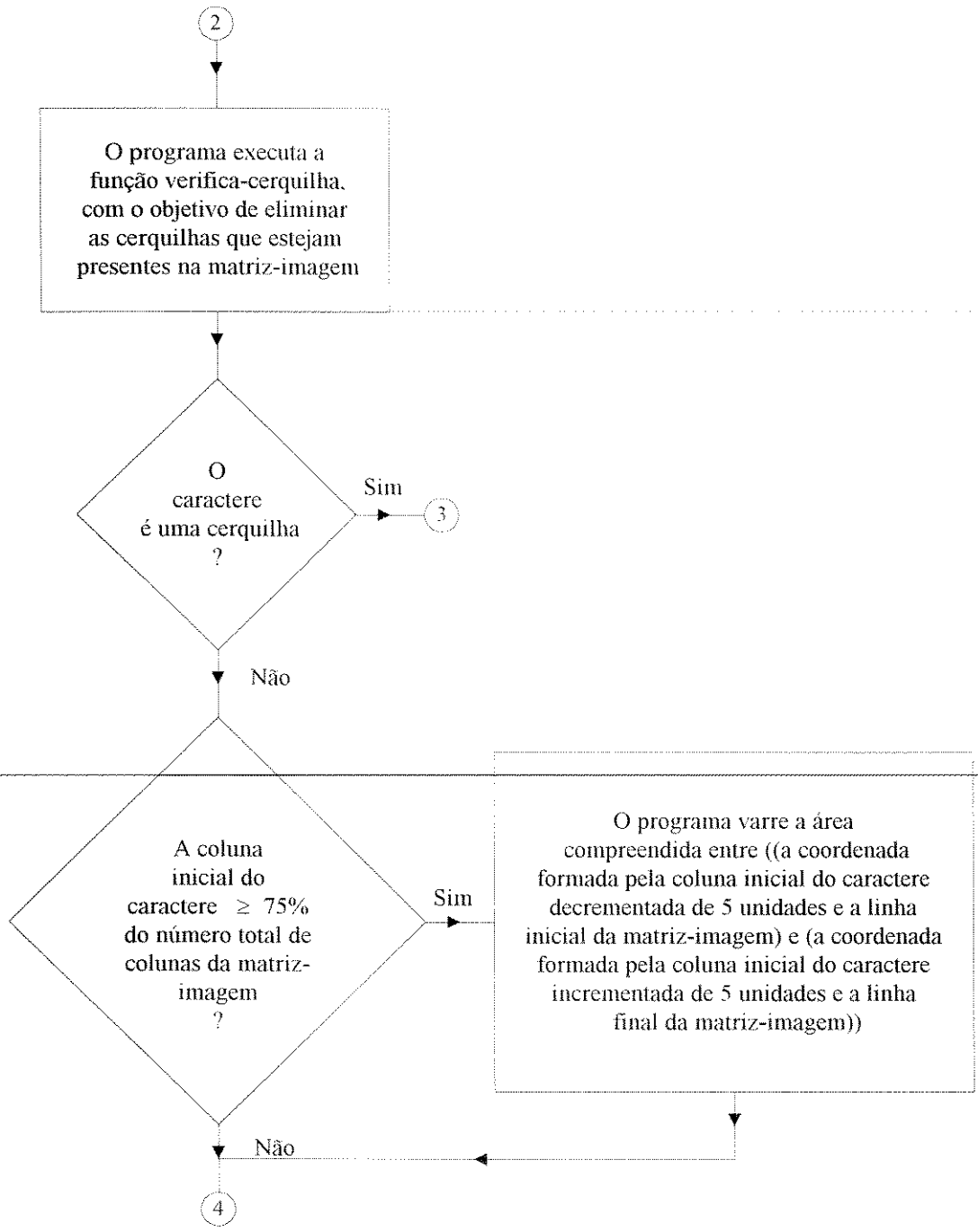


Figura 5.4(c) - Terceira parte do fluxograma de todo o processo de reconhecimento de caracteres manuscritos na área reservada à escrita da quantia numérica de cheques bancários.

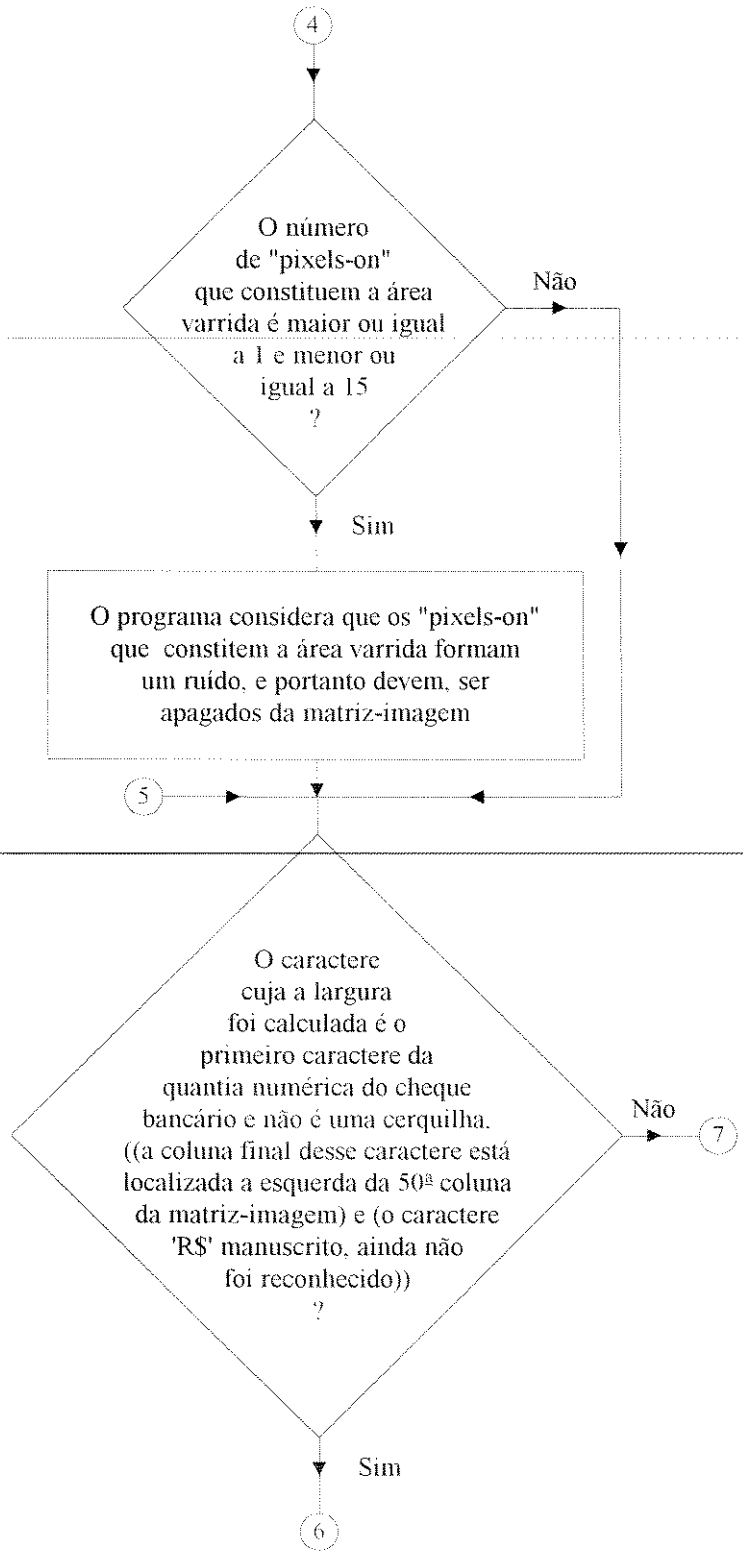


Figura 5.4(d) - Quarta parte do fluxograma de todo o processo de reconhecimento de caracteres manuscritos na área reservada à escrita da quantia numérica de cheques bancários.

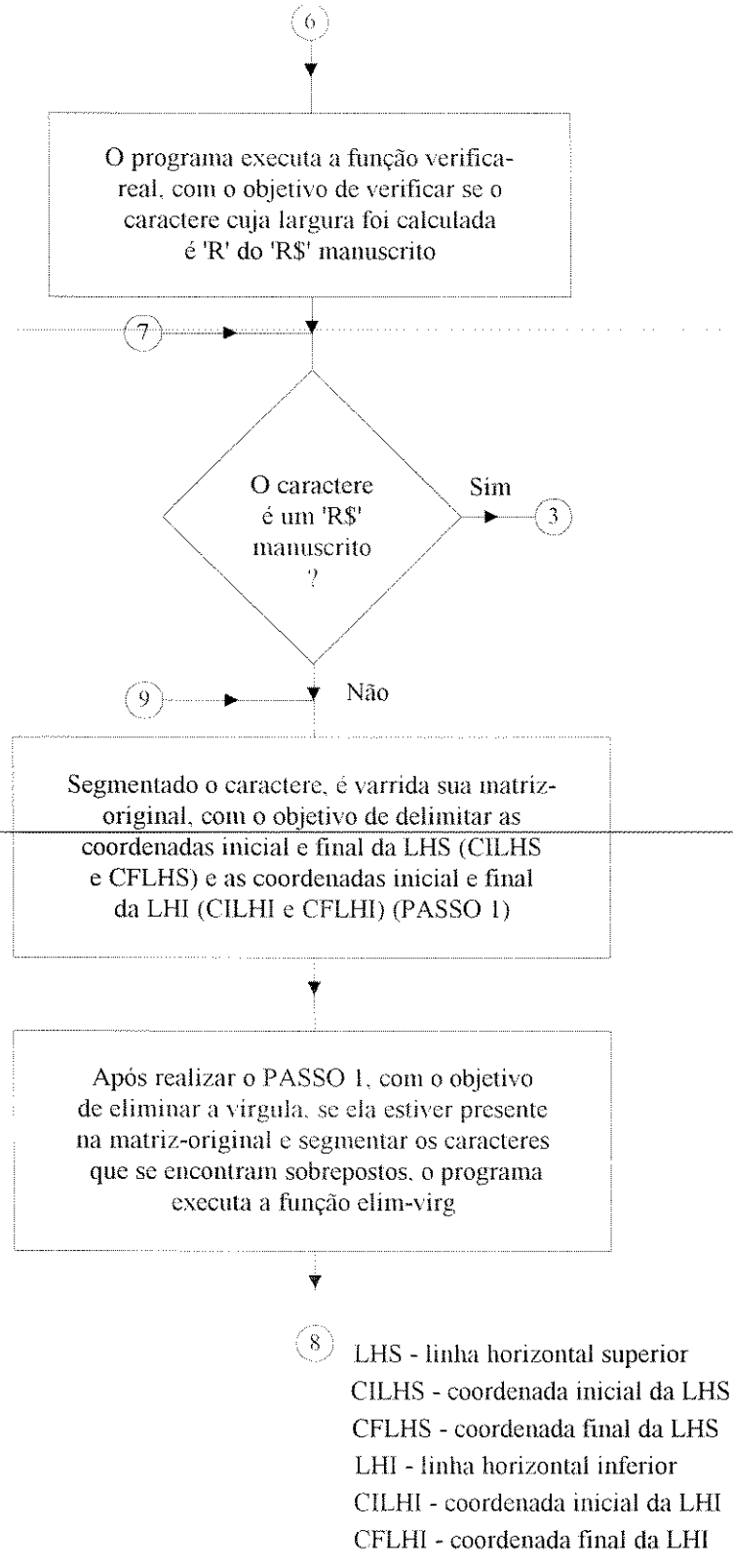


Figura 5.4(e) - Quinta parte do fluxograma de todo o processo de reconhecimento de caracteres manuscritos na área reservada à escrita da quantia numérica de cheques bancários.

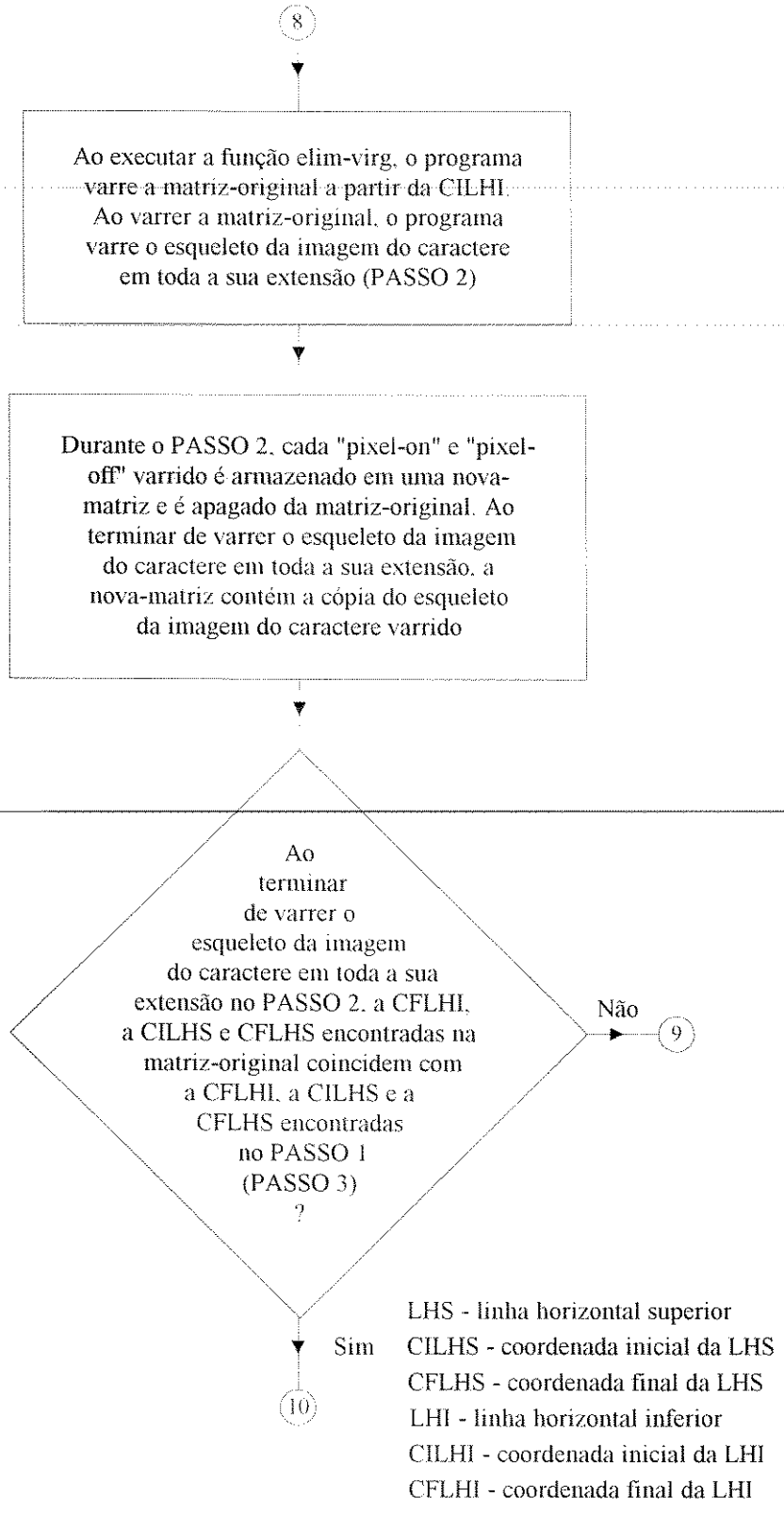


Figura 5.4(f) - Sexta parte do fluxograma de todo o processo de reconhecimento de caracteres manuscritos na área reservada à escrita da quantia numérica de cheques bancários.

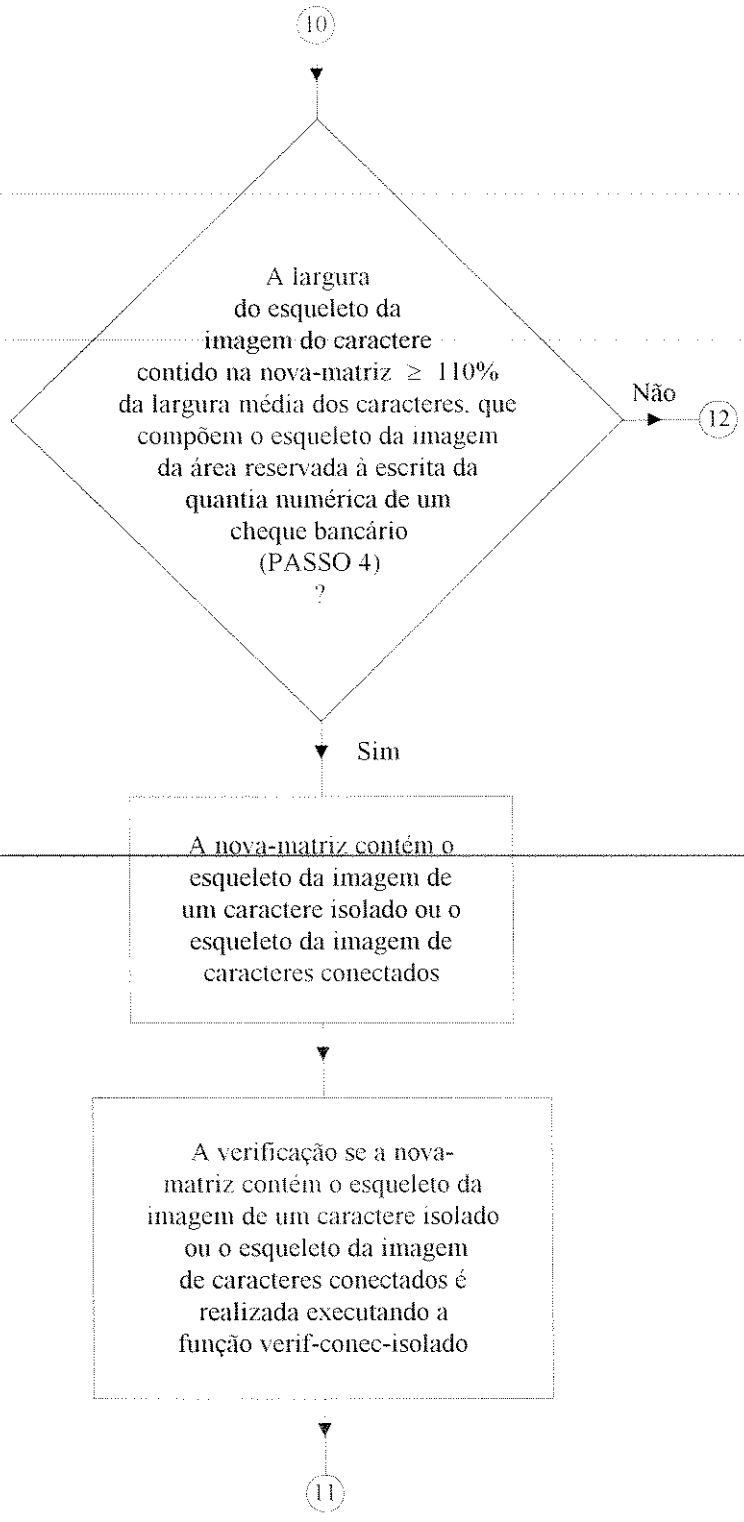


Figura 5.4(g) - Sétima parte do fluxograma de todo o processo de reconhecimento de caracteres manuscritos na área reservada à escrita da quantia numérica de cheques bancários.

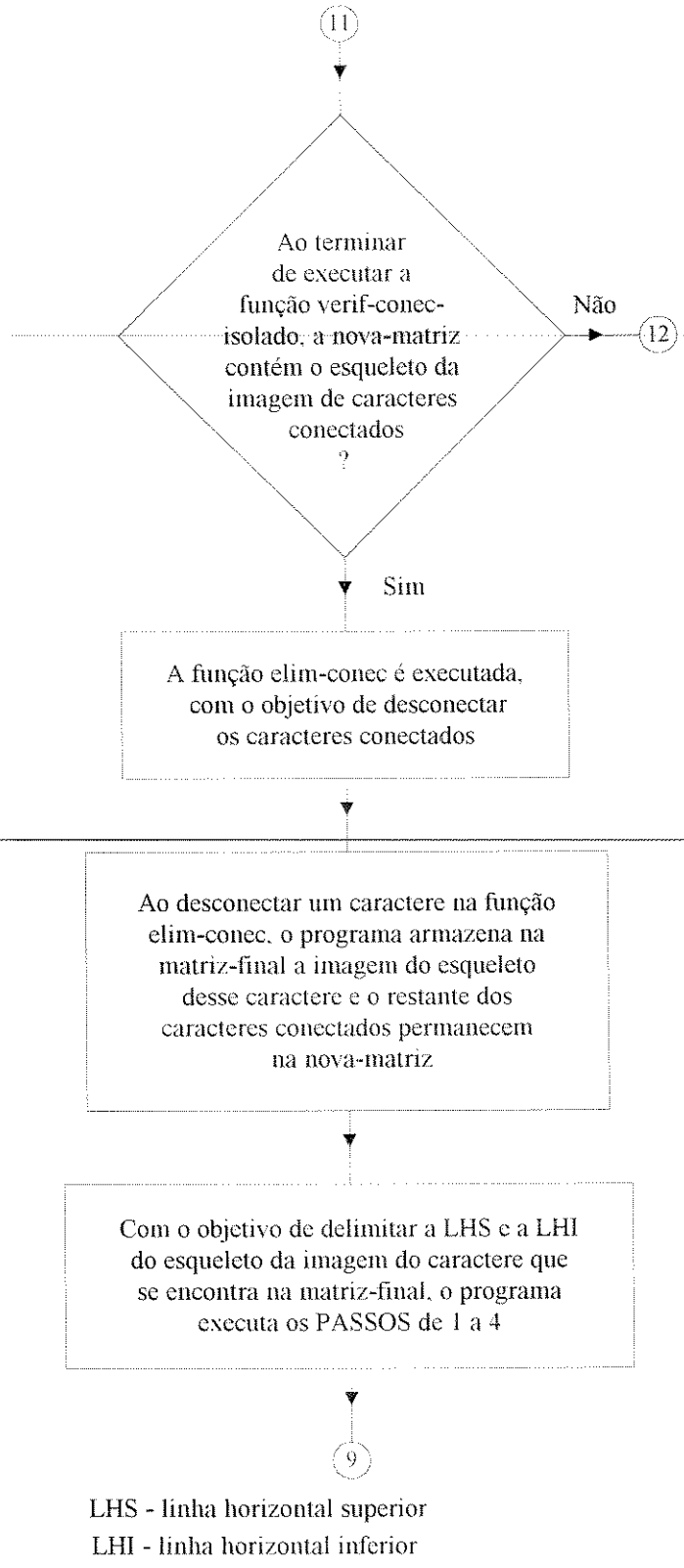


Figura 5.4(h) - Oitava parte do fluxograma de todo o processo de reconhecimento de caracteres manuscritos na área reservada à escrita da quantia numérica de cheques bancários.

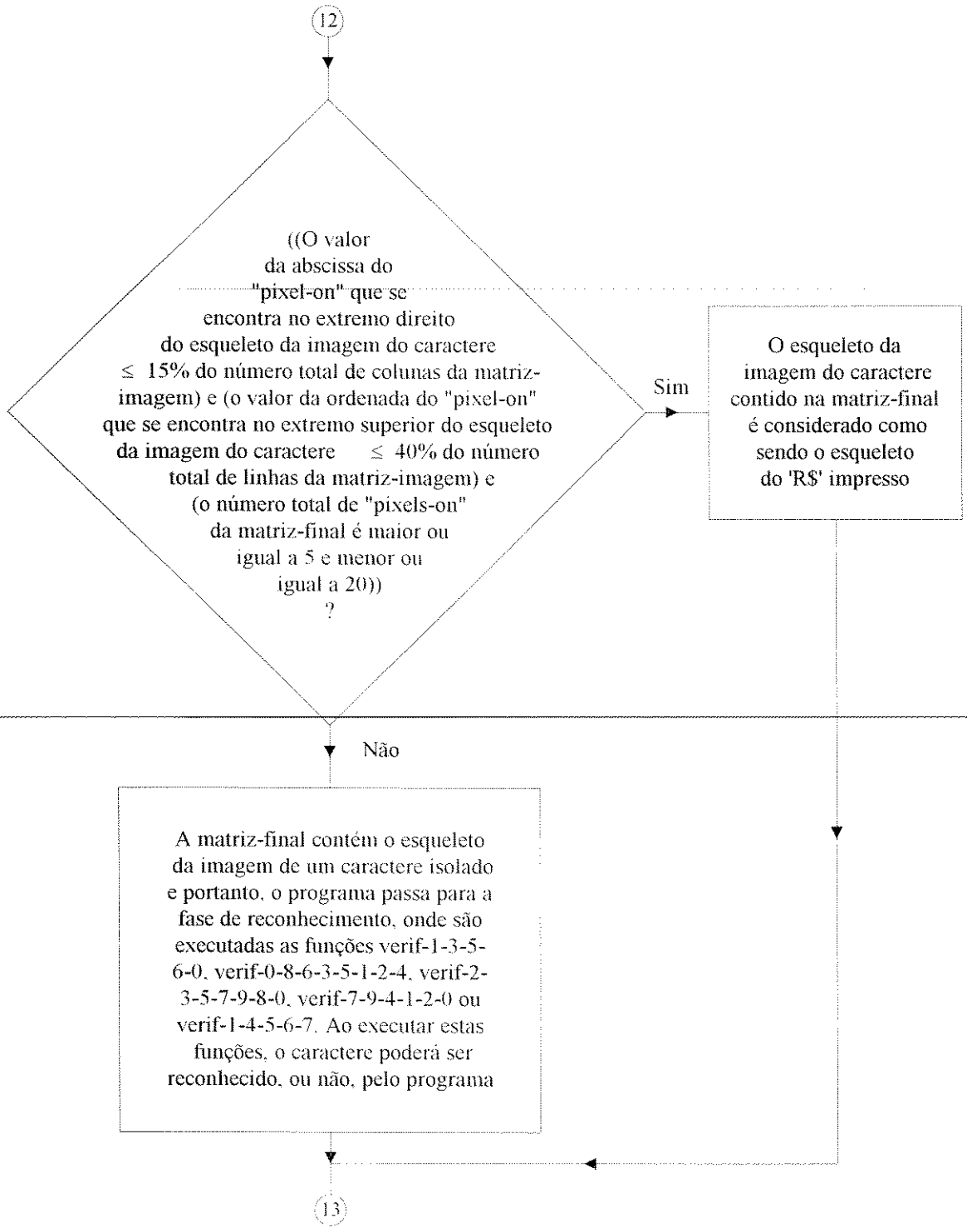


Figura 5.4(i) - Nona parte do fluxograma de todo o processo de reconhecimento de caracteres manuscritos na área reservada à escrita da quantia numérica de cheques bancários.

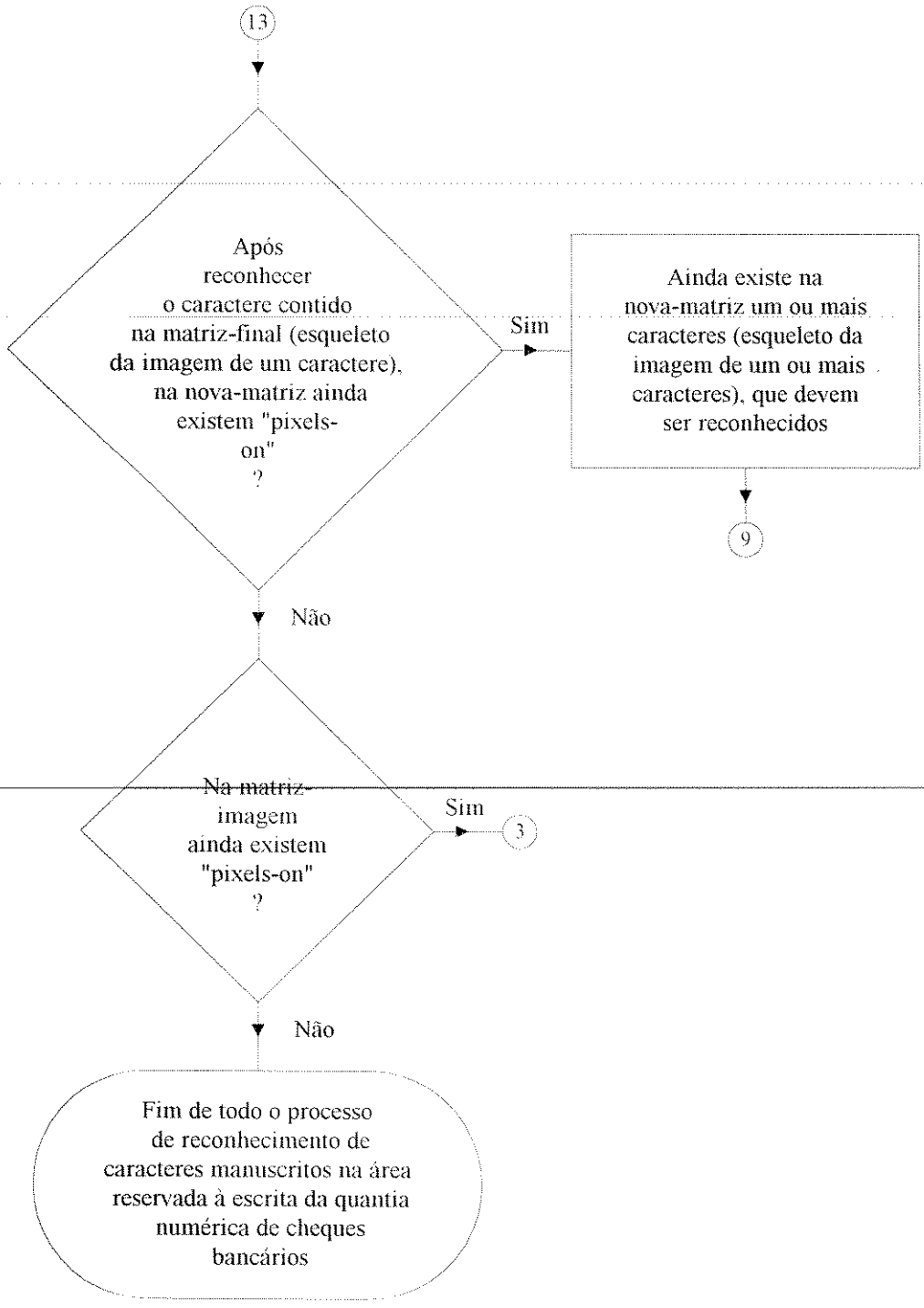
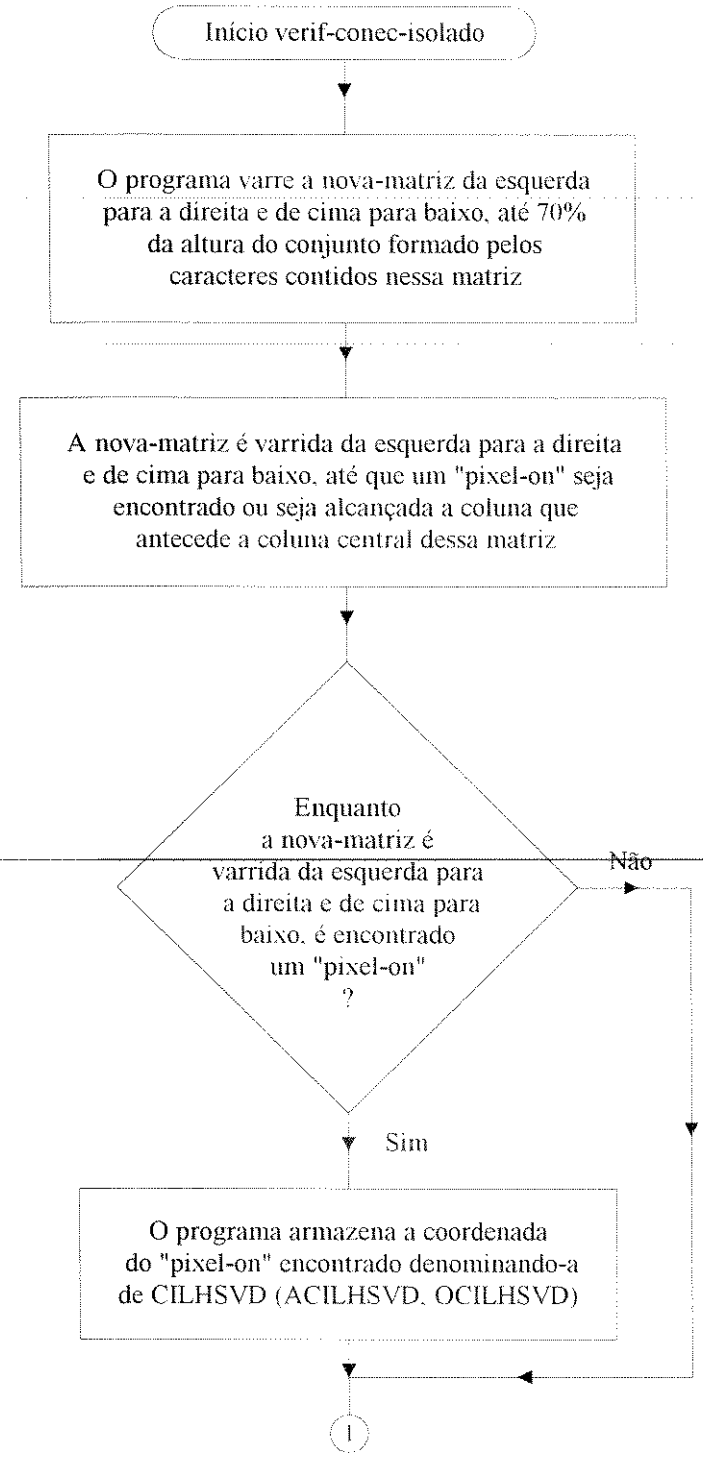


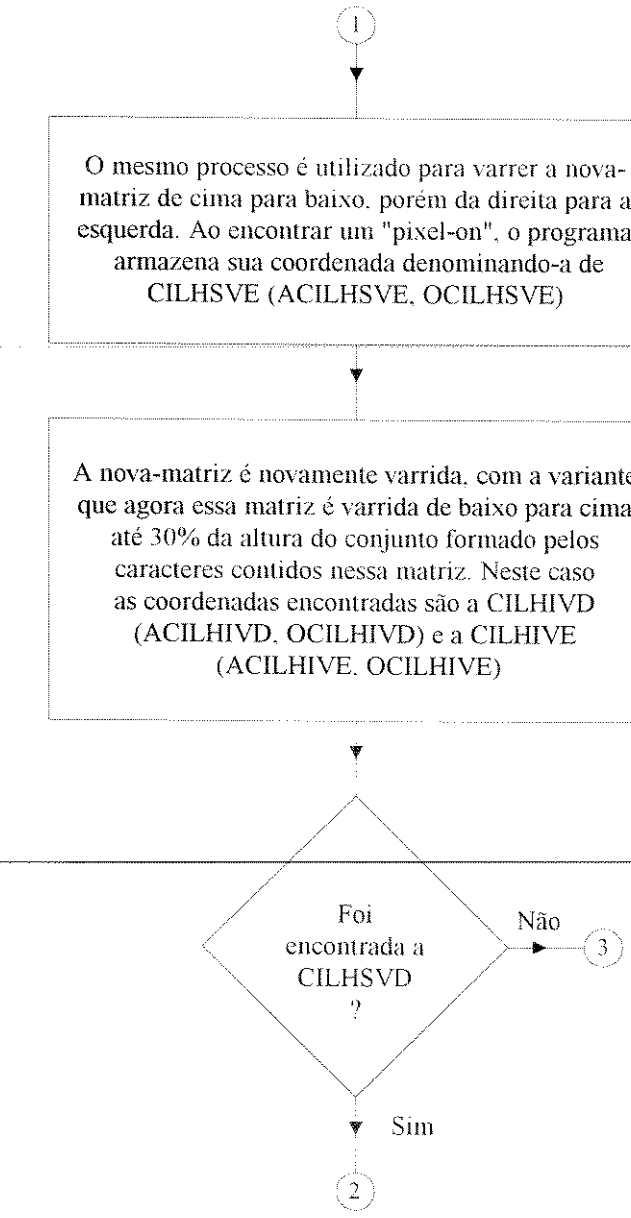
Figura 5.4(j) - Décima parte do fluxograma de todo o processo de reconhecimento de caracteres manuscritos na área reservada à escrita da quantia numérica de cheques bancários.

Fluxograma da função verif-conec-isolado



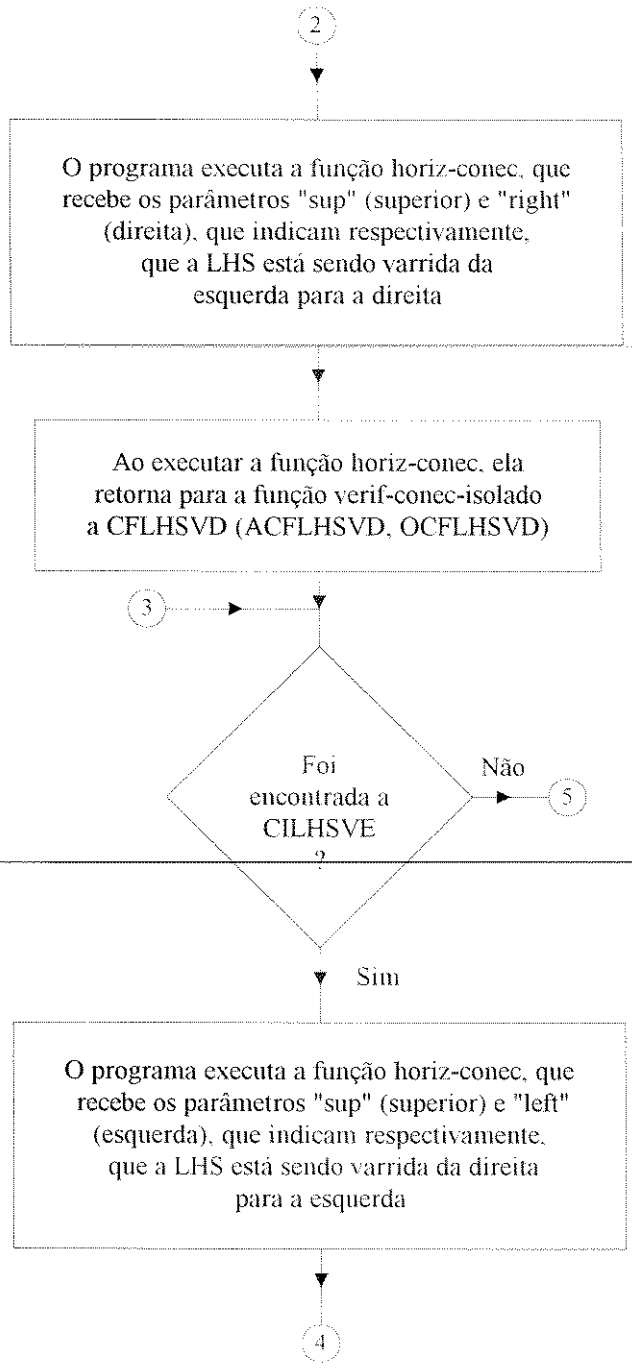
LHS - linha horizontal superior
CILHSVD - coordenada inicial da LHS varrida para a direita
ACILHSVD - abscissa da coordenada inicial da LHS varrida para a direita
OCILHSVD - ordenada da coordenada inicial da LHS varrida para a direita

Figura 5.6(a) - Primeira parte do fluxograma da função verif-conec-isolado.



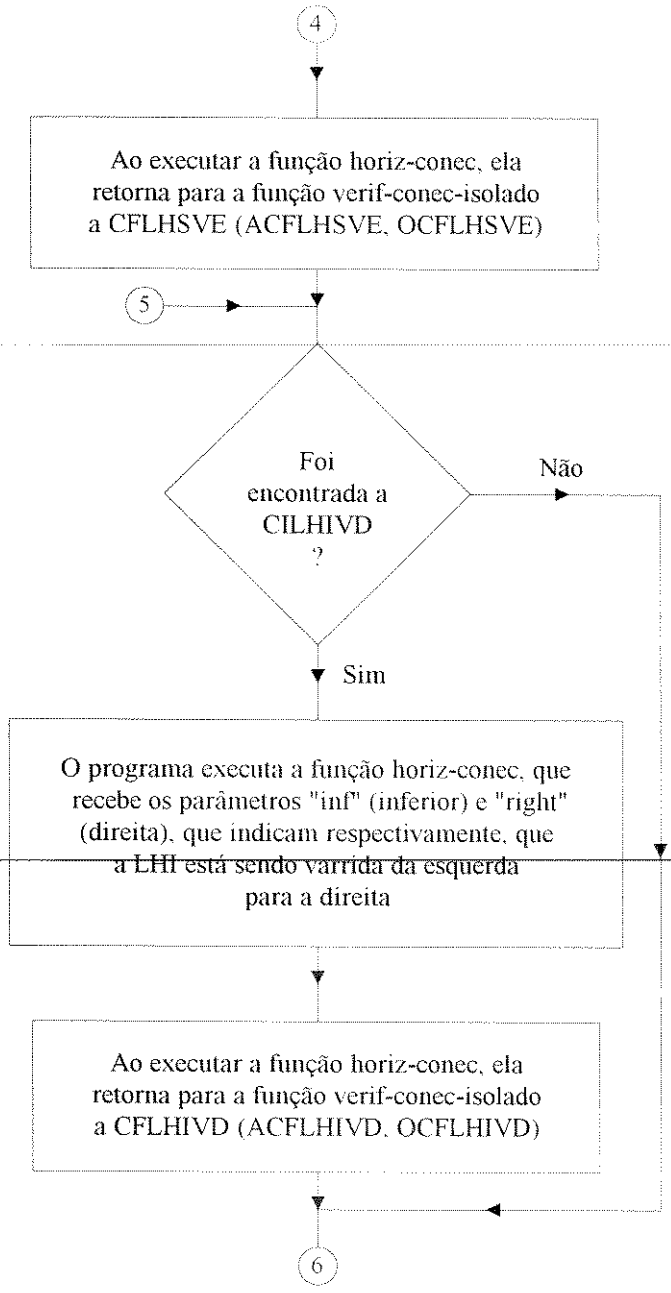
LHS - linha horizontal superior
CILHSVE - coordenada inicial da LHS varrida para a esquerda
ACILHSE - abscissa da coordenada inicial da LHS varrida para a esquerda
OCILHSVE - ordenada da coordenada inicial da LHS varrida para a esquerda
CILHSVD - coordenada inicial da LHS varrida para a direita
LHI - linha horizontal inferior
CILHIVD - coordenada inicial da LHI varrida para a direita
ACILHIVD - abscissa da coordenada inicial da LHI varrida para a direita
OCILHIVD - ordenada da coordenada inicial da LHI varrida para a direita
CILHIVE - coordenada inicial da LHI varrida para a esquerda
ACILHIVE - abscissa da coordenada inicial da LHI varrida para a esquerda
OCILHIVE - ordenada da coordenada inicial da LHI varrida para a esquerda

Figura 5.6(b) - Segunda parte do fluxograma da função verif-conec-isolado.



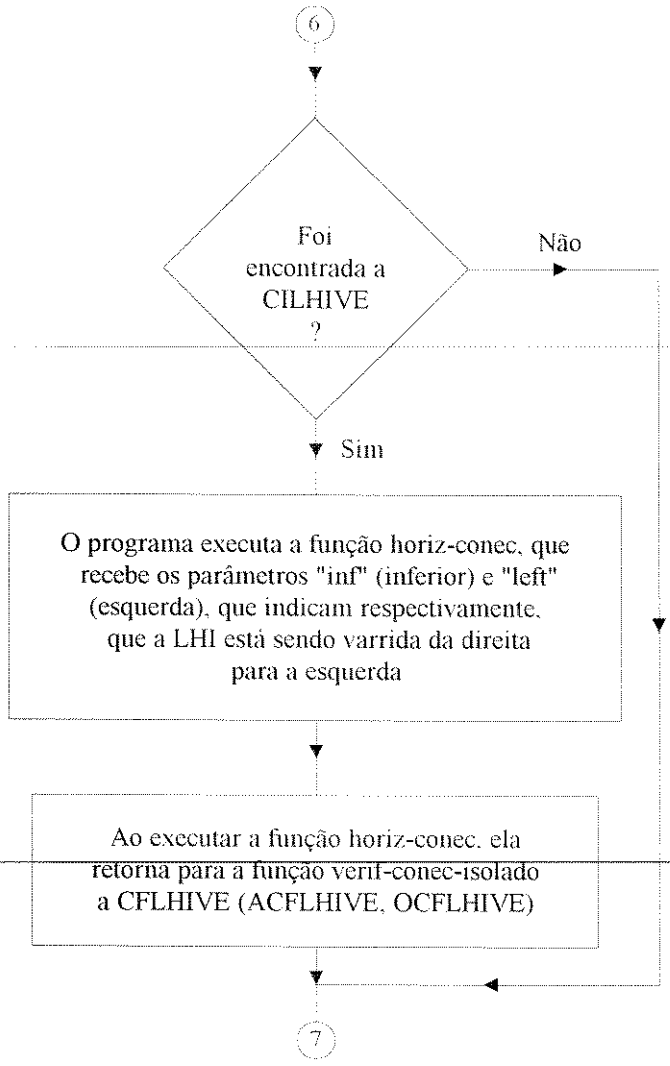
LHS - linha horizontal superior
CFLHSVD - coordenada final da LHS varrida para a direita
ACFLHSVD - abscissa da coordenada final da LHS varrida para a direita
OCFLHSVD - ordenada da coordenada final da LHS varrida para a direita
CILHSVE - coordenada inicial da LHS varrida para a esquerda

Figura 5.6(c) - Terceira parte do fluxograma da função verif-conec-isolado.



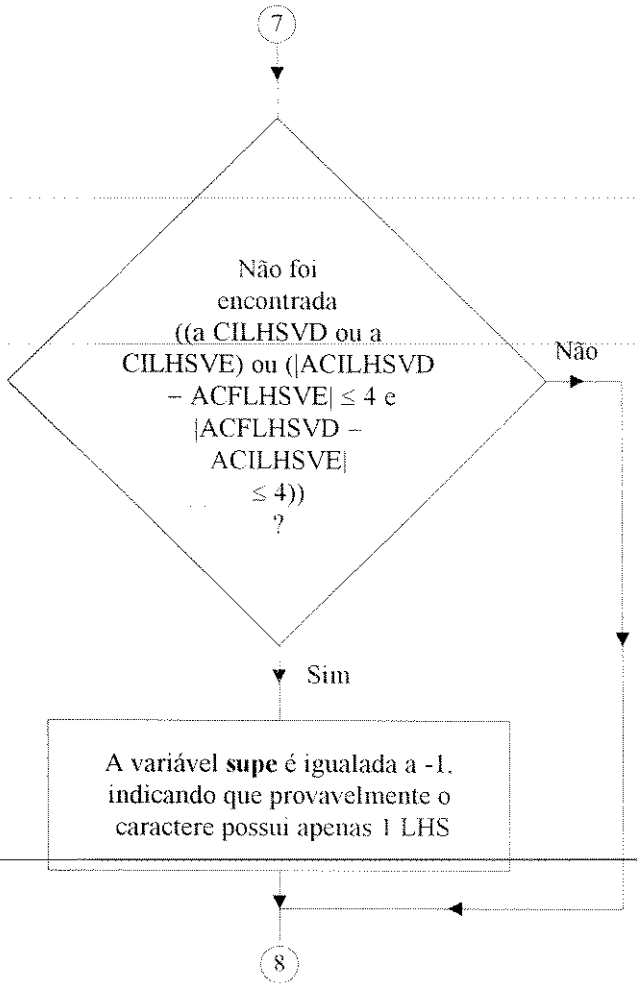
LHS - linha horizontal superior
CFLHSVE - coordenada final da LHS varrida para a esquerda
ACFLHSVE - abscissa da coordenada final da LHS varrida para a esquerda
OCFLHSVE - ordenada da coordenada final da LHS varrida para a esquerda
LHI - linha horizontal inferior
CILHIVD - coordenada inicial da LHI varrida para a direita
CFLHIVD - coordenada final da LHI varrida para a direita
ACFLHIVD - abscissa da coordenada final da LHI varrida para a direita
OCFLHIVD - ordenada da coordenada final da LHI varrida para a direita

Figura 5.6(d) - Quarta parte do fluxograma da função verif-conec-isolado.



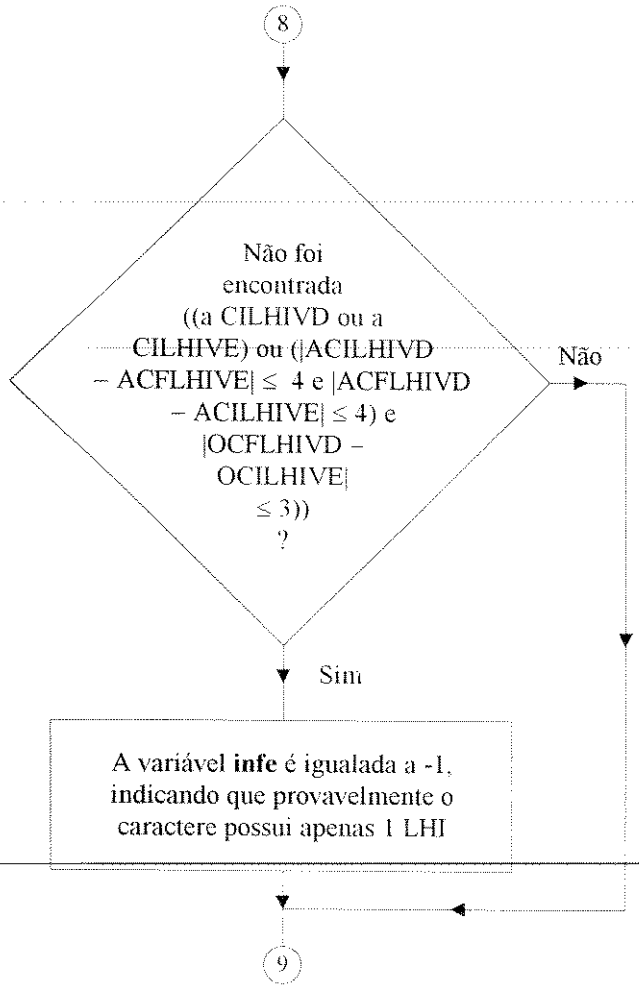
LHI - linha horizontal inferior
CILHIVE - coordenada inicial da LHI varrida para a esquerda
CFLHIVE - coordenada final da LHI varrida para a esquerda
ACFLHIVE - abscissa da coordenada final da LHI varrida para a esquerda
OCFLHIVE - ordenada da coordenada final da LHI varrida para a esquerda

Figura 5.6(e) - Quinta parte do fluxograma da função verif-conec-isolado.



LHS - linha horizontal superior
CILHSVD - coordenada inicial da LHS varrida para a direita
ACILHSVD - abscissa da coordenada inicial da LHS varrida para a direita
ACFLHSVD - abscissa da coordenada final da LHS varrida para a direita
CILHSVE - coordenada inicial da LHS varrida para a esquerda
ACILHSVE - abscissa da coordenada inicial da LHS varrida para a esquerda
ACFLHSVE - abscissa da coordenada final da LHS varrida para a esquerda

Figura 5.6(f) - Sexta parte do fluxograma da função verif-conec-isolado.



LHI - linha horizontal inferior
CILHIVD - coordenada inicial da LHI varrida para a direita
ACILHIVD - abscissa da coordenada inicial da LHI varrida para a direita
ACFLHIVD - abscissa da coordenada final da LHI varrida para a direita
OCFLHIVD - ordenada da coordenada final da LHI varrida para a direita
CILHIVE - coordenada inicial da LHI varrida para a esquerda
ACILHIVE - abscissa da coordenada inicial da LHI varrida para a esquerda
OCILHIVE - ordenada da coordenada inicial da LHI varrida para a esquerda
ACFLHIVE - abscissa da coordenada final da LHI varrida para a esquerda

Figura 5.6(g) - Sétima parte do fluxograma da função verif-conec-isolado.

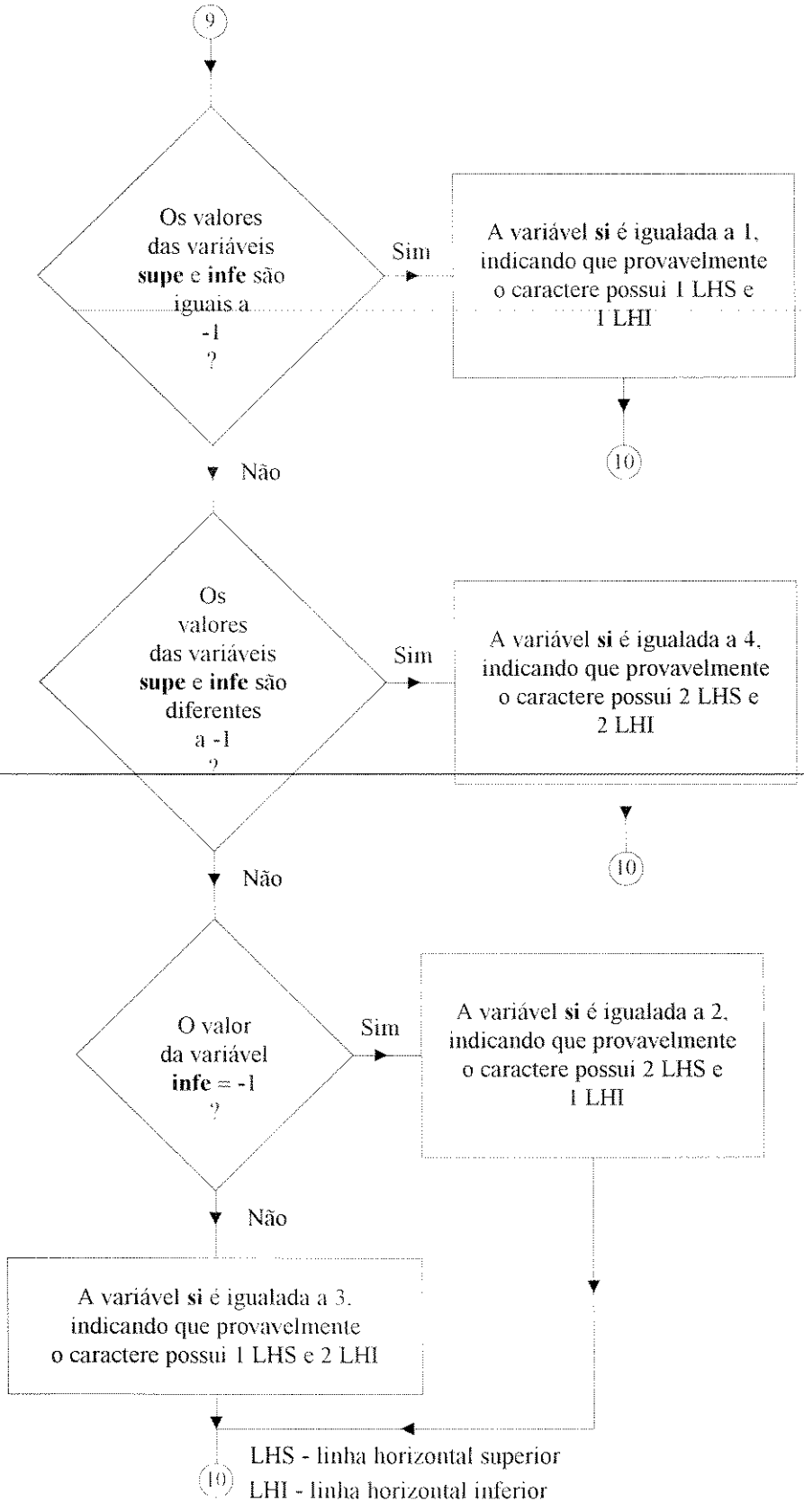


Figura 5.6(h) - Oitava parte do fluxograma da função verif-conec-isolado.

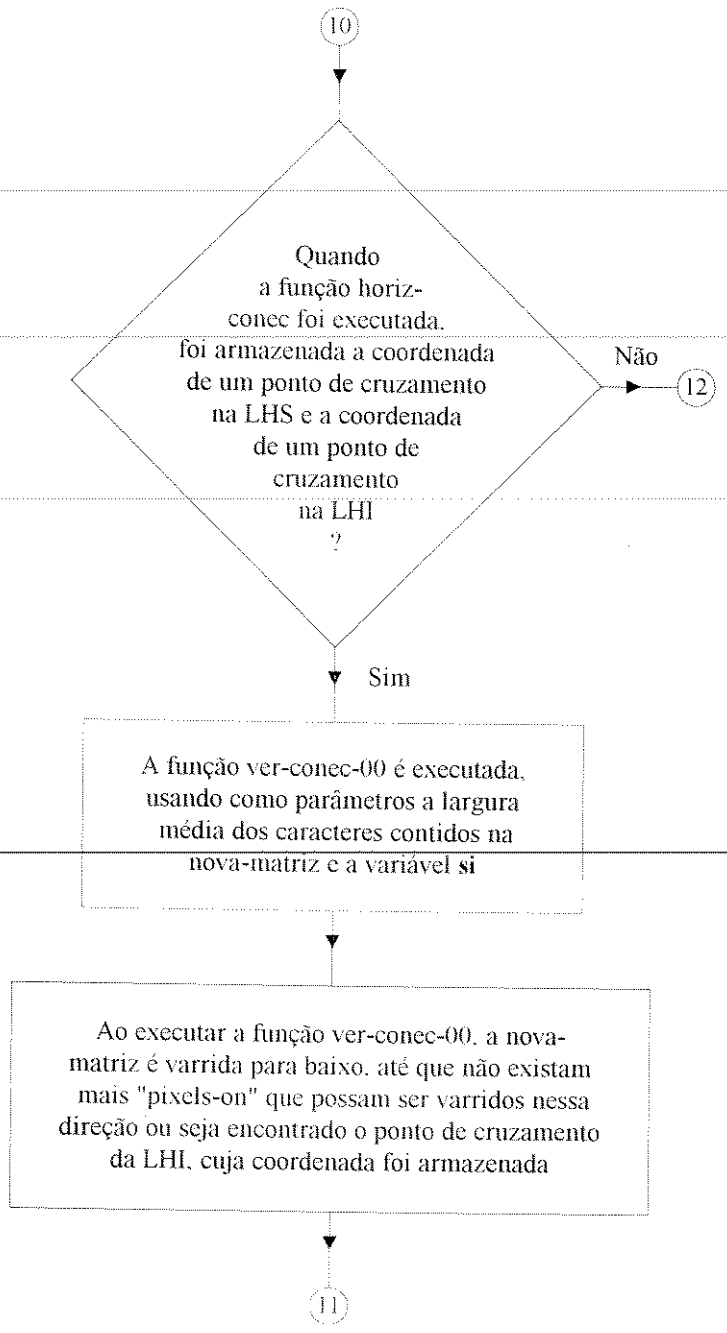
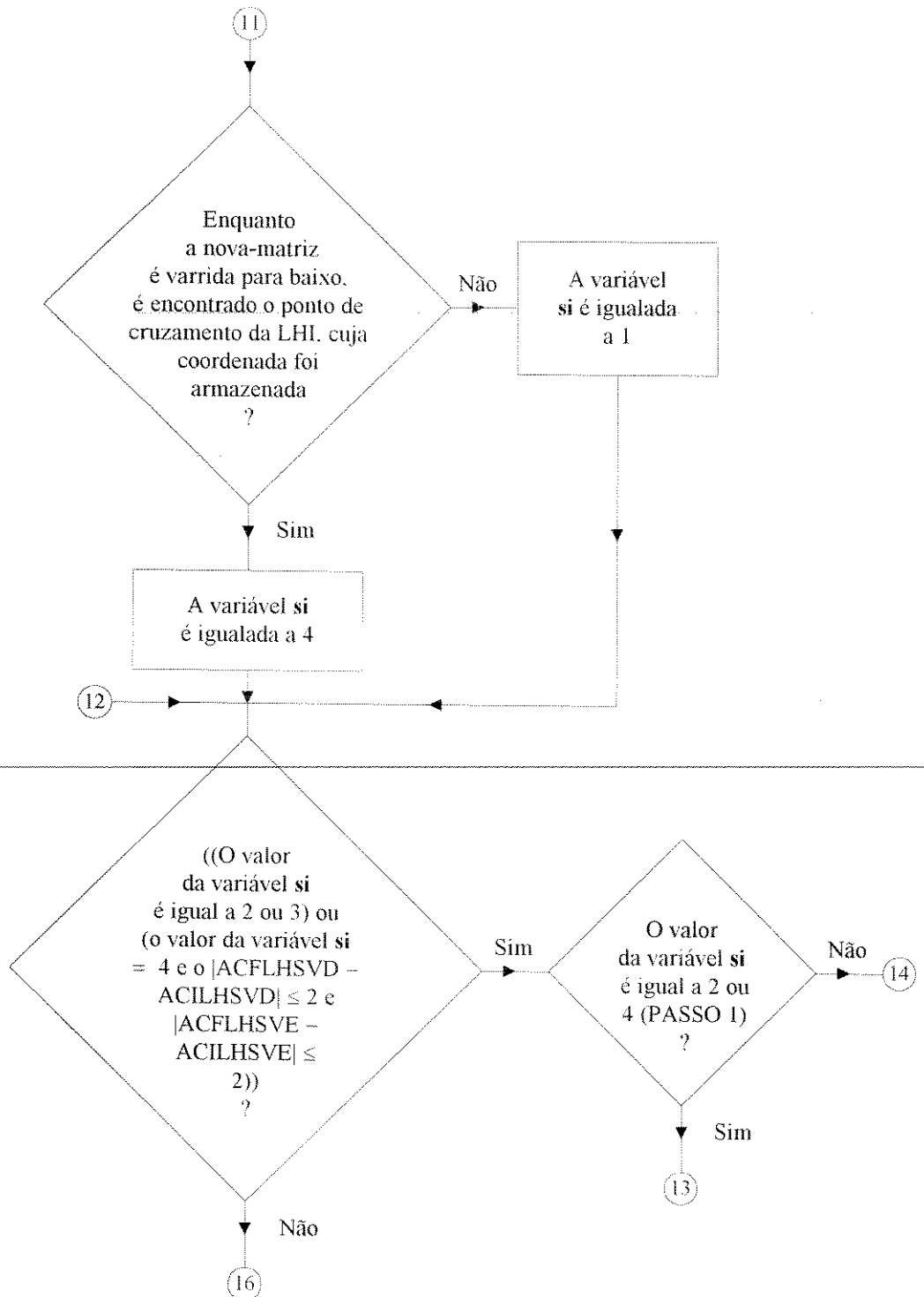
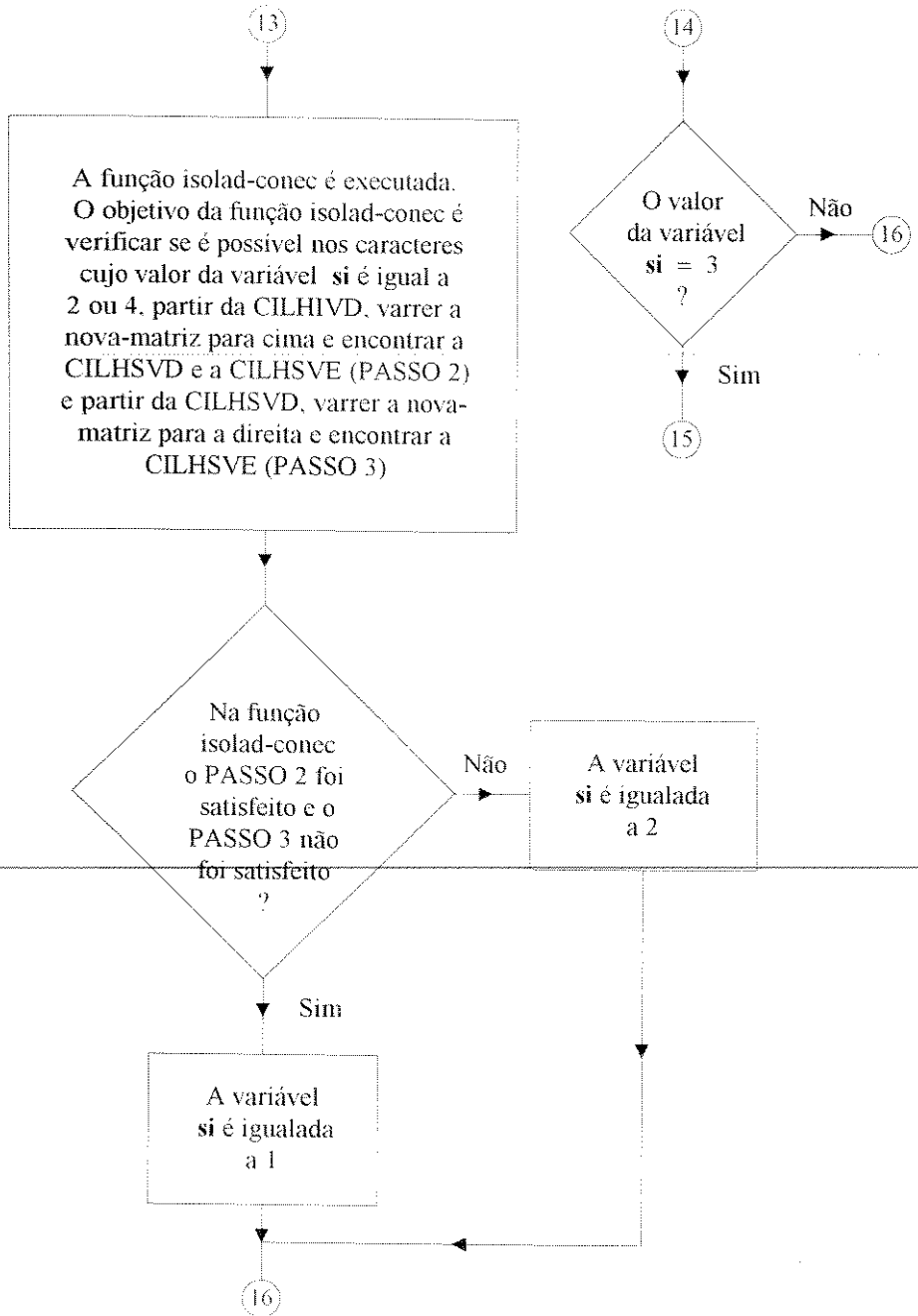


Figura 5.6(i) - Nona parte do fluxograma da função verif-conec-isolado.



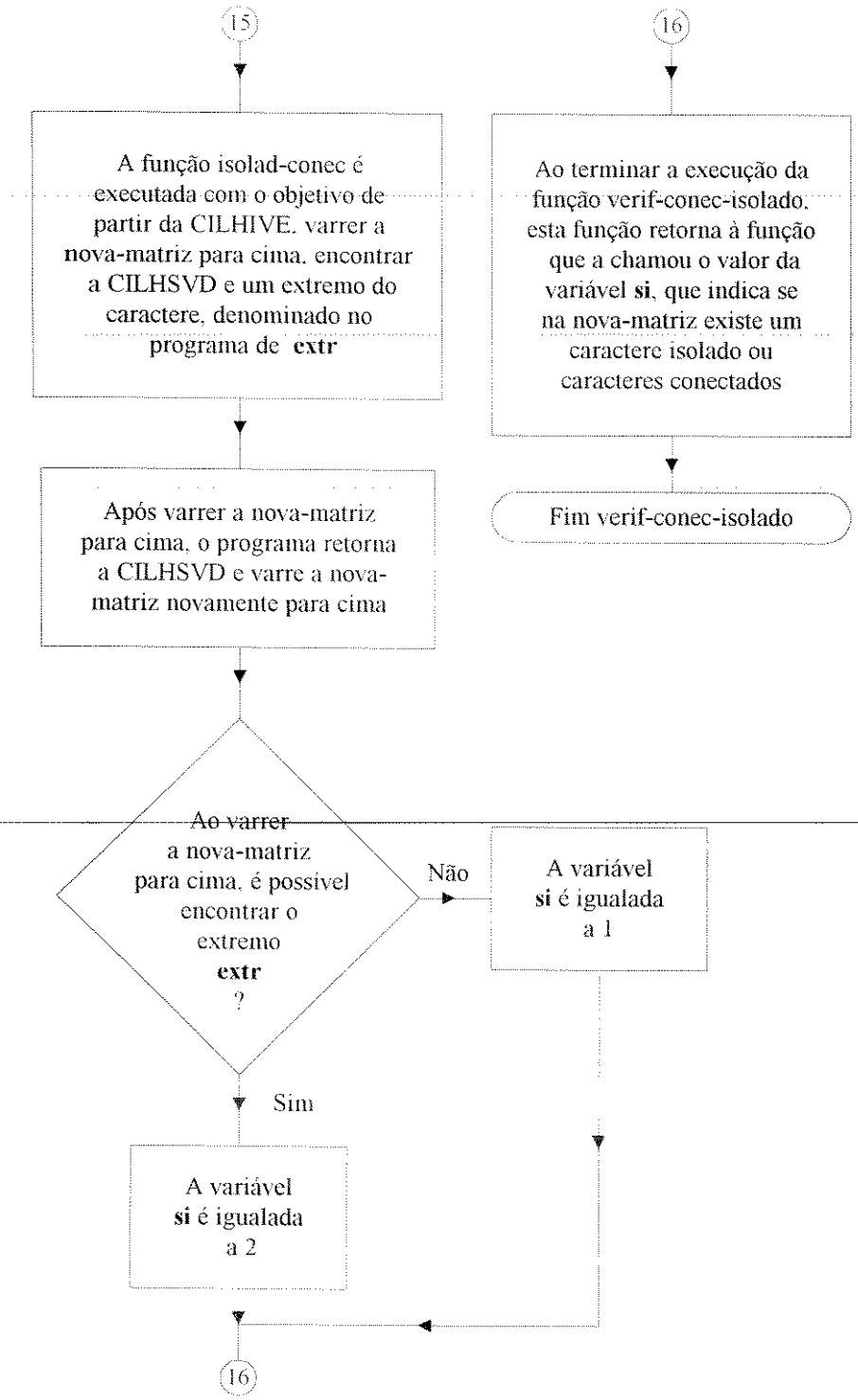
LHS - linha horizontal superior
ACILHSVD - abscissa da coordenada inicial da LHS varrida para a direita
ACFLHSVD - abscissa da coordenada final da LHS varrida para a direita
ACILHSVE - abscissa da coordenada inicial da LHS varrida para a esquerda
ACFLHSVE - abscissa da coordenada final da LHS varrida para a esquerda

Figura 5.6(j) - Décima parte do fluxograma da função verif-conec-isolado.



LHS - linha horizontal superior
CILHSVE - coordenada inicial da LHS varrida para a esquerda
CILHSVD - coordenada inicial da LHS varrida para a direita
LHI - linha horizontal inferior
CILHIVD - coordenada inicial da LHI varrida para a direita

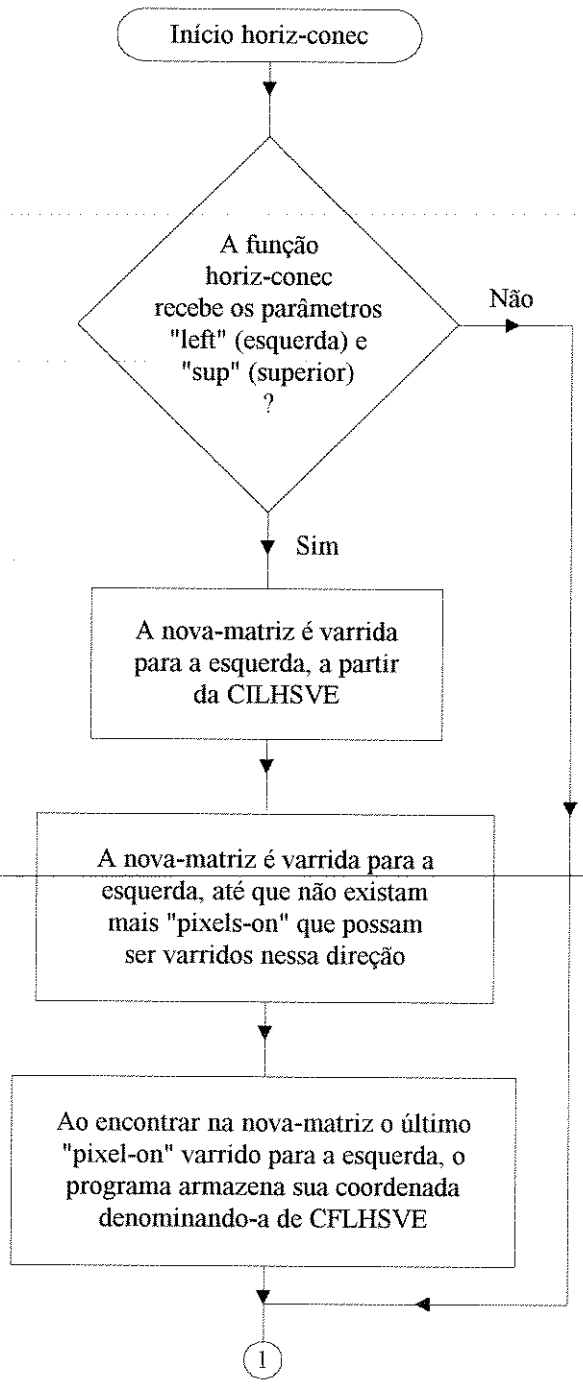
Figura 5.6(k) - Décima primeira parte do fluxograma da função verif-conec-isolado.



LHS - linha horizontal superior
CILHSVD - coordenada inicial da LHS varrida para a direita
LHI - linha horizontal inferior
CILHIVE - coordenada inicial da LHI varrida para a esquerda

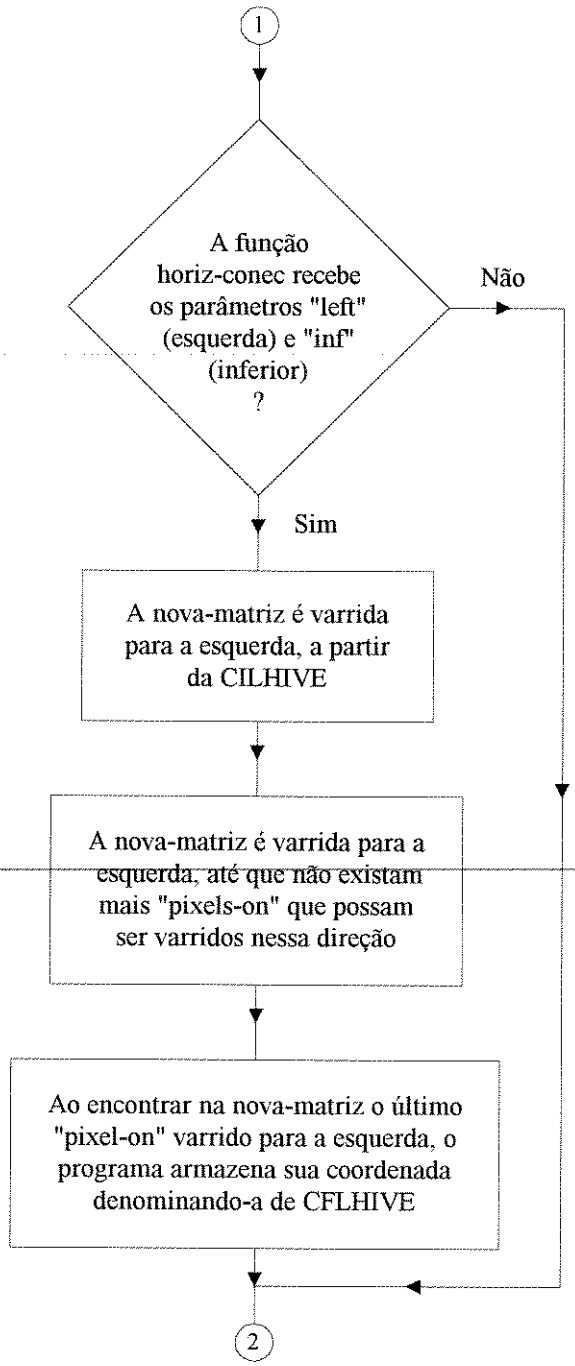
Figura 5.6(l) - Décima segunda parte do fluxograma da função verif-conec-isolado.

Fluxograma da função horiz-conec



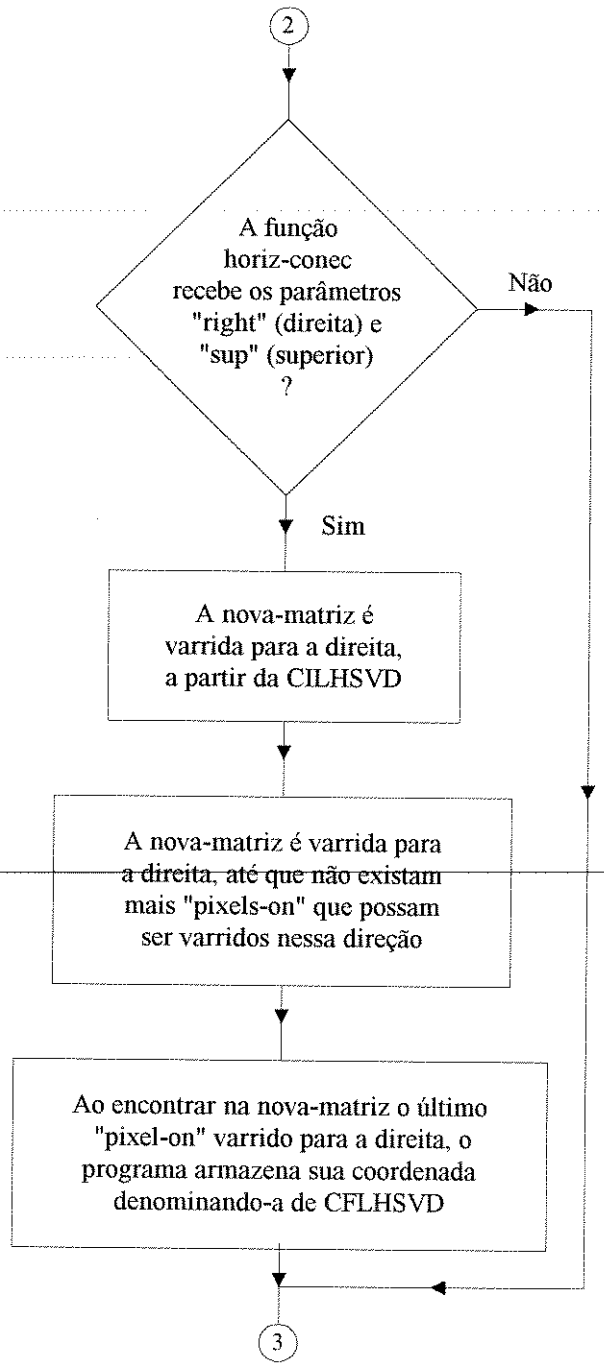
LHS - linha horizontal superior
CILHSVE - coordenada inicial da LHS varrida para a esquerda
CFLHSVE - coordenada final da LHS varrida para a esquerda

Figura 5.7(a) - Primeira parte do fluxograma da função horiz-conec.



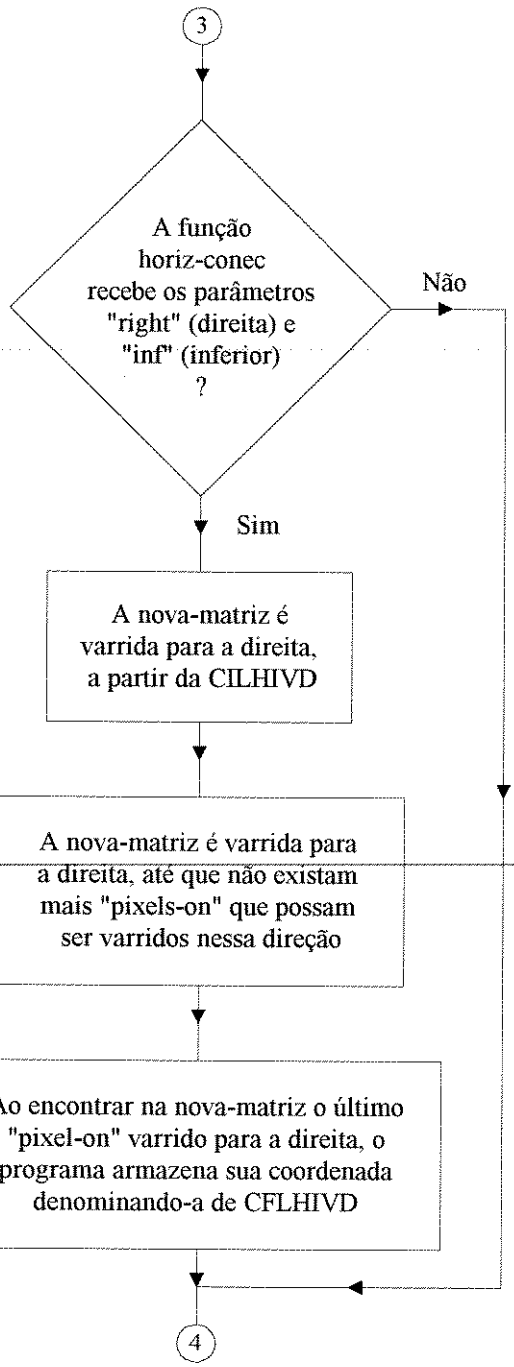
LHI - linha horizontal inferior
CILHIVE - coordenada inicial da LHI varrida para a esquerda
CFLHIVE - coordenada final da LHI varrida para a esquerda

Figura 5.7(b) - Segunda parte do fluxograma da função horiz-conec.



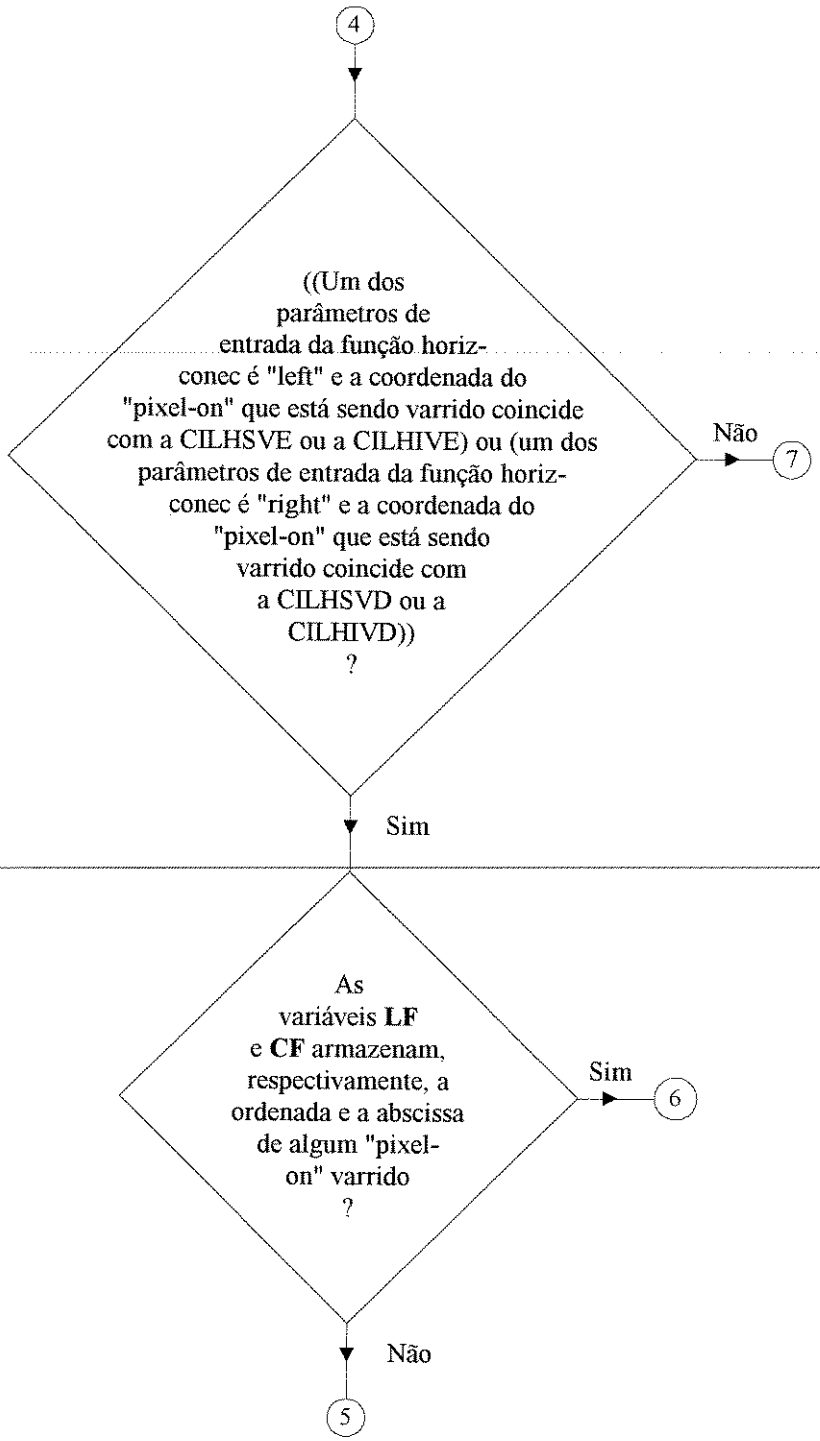
LHS - linha horizontal superior
CILHSVD - coordenada inicial da LHS varrida para a direita
CFLHSVD - coordenada final da LHS varrida para a direita

Figura 5.7(c) - Terceira parte do fluxograma da função horiz-conec.



LHI - linha horizontal inferior
CILHIVD - coordenada inicial da LHI varrida para a direita
CFLHIVD - coordenada final da LHI varrida para a direita

Figura 5.7(d) - Quarta parte do fluxograma da função horiz-conec.



LHS - linha horizontal superior
CILHSVD - coordenada inicial da LHS varrida para a direita
CILHSVE - coordenada inicial da LHS varrida para a esquerda
LHI - linha horizontal inferior
CILHIVD - coordenada inicial da LHI varrida para a direita
CILHIVE - coordenada inicial da LHI varrida para a esquerda

Figura 5.7(e) - Quinta parte do fluxograma da função horiz-conec.

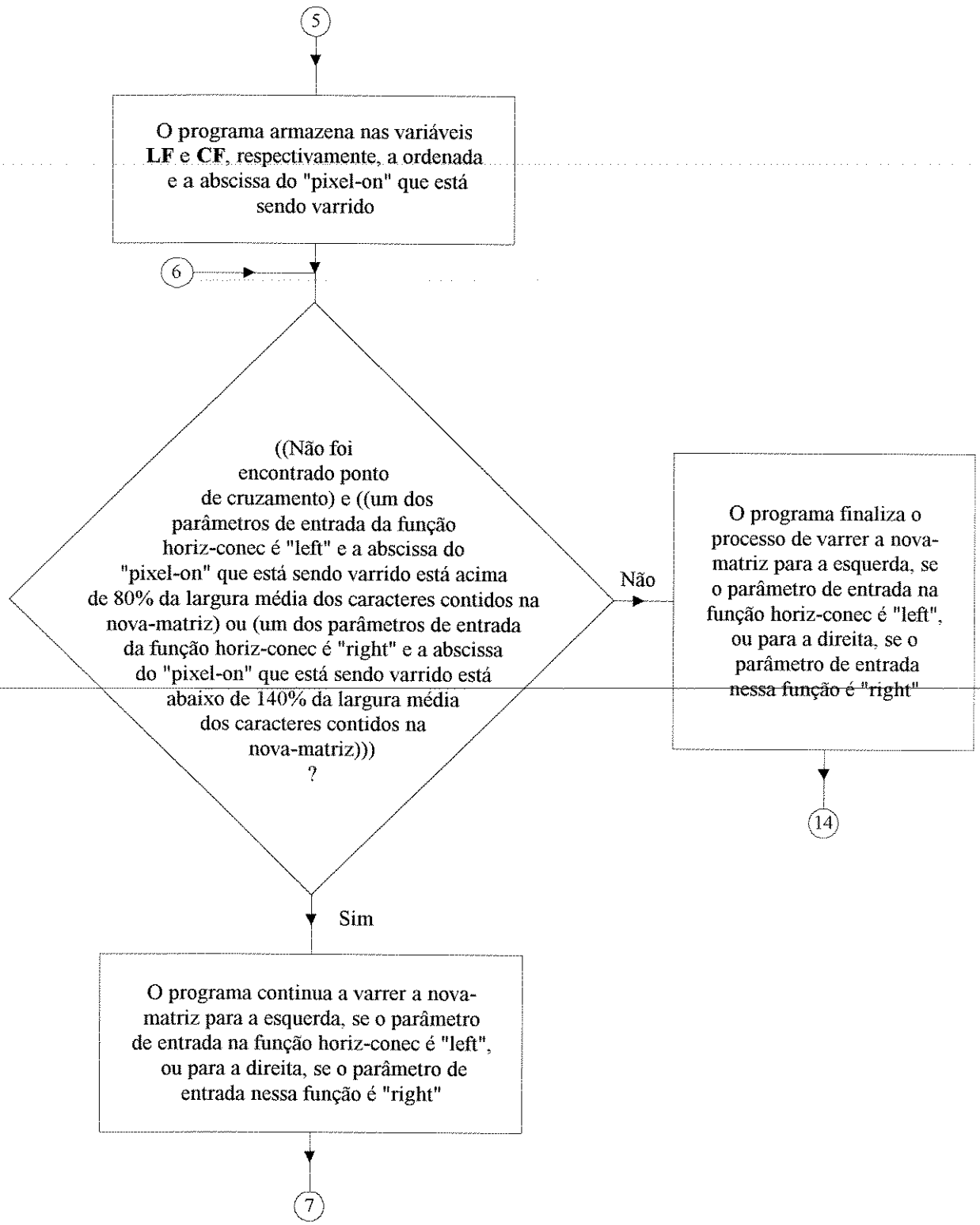


Figura 5.7(f) - Sexta parte do fluxograma da função horiz-conec.

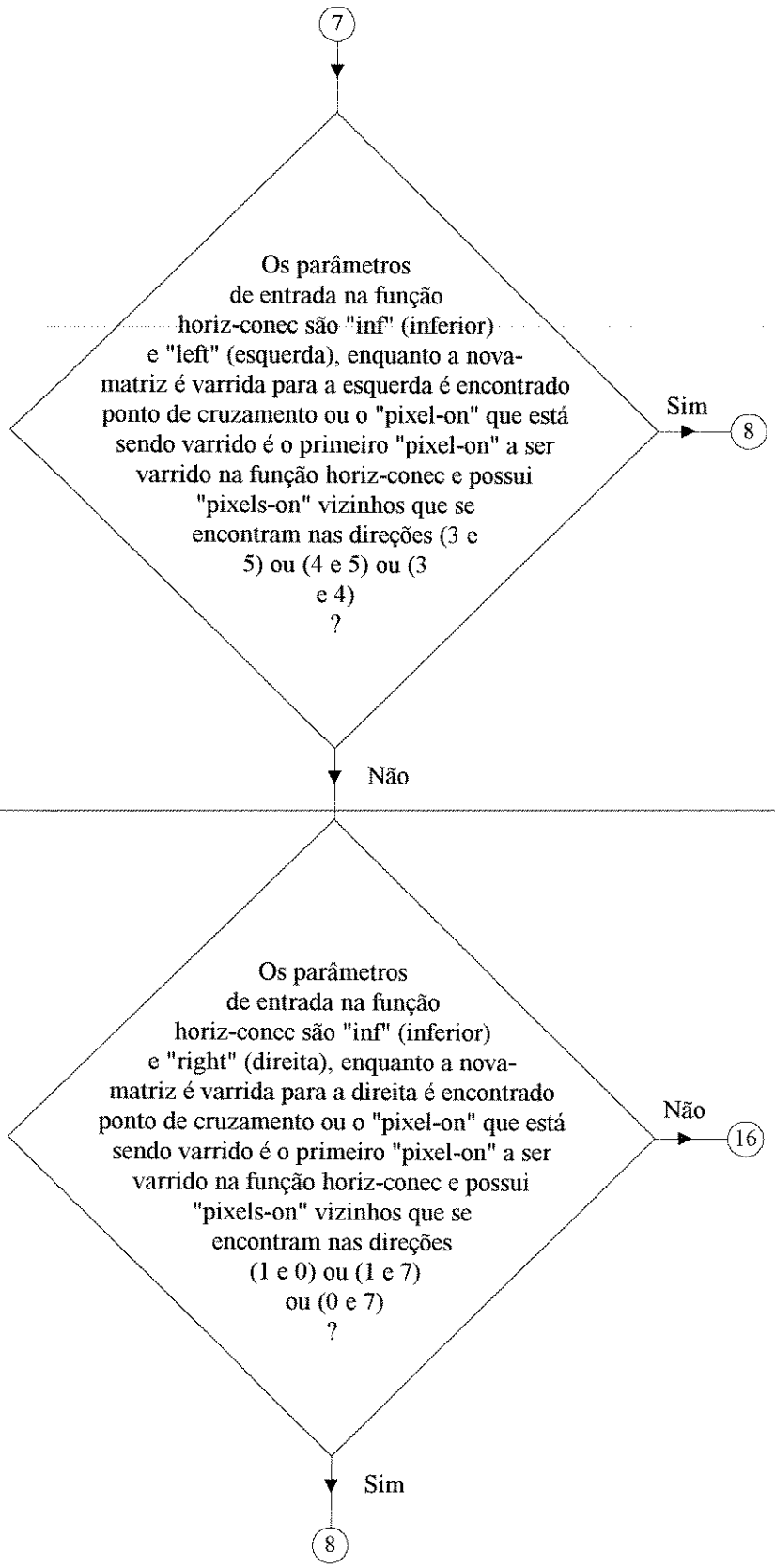


Figura 5.7(g) - Sétima parte do fluxograma da função horiz-conec.

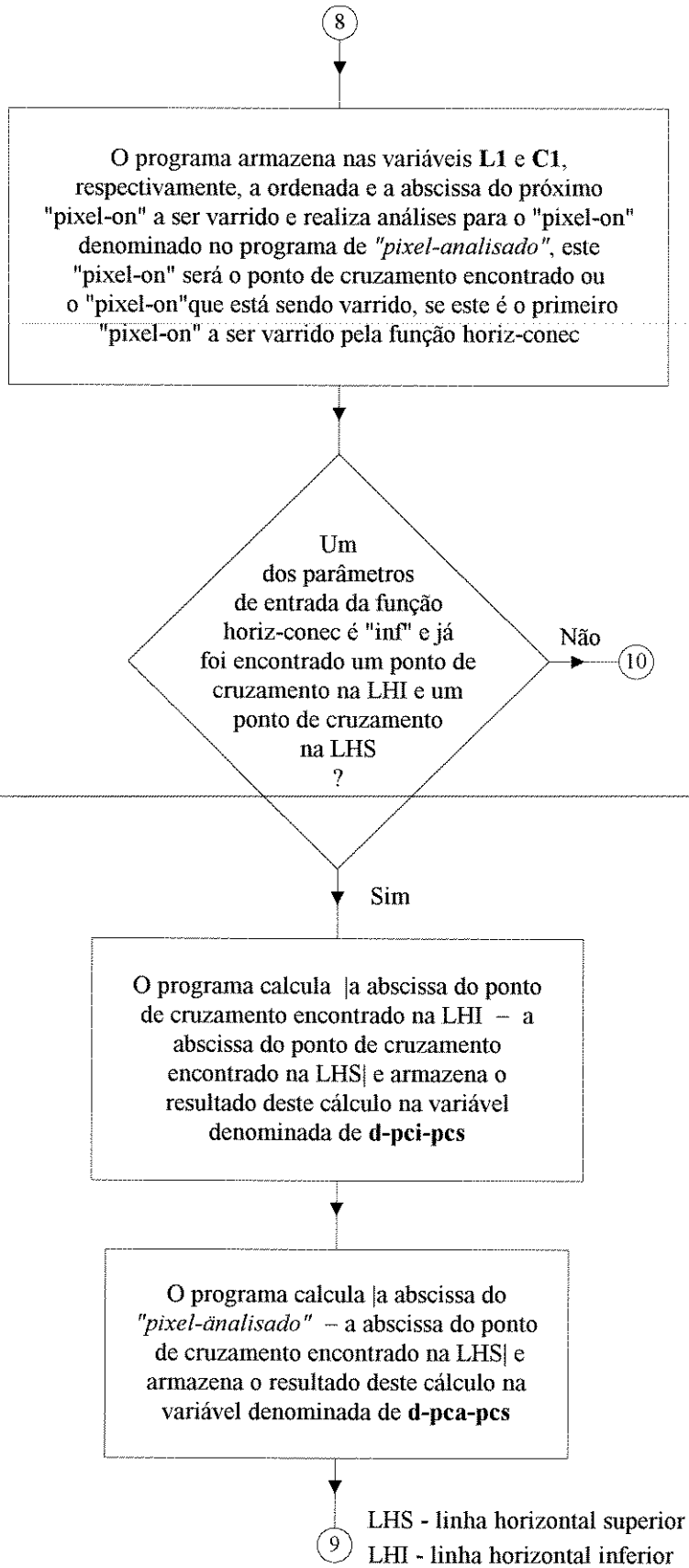


Figura 5.7(h) - Oitava parte do fluxograma da função horiz-conec.

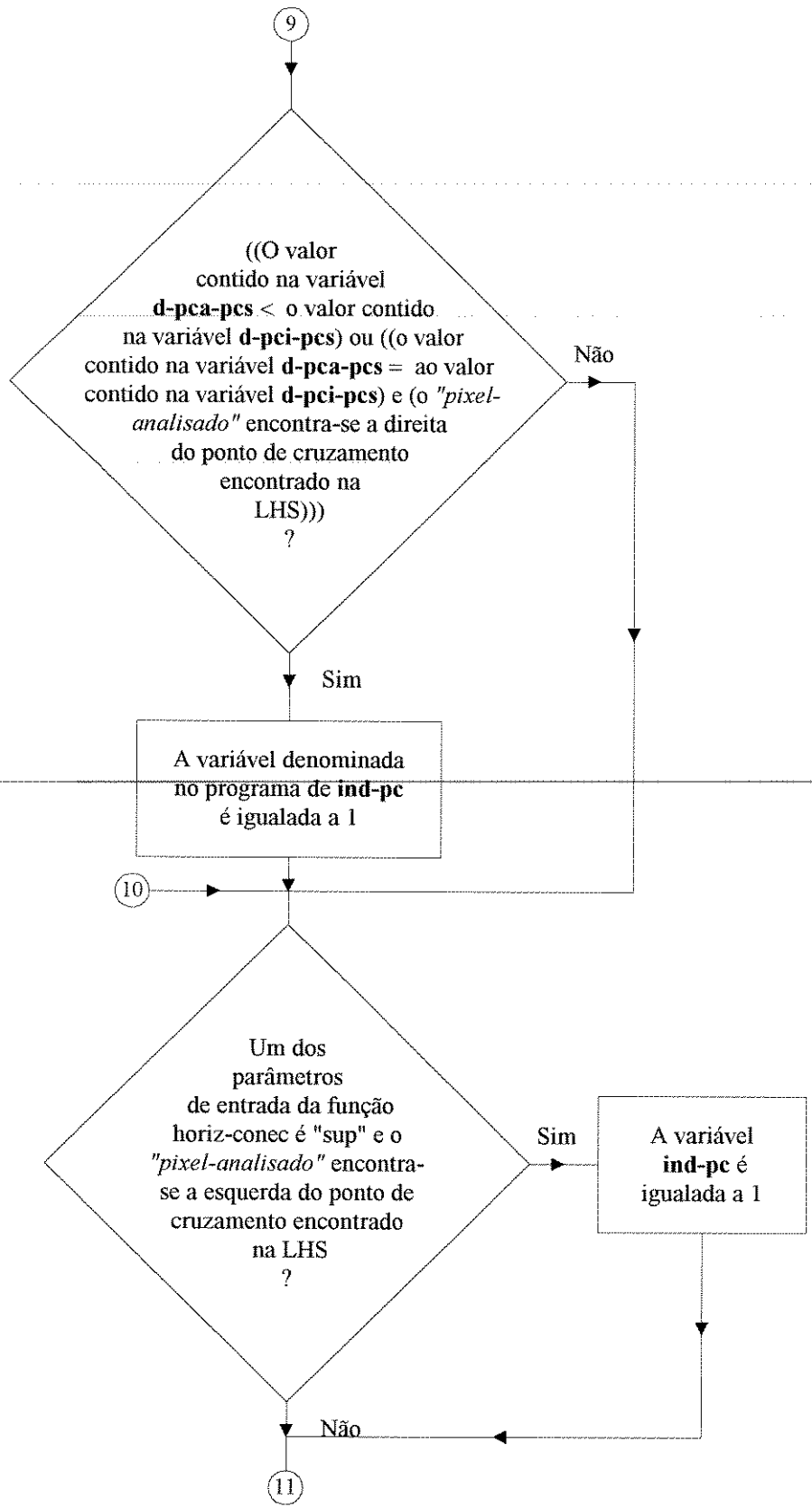


Figura 5.7(i) - Nona parte do fluxograma da função horiz-conec.

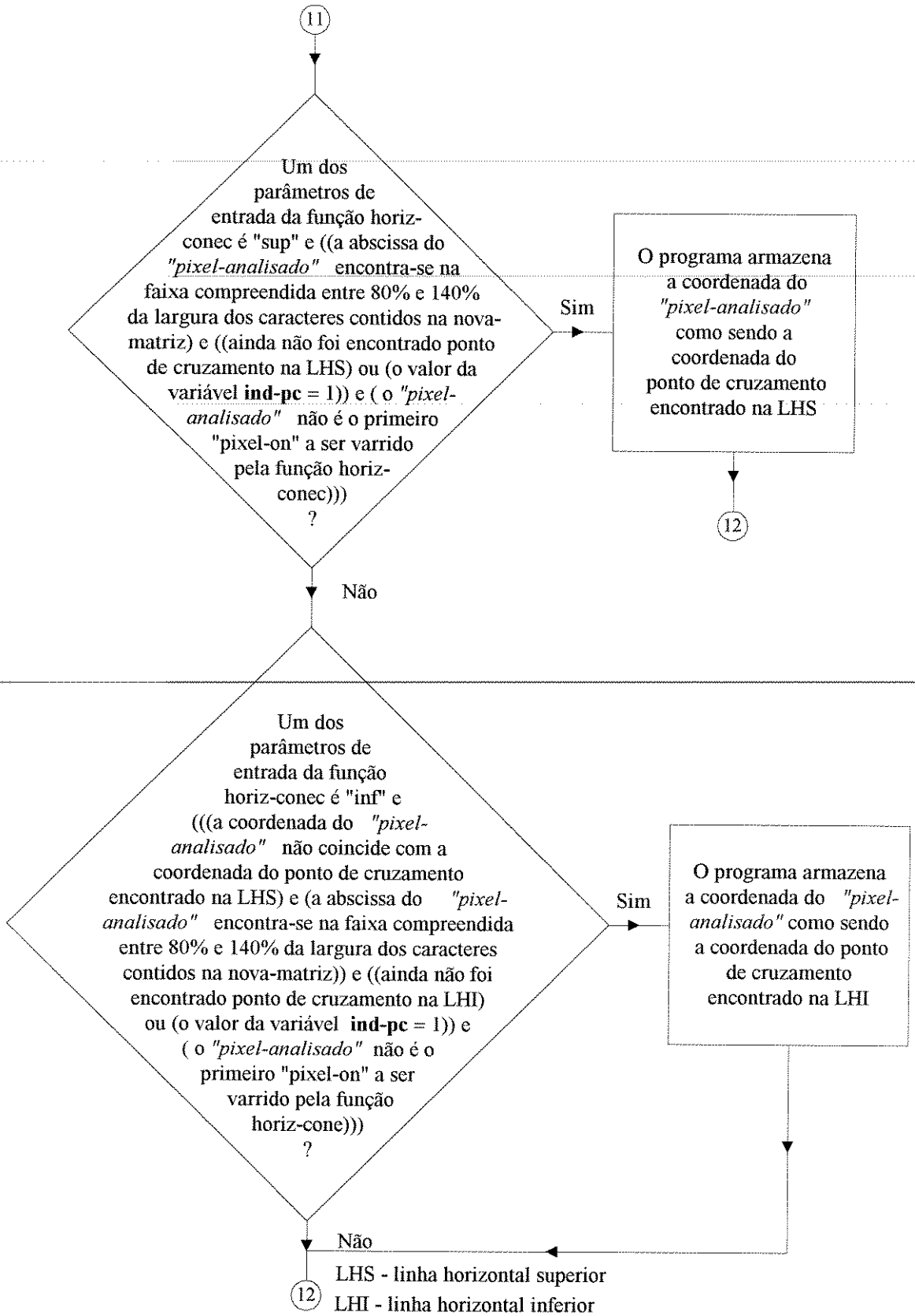


Figura 5.7(j) - Décima parte do fluxograma da função horiz-conec.

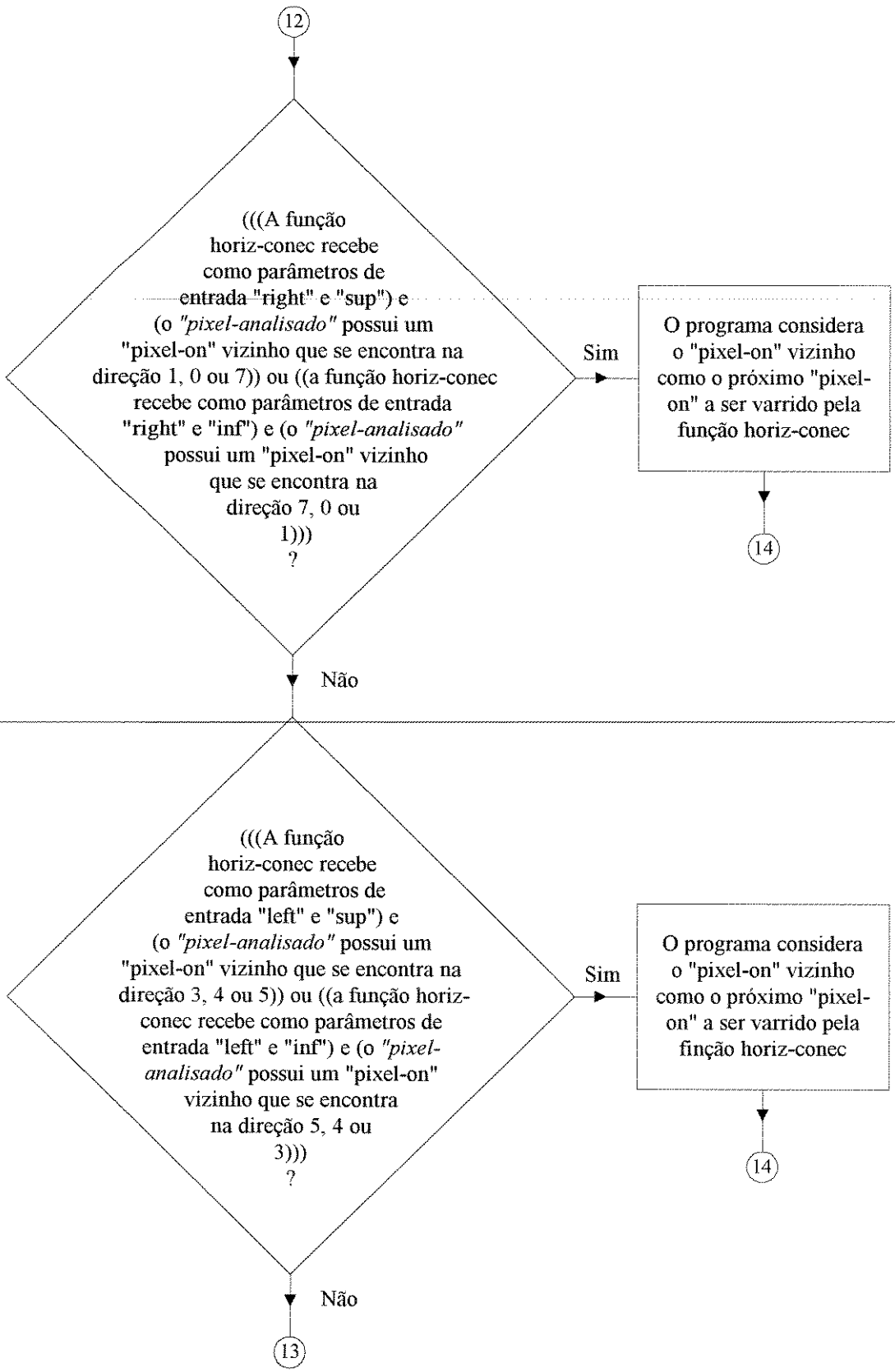


Figura 5.7(k) - Décima primeira parte do fluxograma da função horiz-conec.

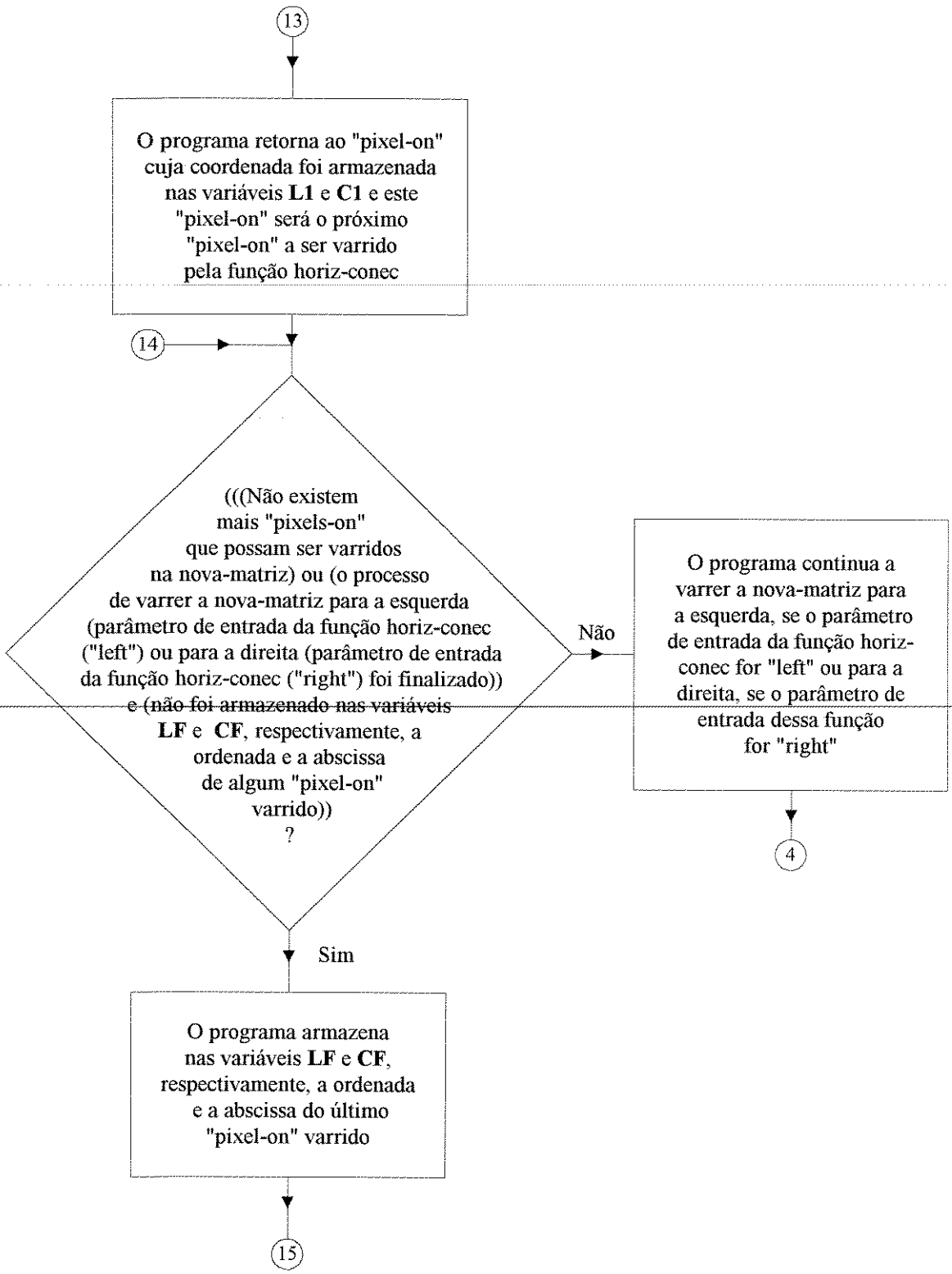


Figura 5.7(l) - Décima segunda parte do fluxograma da função horiz-conec.

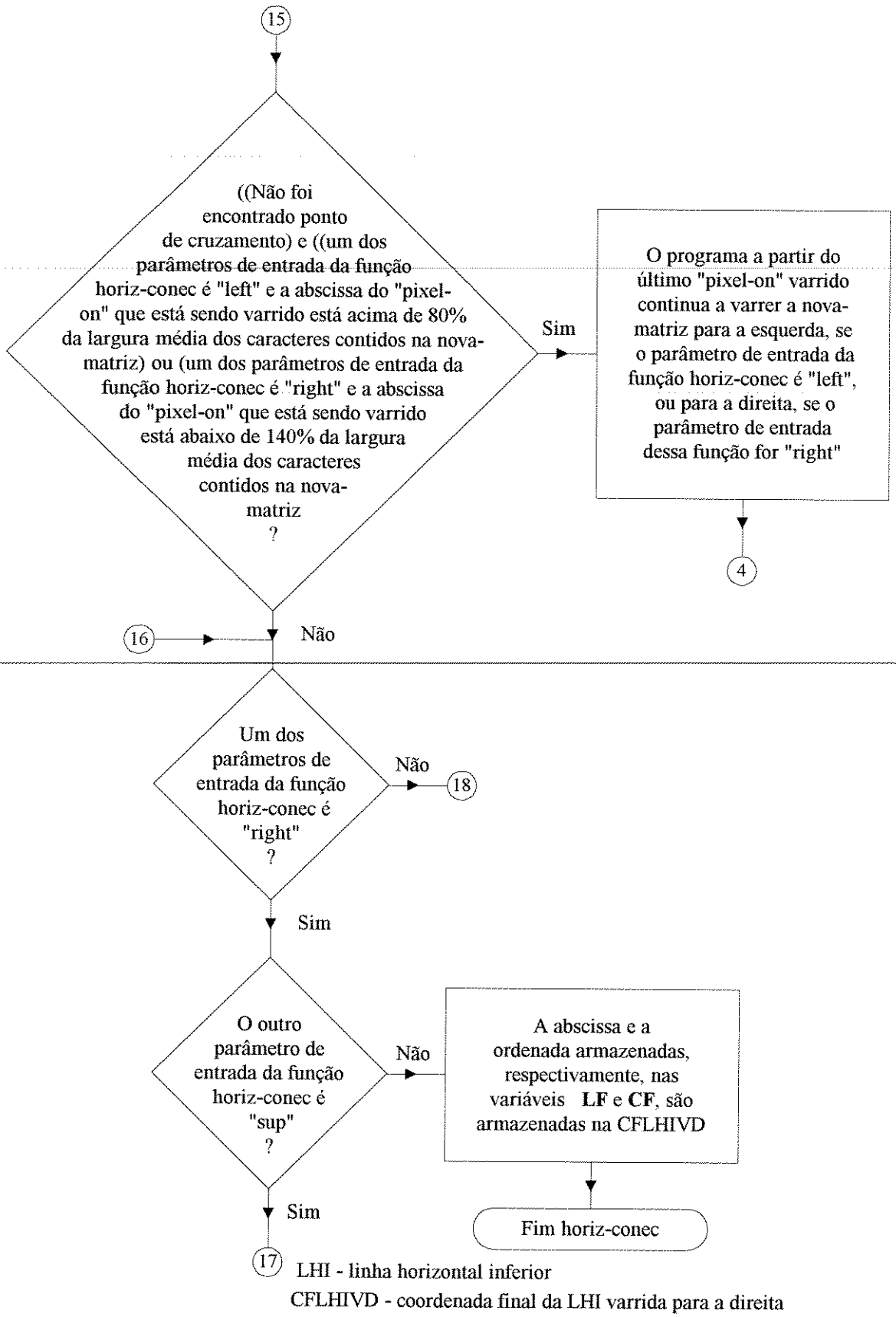
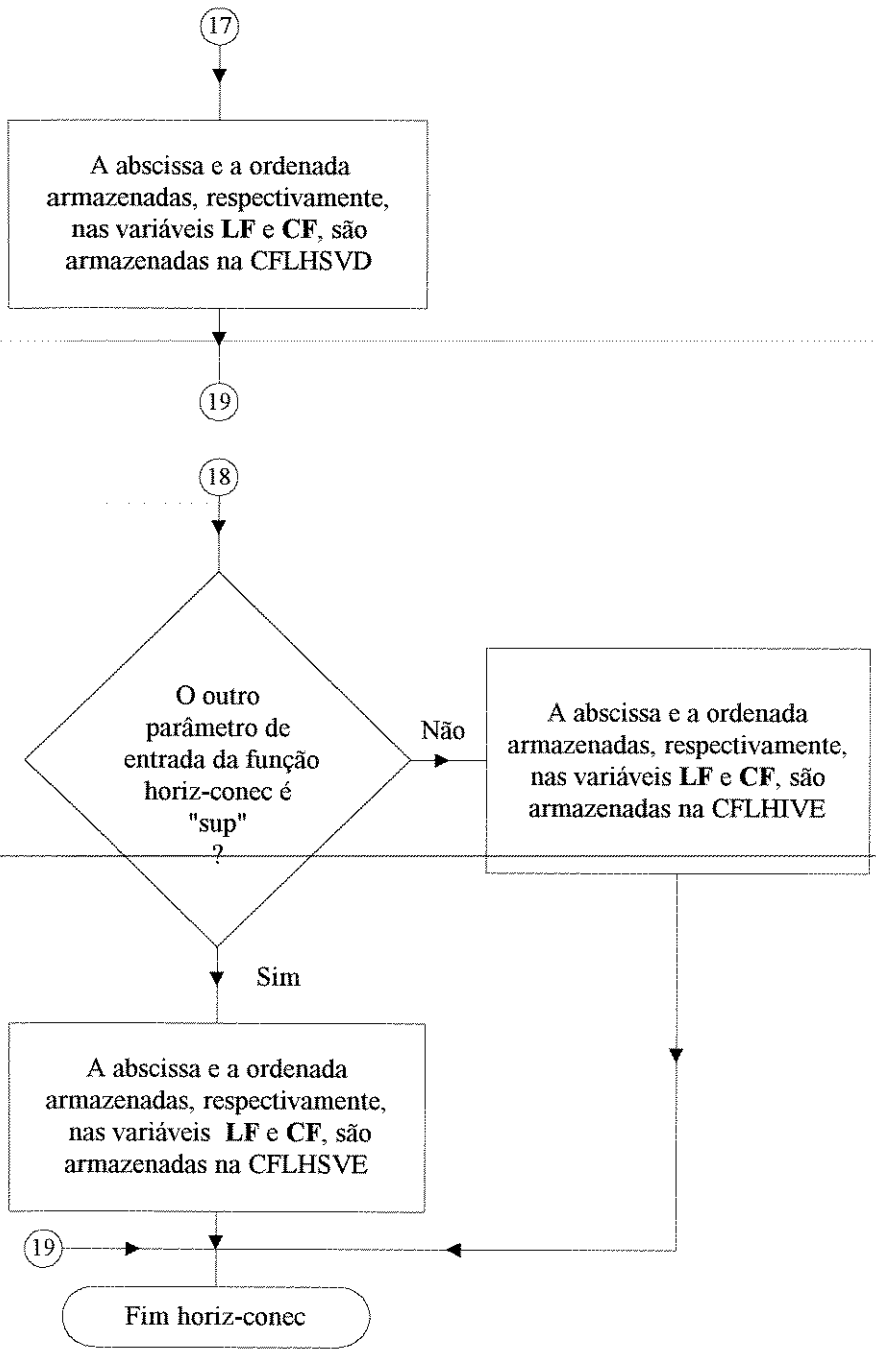


Figura 5.7(m) - Décima terceira parte do fluxograma da função horiz-conec.



LHS - linha horizontal superior
CFLHSVD - coordenada final da LHS varrida para a direita
CFLHSVE - coordenada final da LHS varrida para a esquerda
LHI - linha horizontal inferior
CFLHIVE - coordenada final da LHI varrida para a esquerda

Figura 5.7(n) - Décima quarta parte do fluxograma da função horiz-conec.

Fluxograma da função ver-conec-00

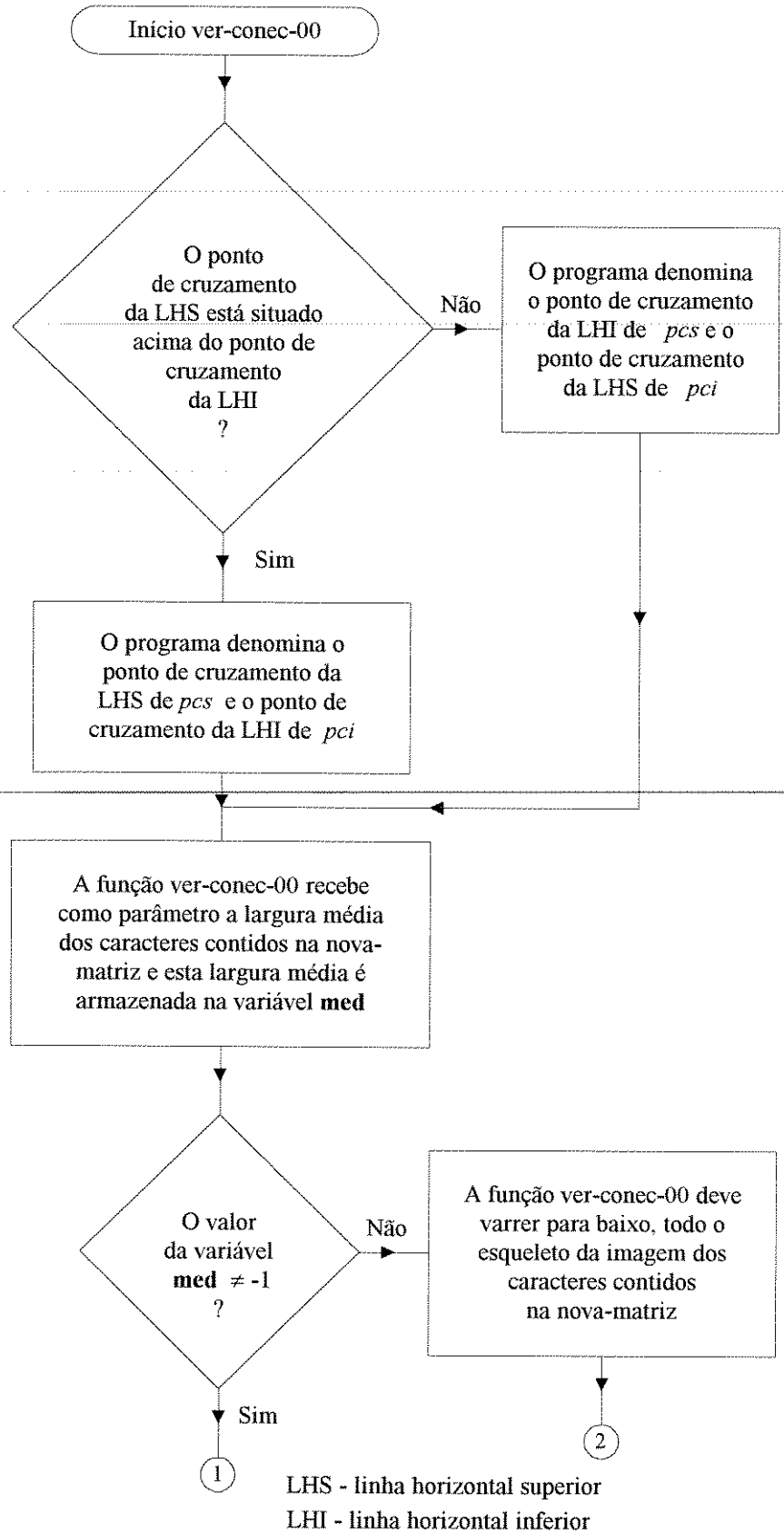


Figura 5.8(a) - Primeira parte do fluxograma da função ver-conec-00.

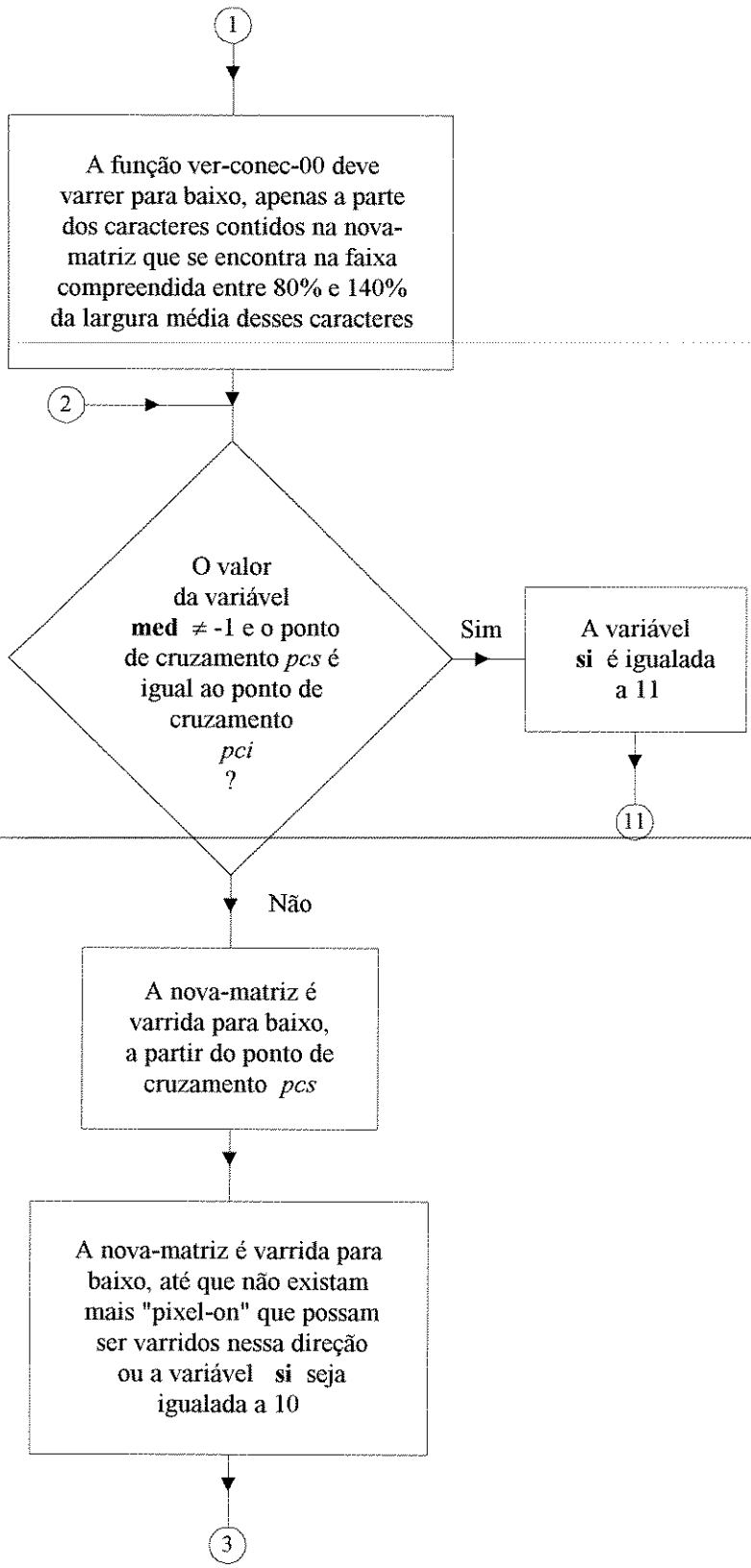


Figura 5.8(b) - Segunda parte do fluxograma da função ver-conec-00.

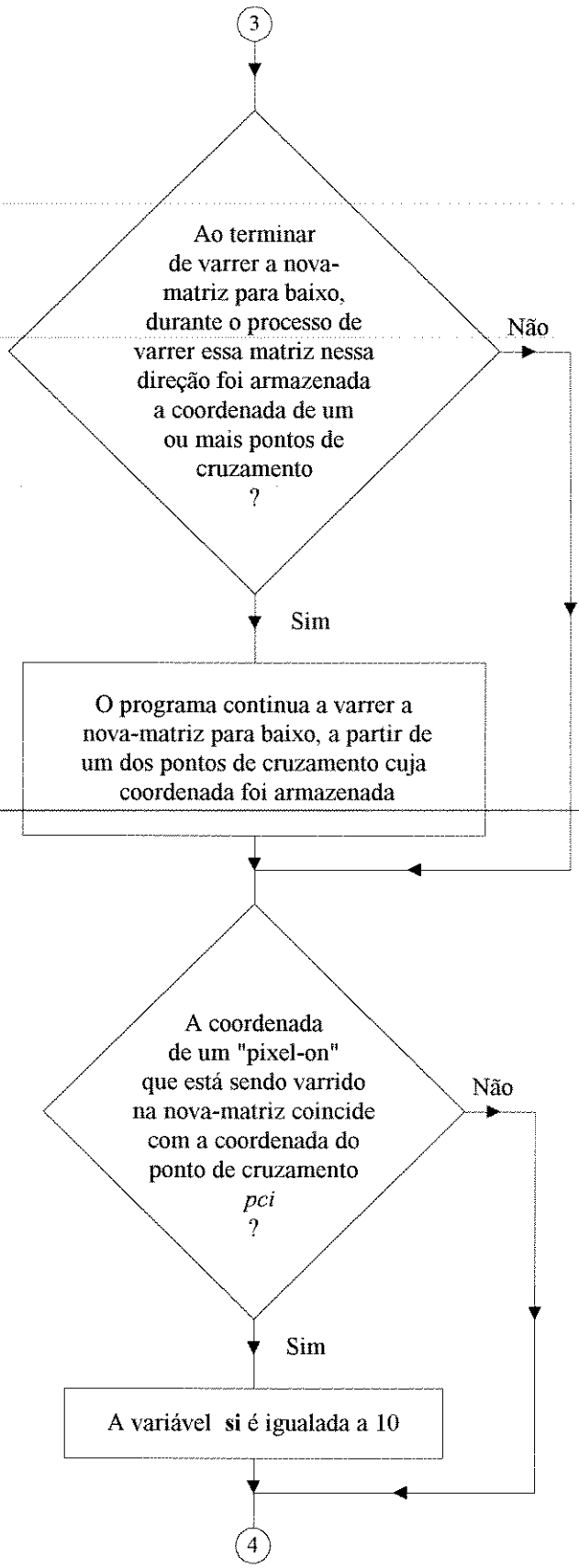


Figura 5.8(c) - Terceira parte do fluxograma da função ver-conec-00.

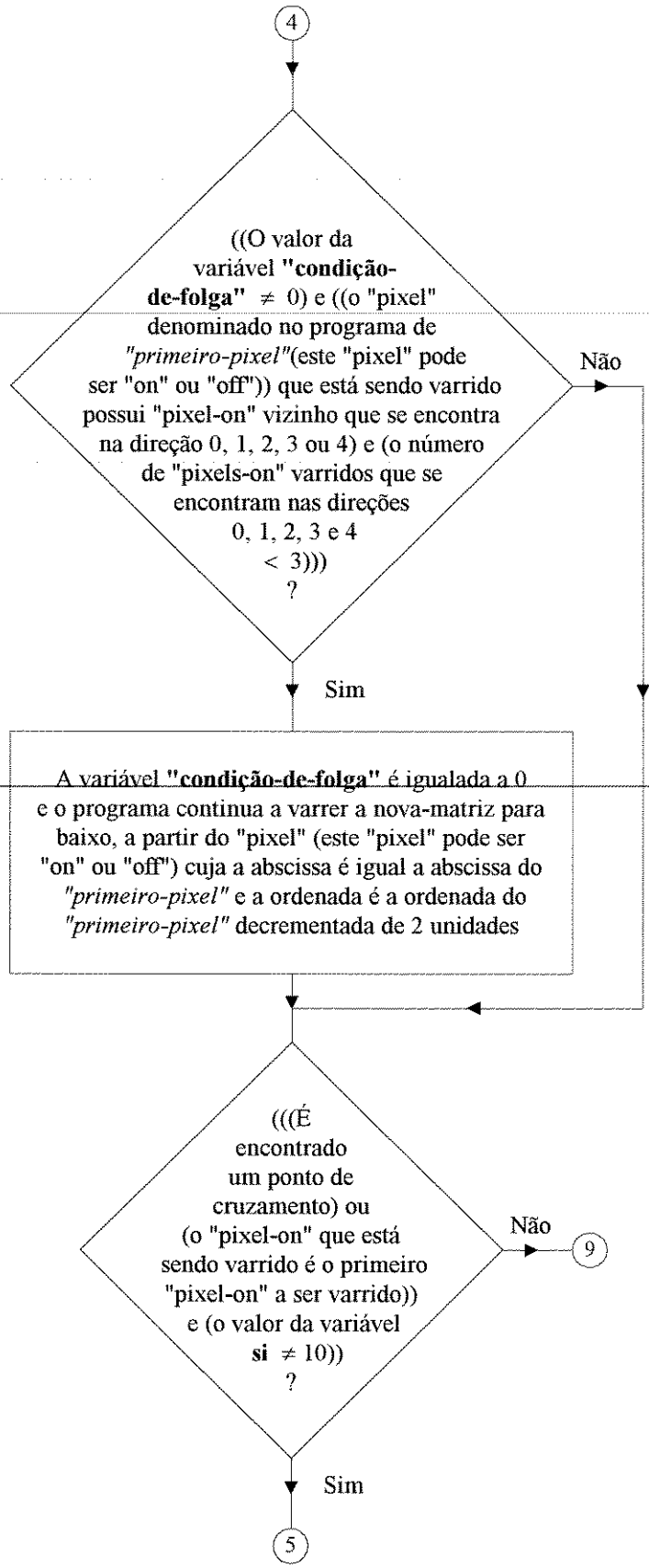


Figura 5.8(d) - Quarta parte do fluxograma da função ver-conec-00.

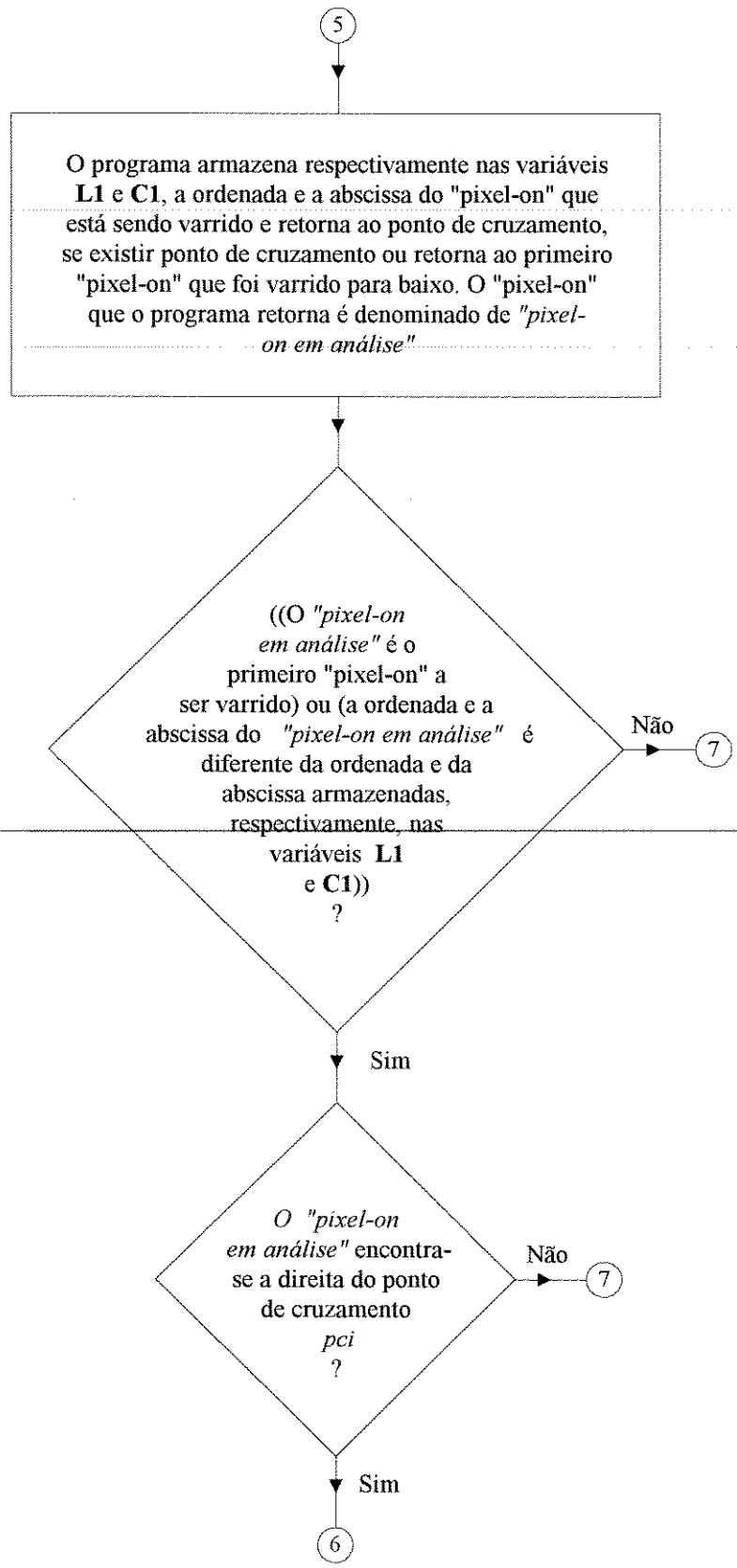


Figura 5.8(e) - Quinta parte do fluxograma da função ver-conec-00.

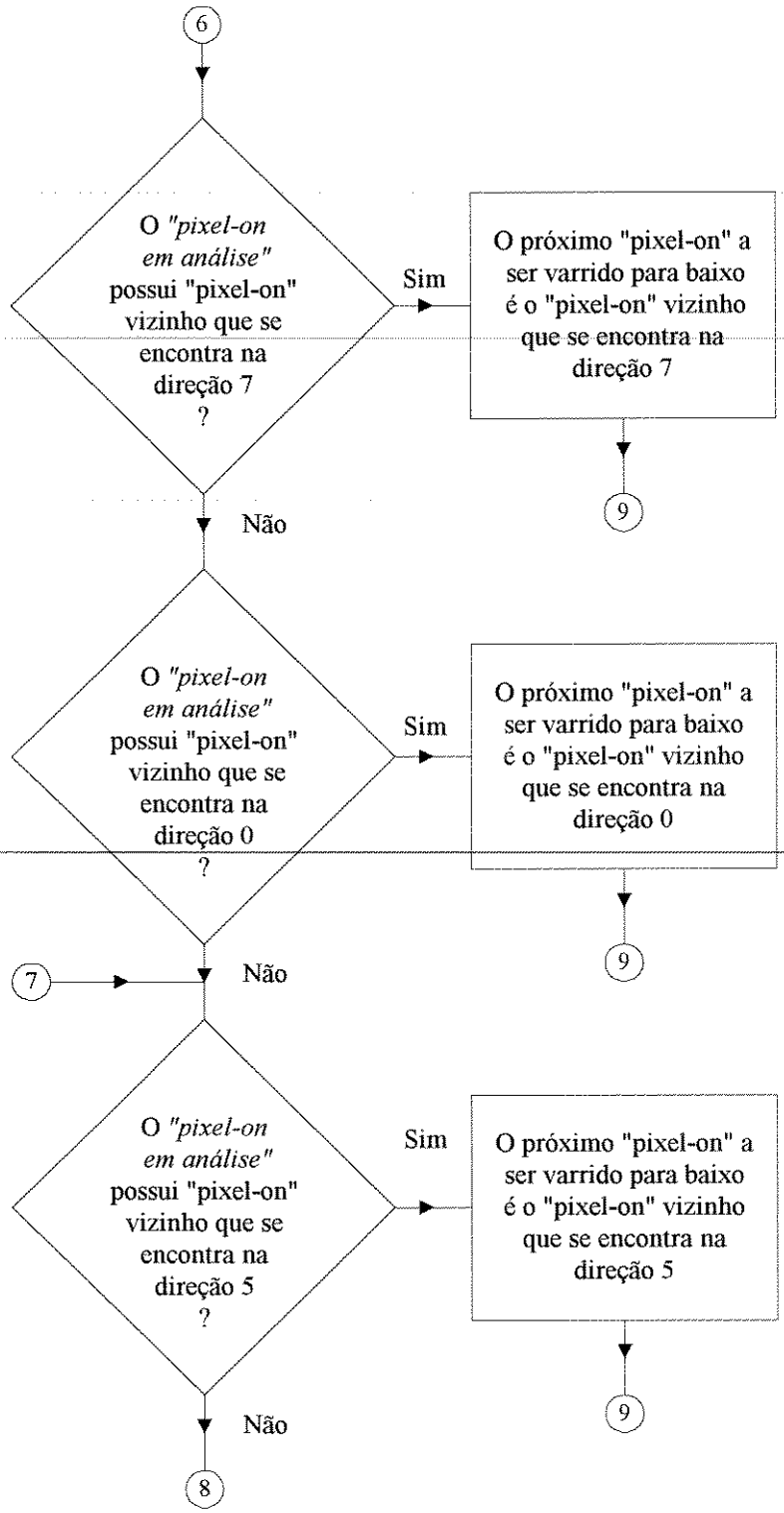


Figura 5.8(f) - Sexta parte do fluxograma da função ver-conec-00.

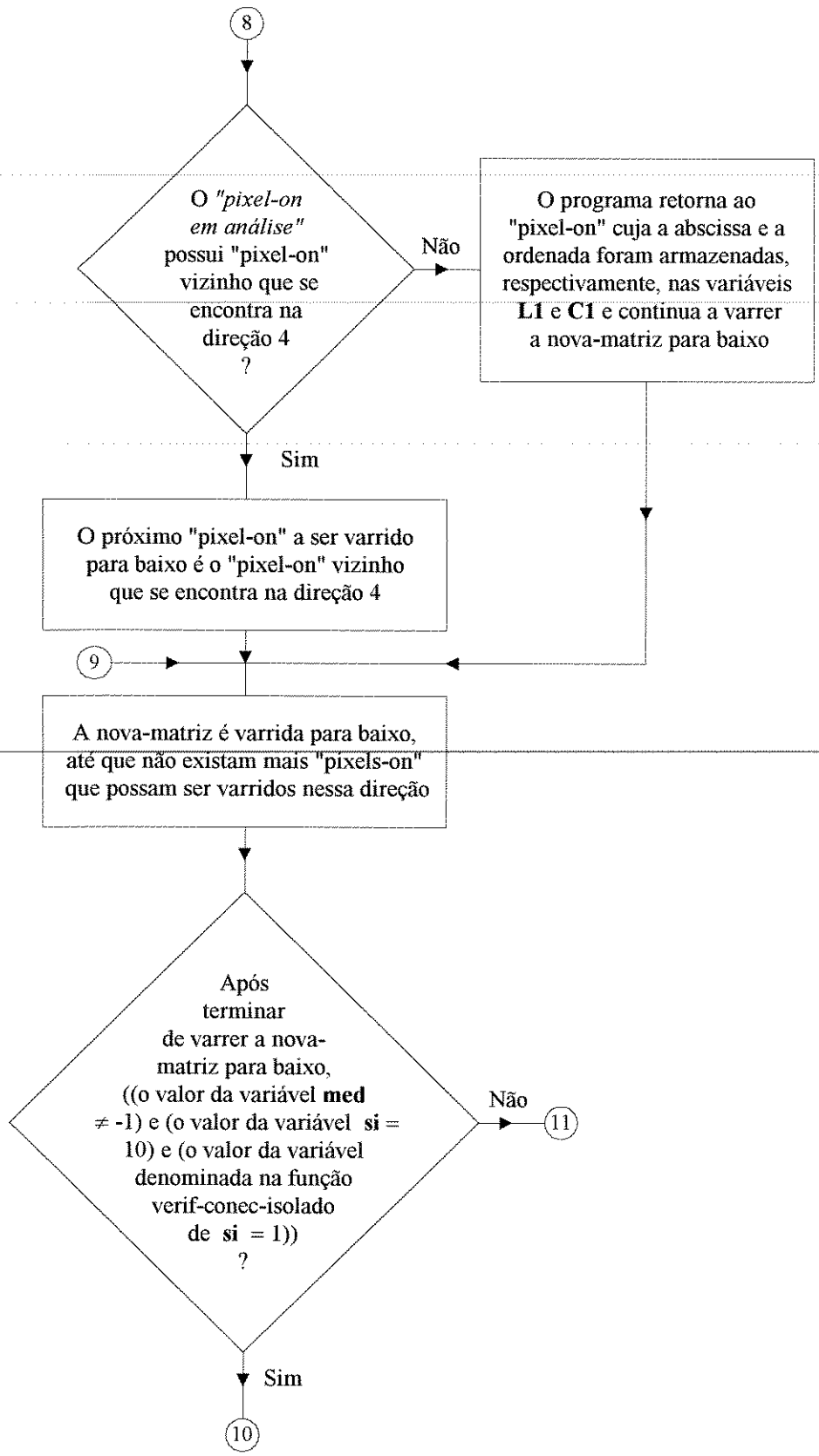
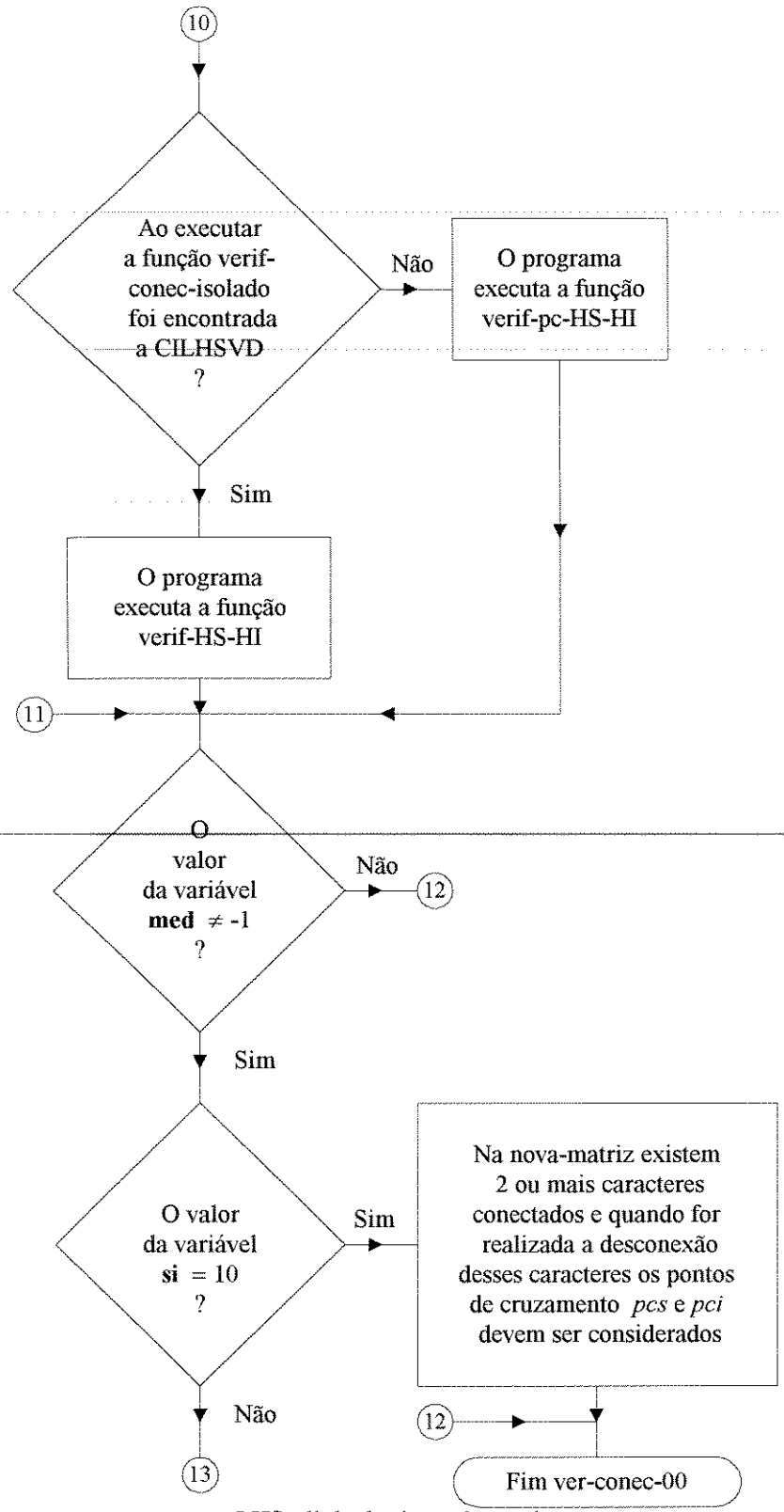


Figura 5.8(g) - Sétima parte do fluxograma da função ver-conec-00.



LHS - linha horizontal superior

CILHSVD - coordenada inicial da LHS varrida para a direita

Figura 5.8(h) - Oitava parte do fluxograma da função ver-conec-00.

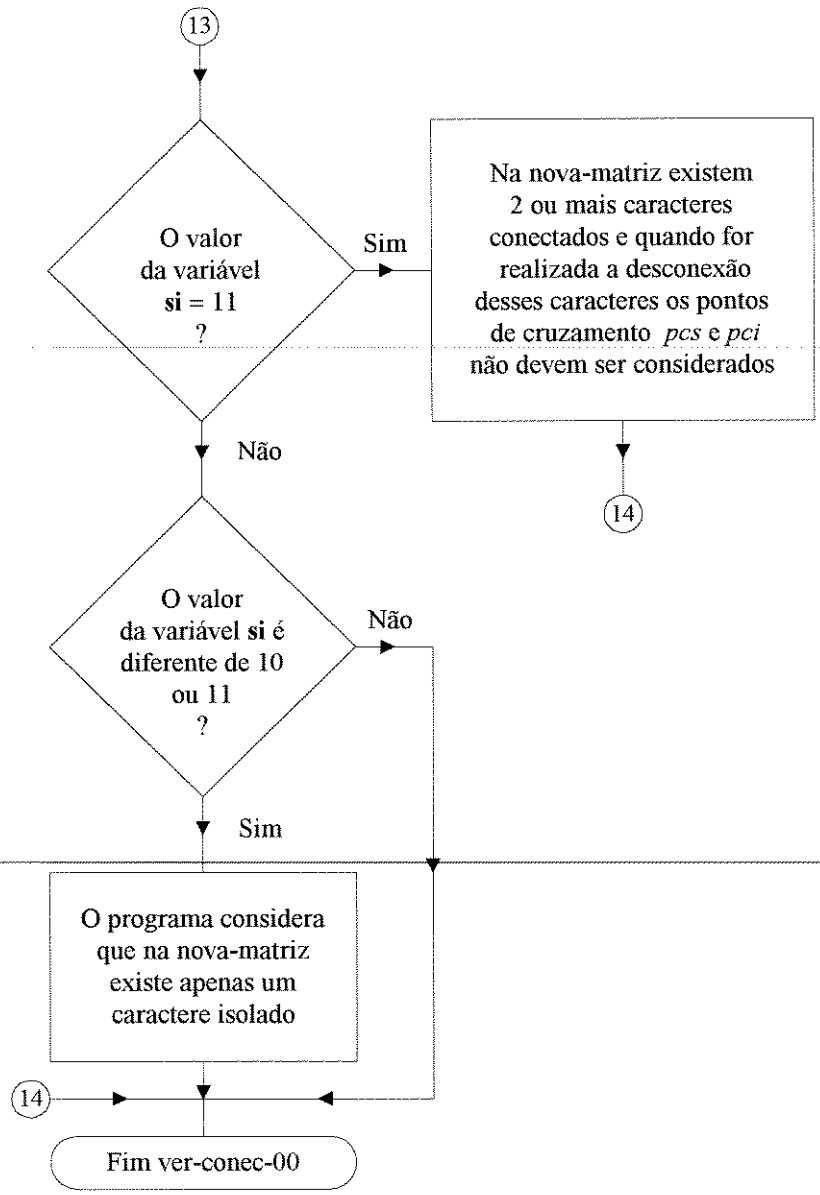


Figura 5.8(i) - Nona parte do fluxograma da função ver-conec-00.

Fluxograma da função verif-HS-HI

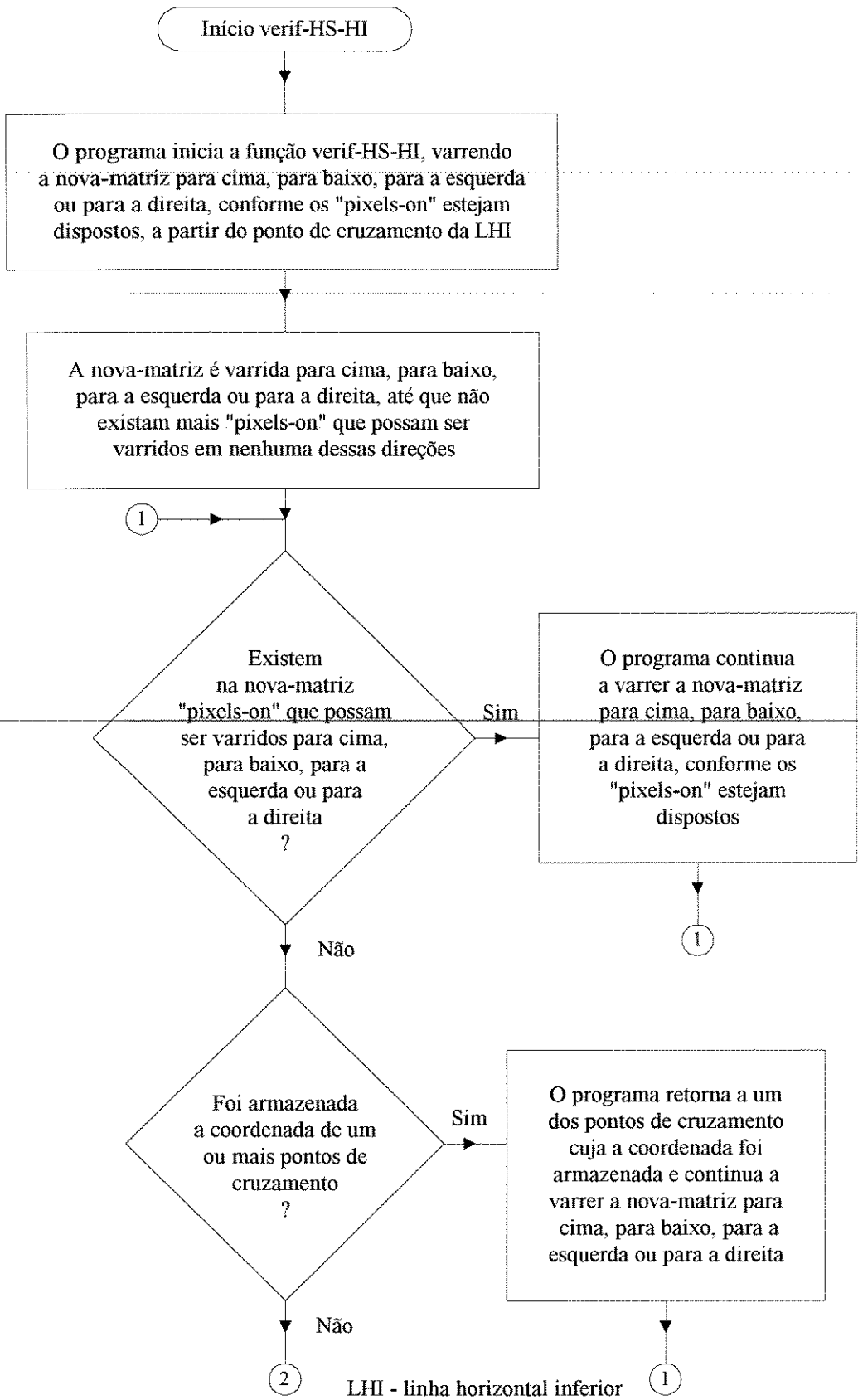
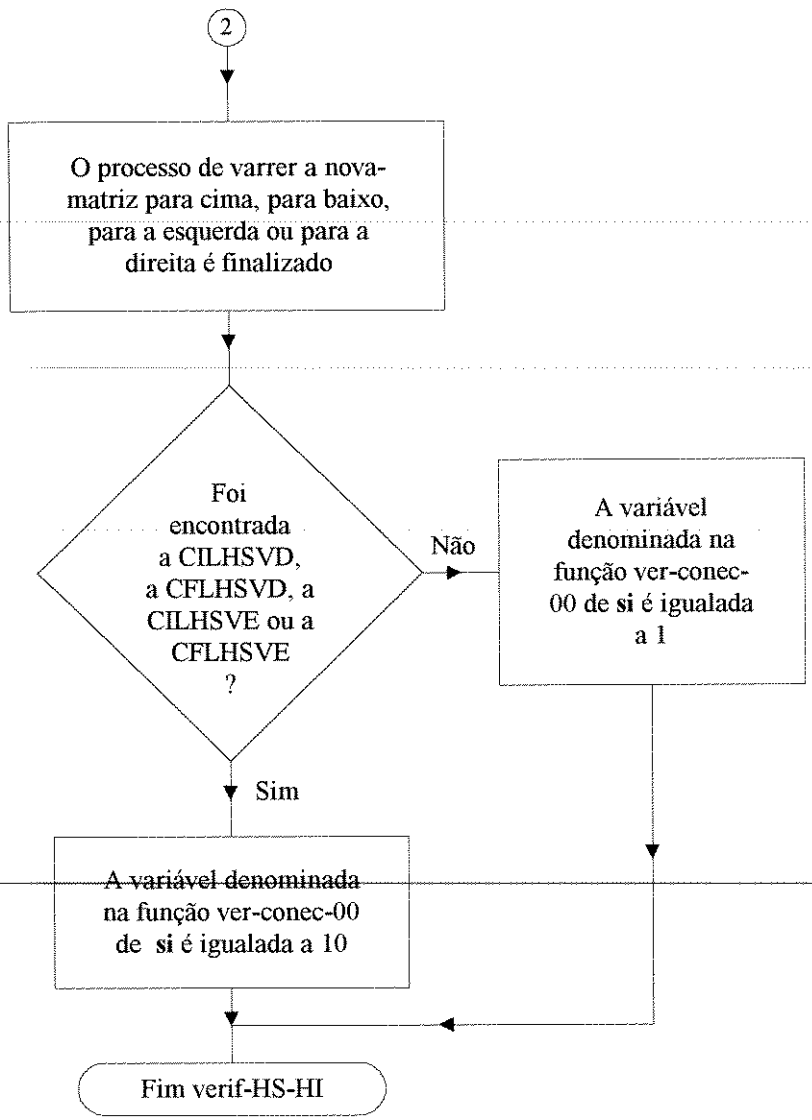


Figura 5.9(a) - Primeira parte do fluxograma da função verif-HS-HI.



LHS - linha horizontal superior
CILHSVD - coordenada inicial da LHS varrida para a direita
CFLHSVD - coordenada final da LHS varrida para a direita
CILHSVE - coordenada inicial da LHS varrida para a esquerda
CFLHSVE - coordenada final da LHS varrida para a esquerda

Figura 5.9(b) - Segunda parte do fluxograma da função verif-HS-HI.

Fluxograma da função verif-pc-HS-HI

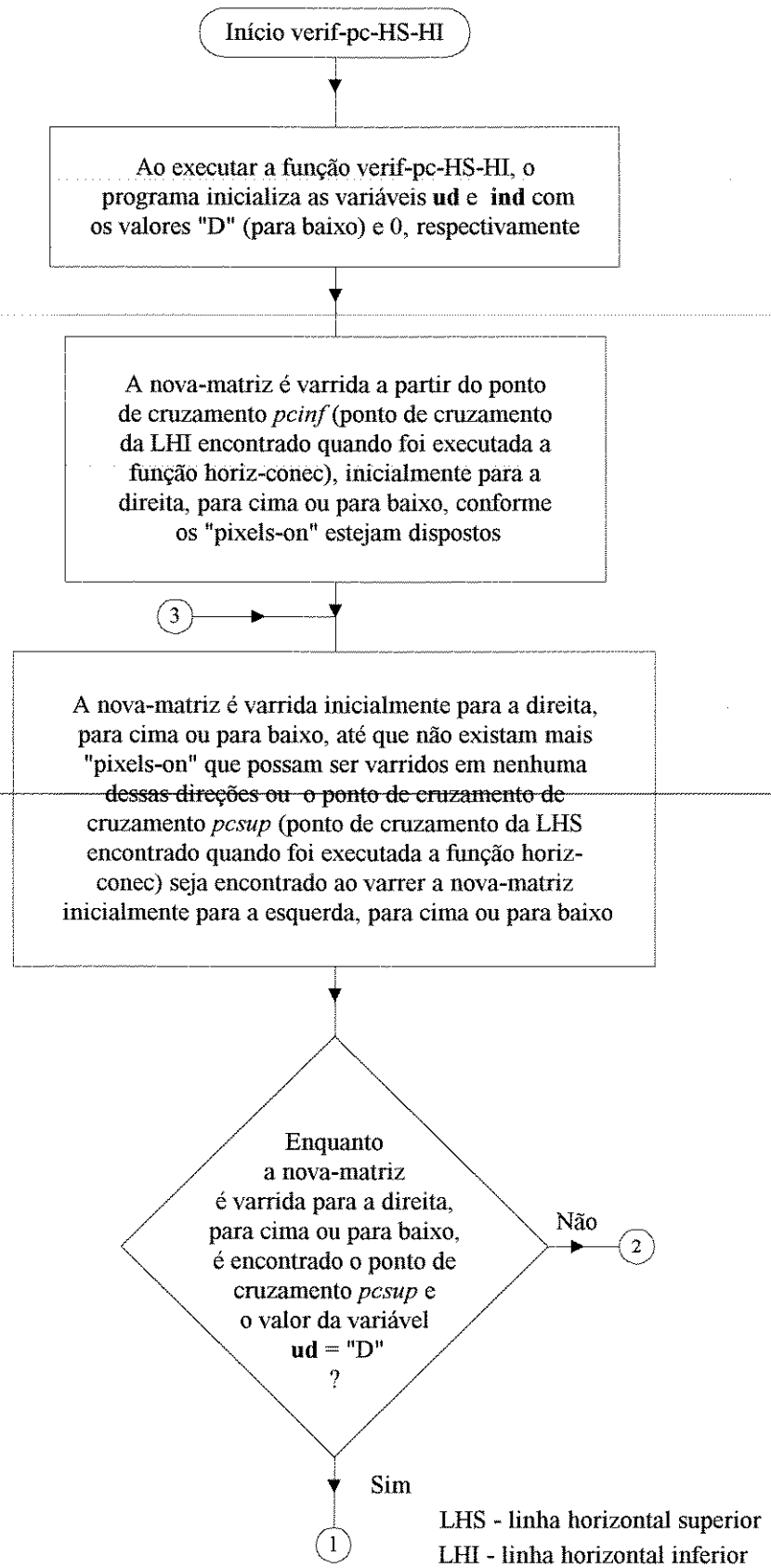


Figura 5.10(a) - Primeira parte do fluxograma da função verif-pc-HS-HI.

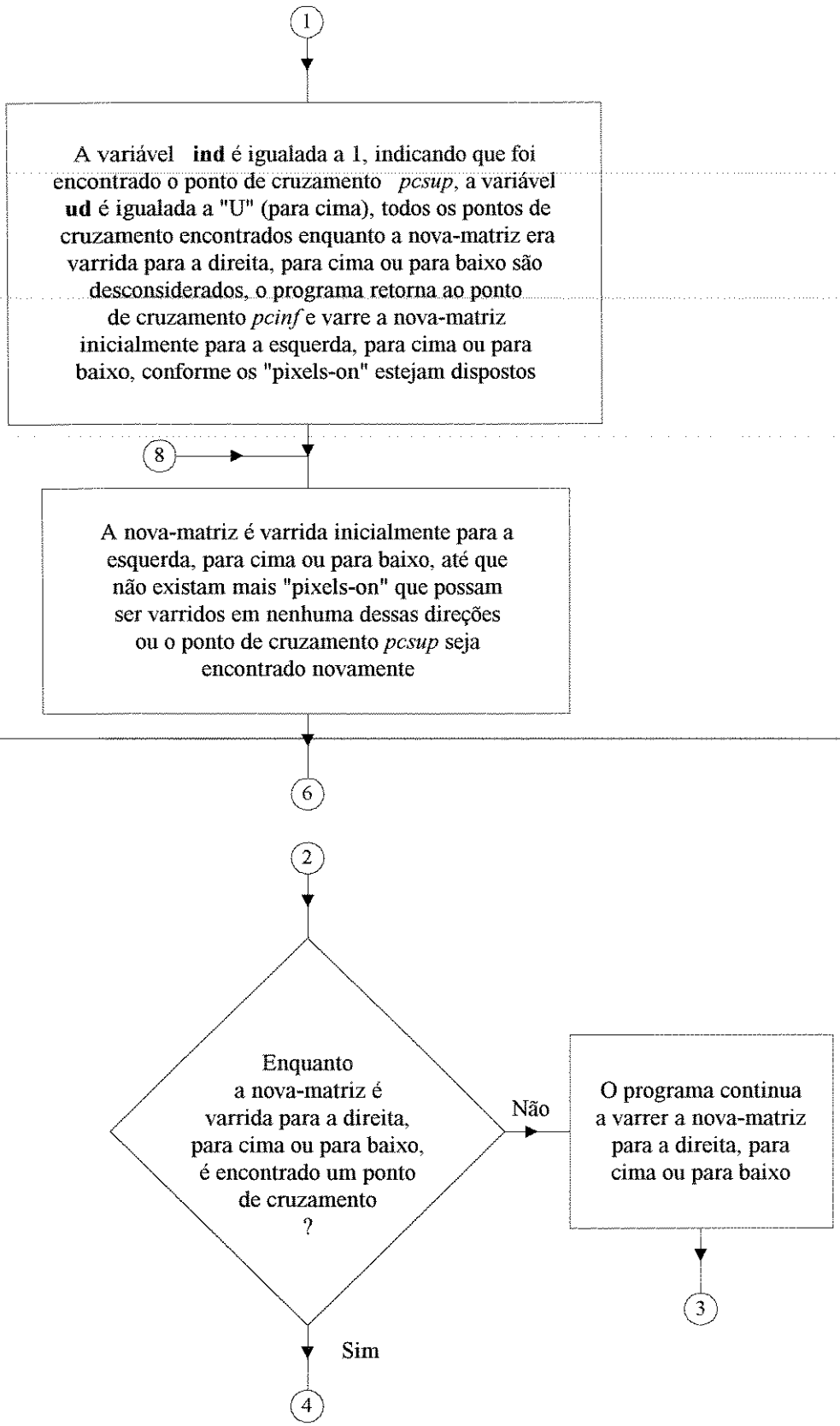


Figura 5.10(b) - Segunda parte do fluxograma da função verif-pc-HS-HI.

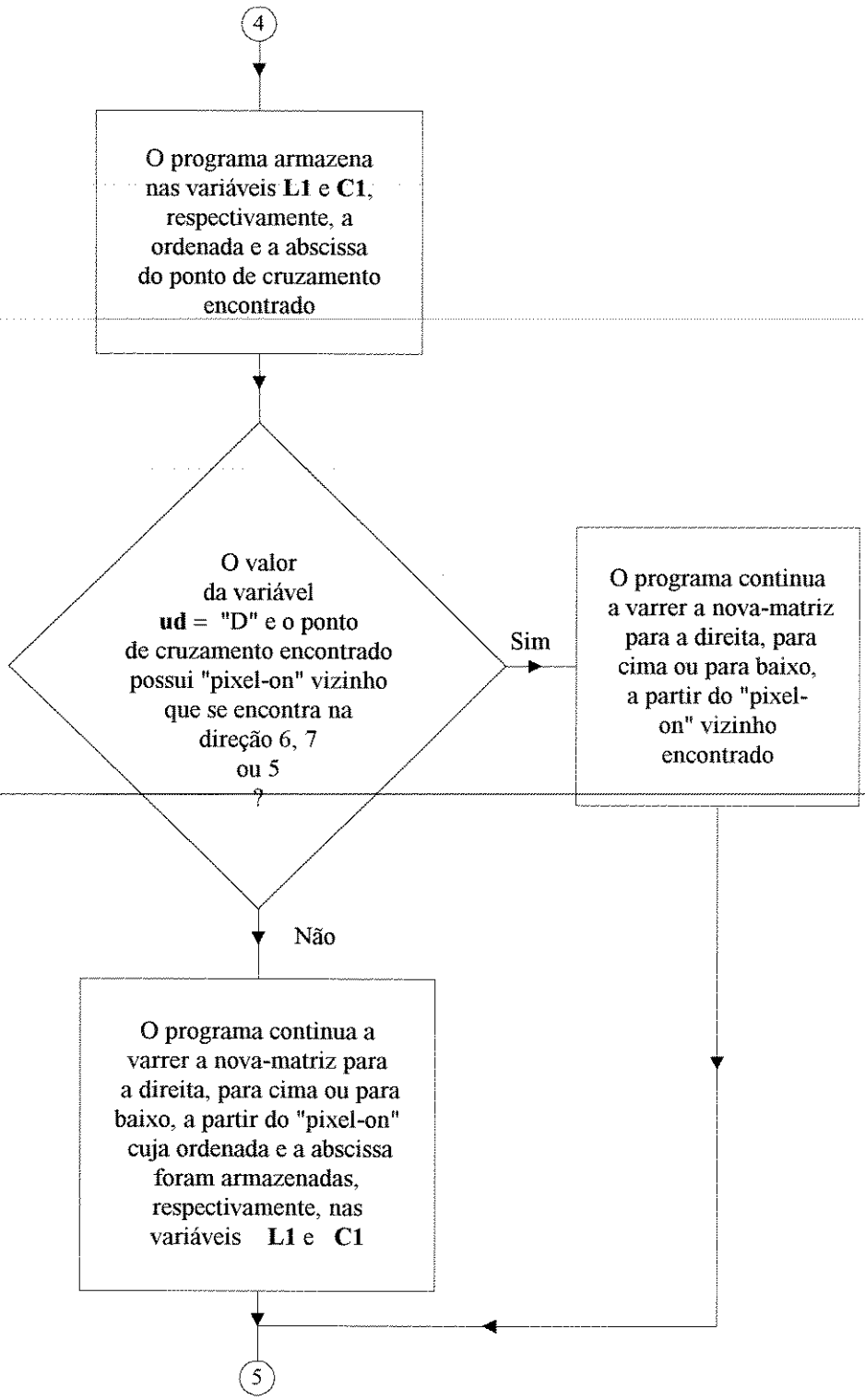


Figura 5.10(c) - Terceira parte do fluxograma da função verif-pc-HS-HI.

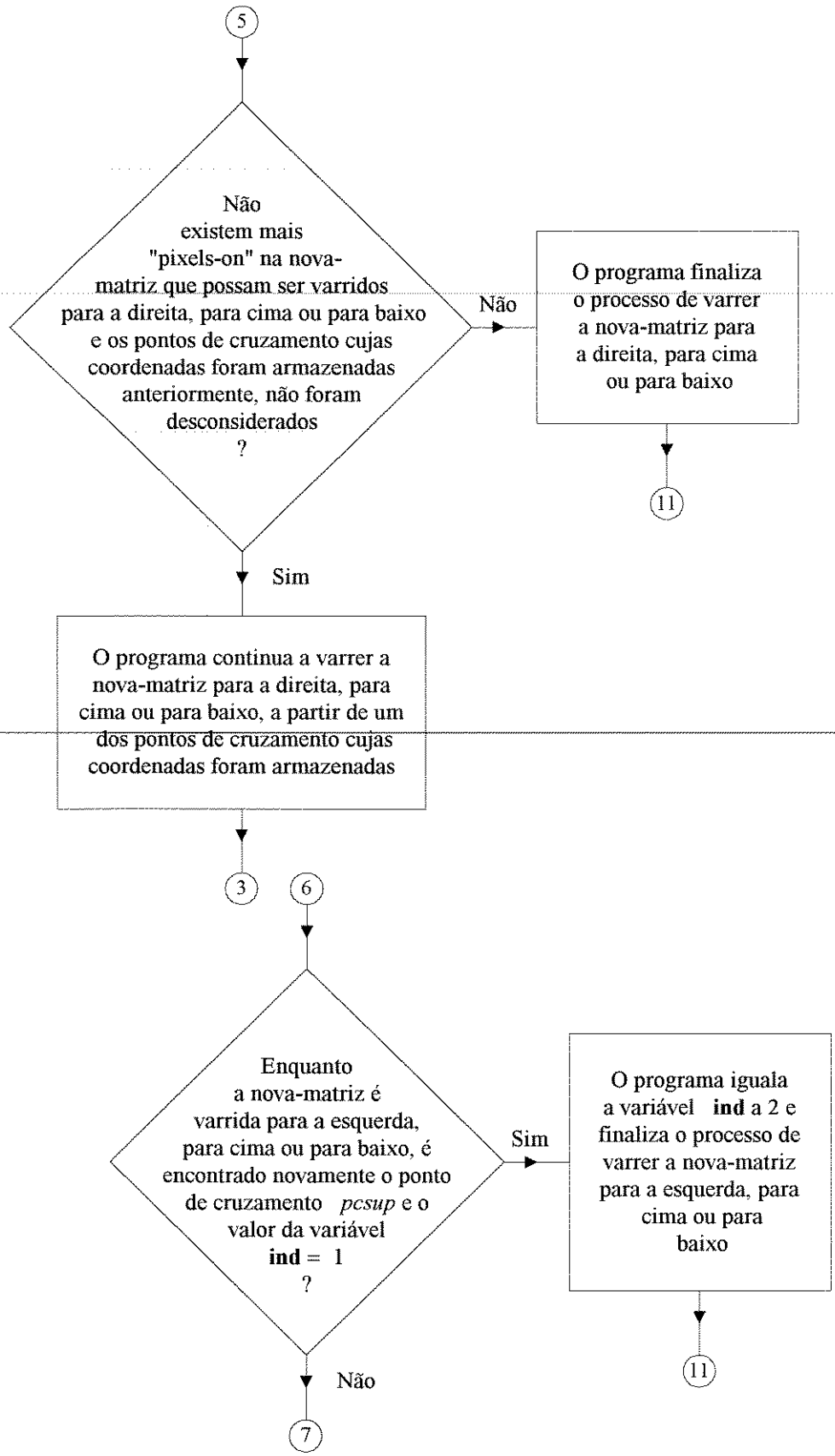


Figura 5.10(d) - Quarta parte do fluxograma da função verif-pc-HS-HI.

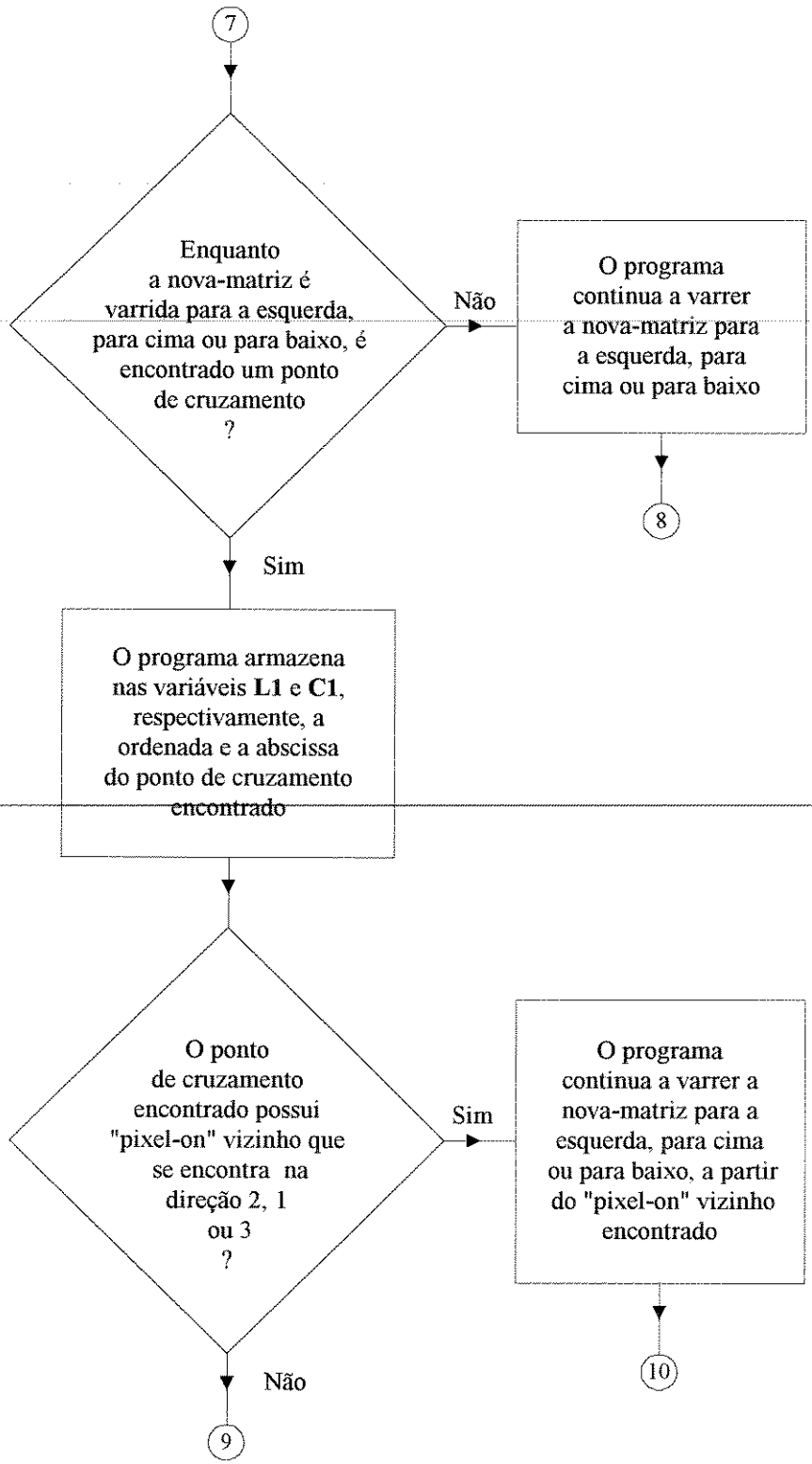


Figura 5.10(e) - Quinta parte do fluxograma da função verif-pc-HS-HI.

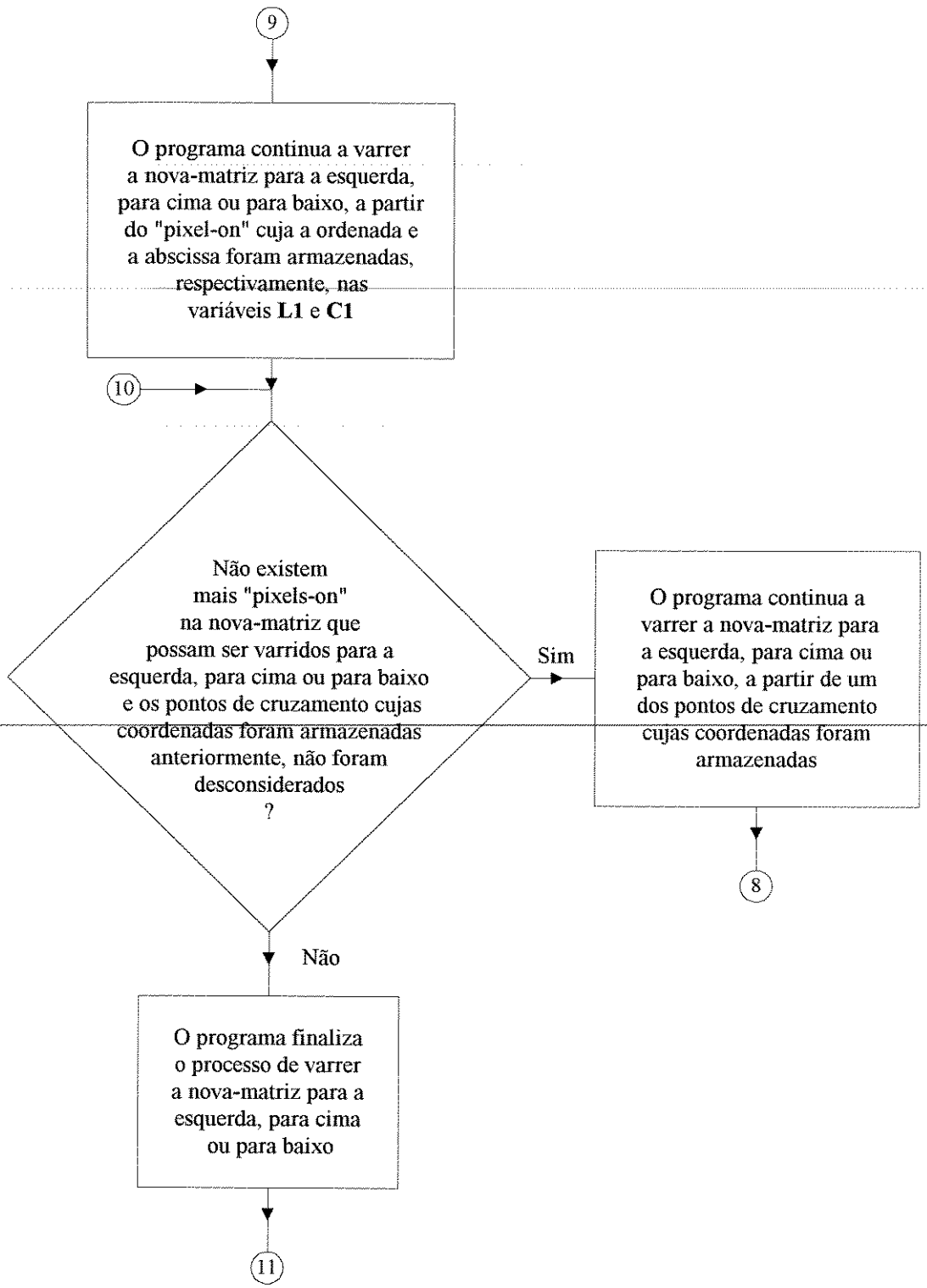


Figura 5.10(f) - Sexta parte do fluxograma da função verif-pc-HS-HI.

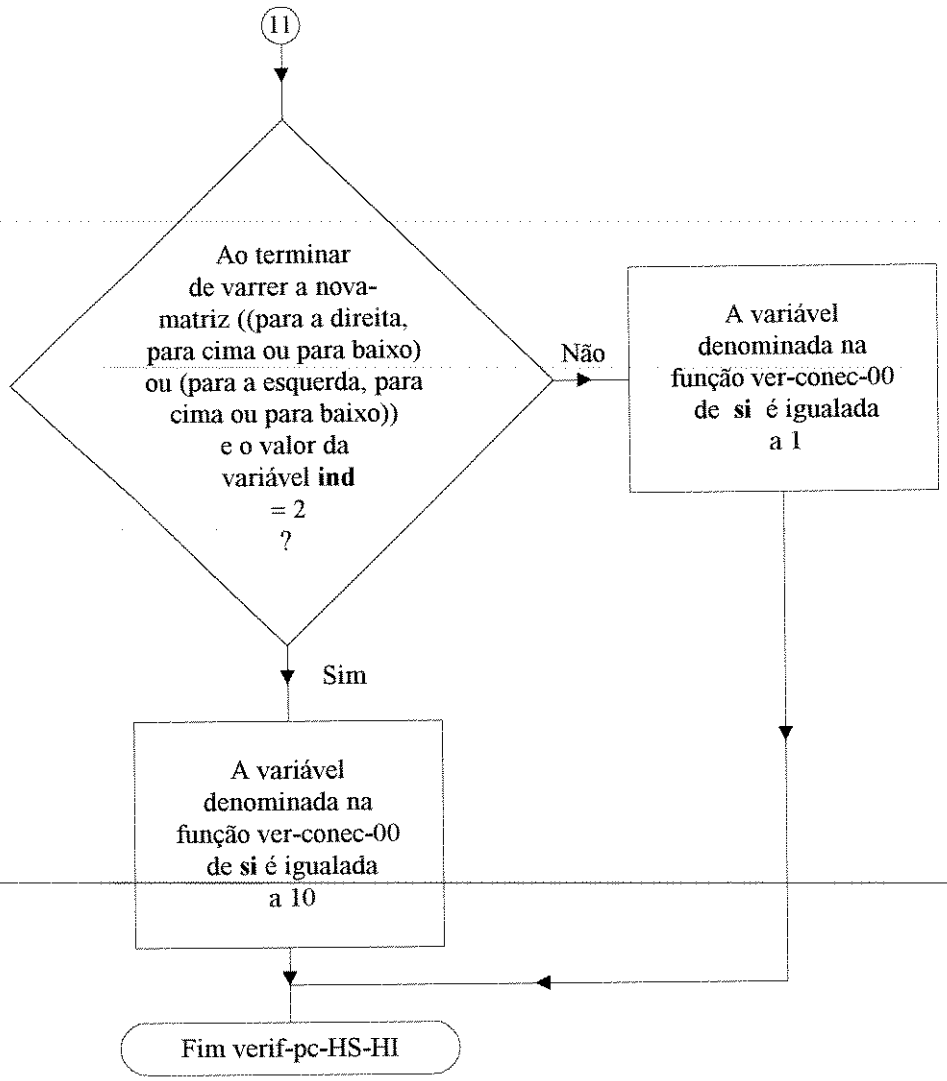
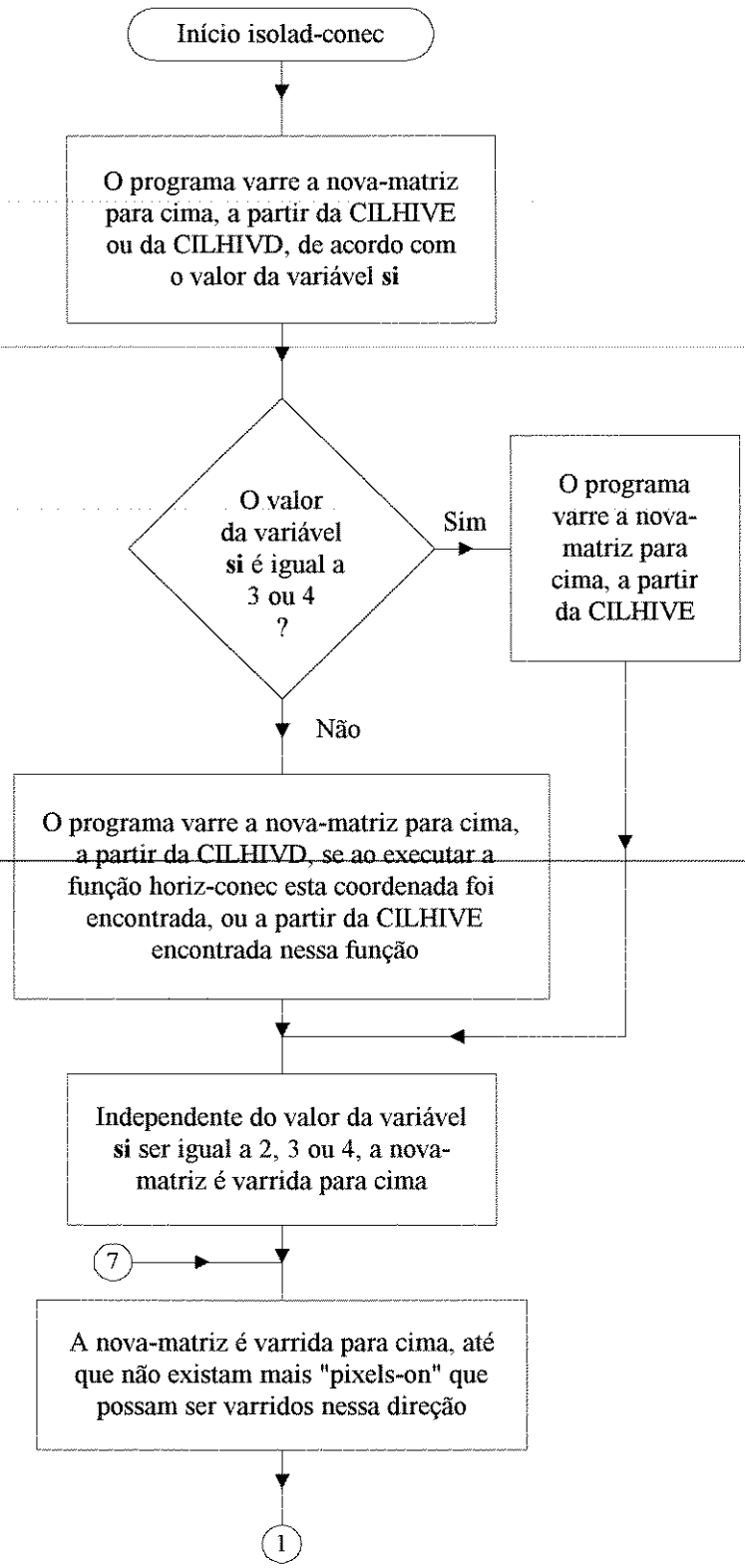


Figura 5.10(g) - Sétima parte do fluxograma da função verif-pc-HS-HI.

Fluxograma da função isolad-conec



LHI - linha horizontal inferior
CILHIVE - coordenada inicial da LHI varrida para a esquerda
CILHIVD - coordenada inicial da LHI varrida para a direita

Figura 5.11(a) - Primeira parte do fluxograma da função isolad-conec.

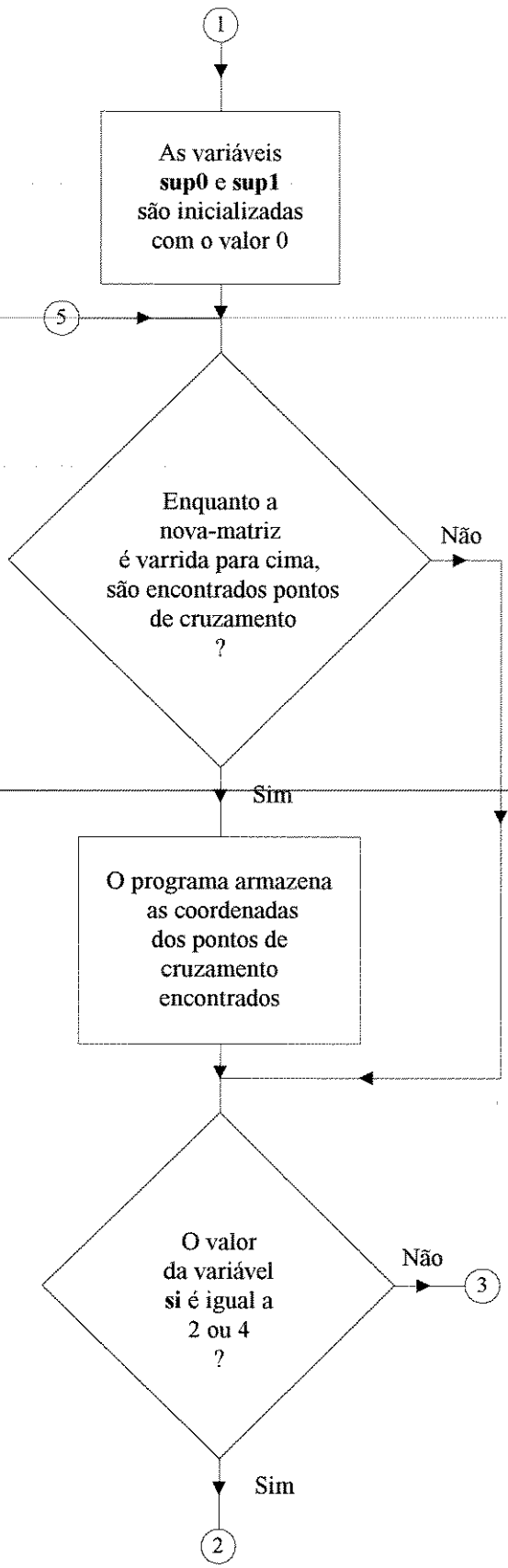
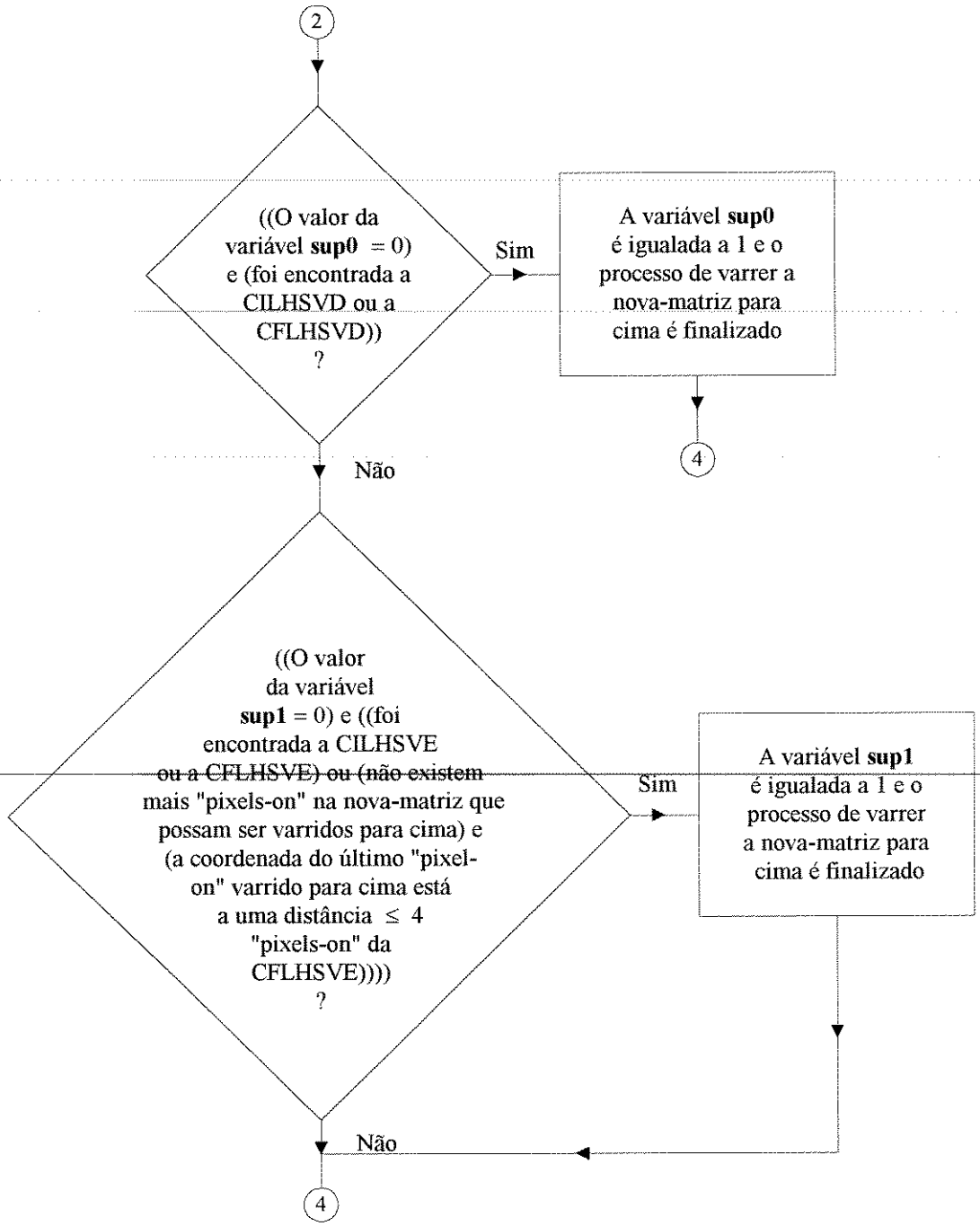
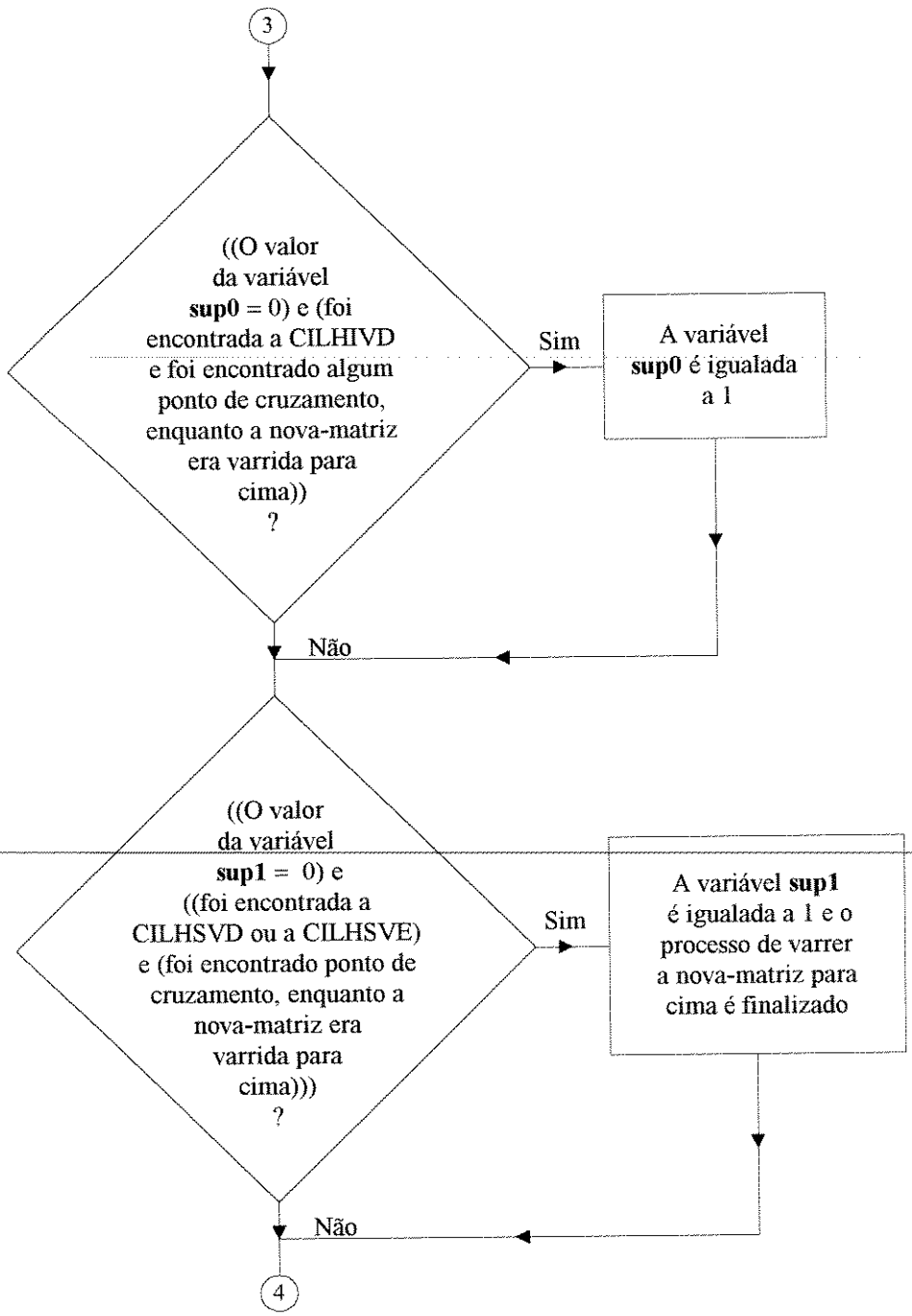


Figura 5.11(b) - Segunda parte do fluxograma da função isolad-conec.



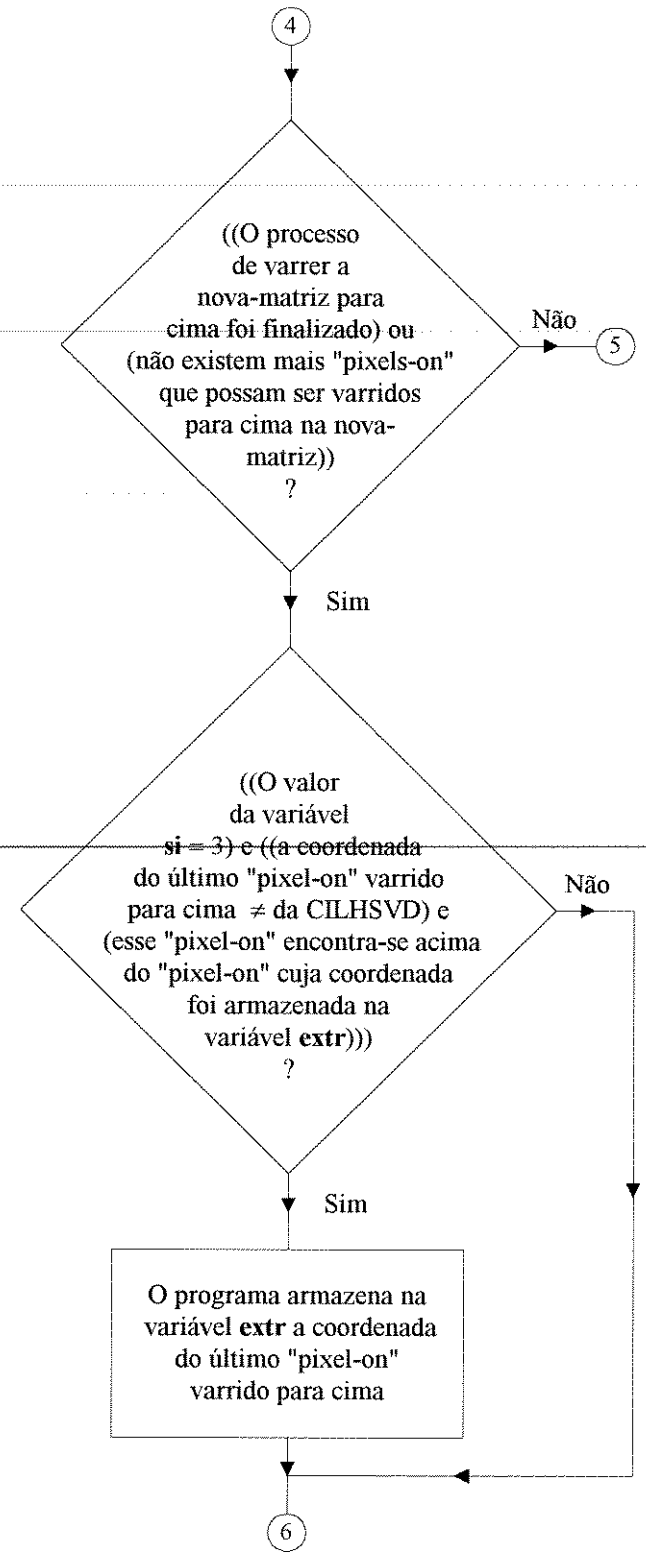
LHS - linha horizontal superior
CILHSVD - coordenada inicial da LHS varrida para a direita
CFLHSVD - coordenada final da LHS varrida para a direita
CILHSVE - coordenada inicial da LHS varrida para a esquerda
CFLHSVE - coordenada final da LHS varrida para a esquerda

Figura 5.11(c) - Terceira parte do fluxograma da função isolad-conec.



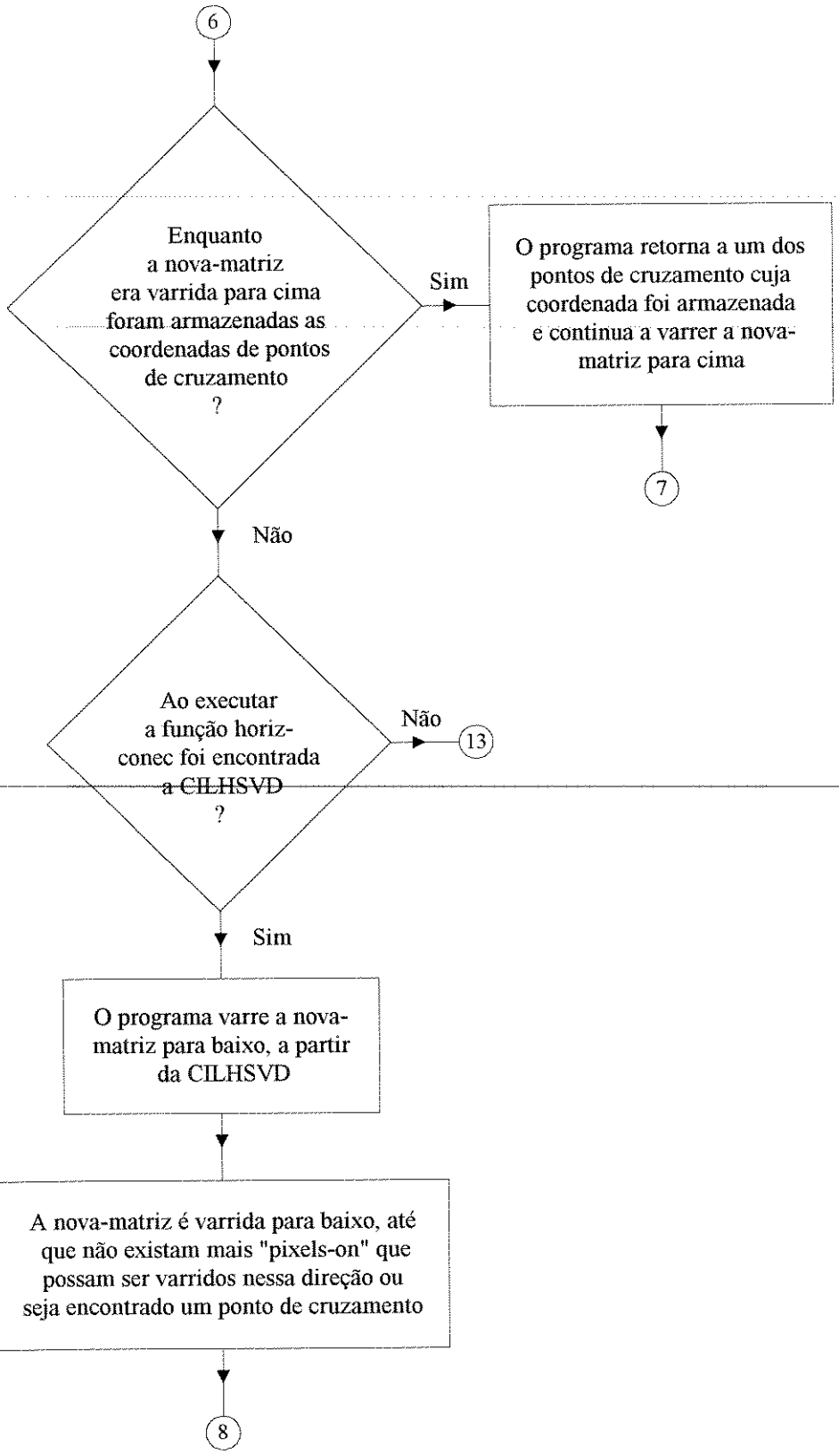
LHS - linha horizontal superior
CILHSVD - coordenada inicial da LHS varrida para a direita
CILHSVE - coordenada inicial da LHS varrida para a esquerda
LHI - linha horizontal inferior
CILHIVD - coordenada inicial da LHI varrida para a direita

Figura 5.11(d) - Quarta parte do fluxograma da função isolad-conec.



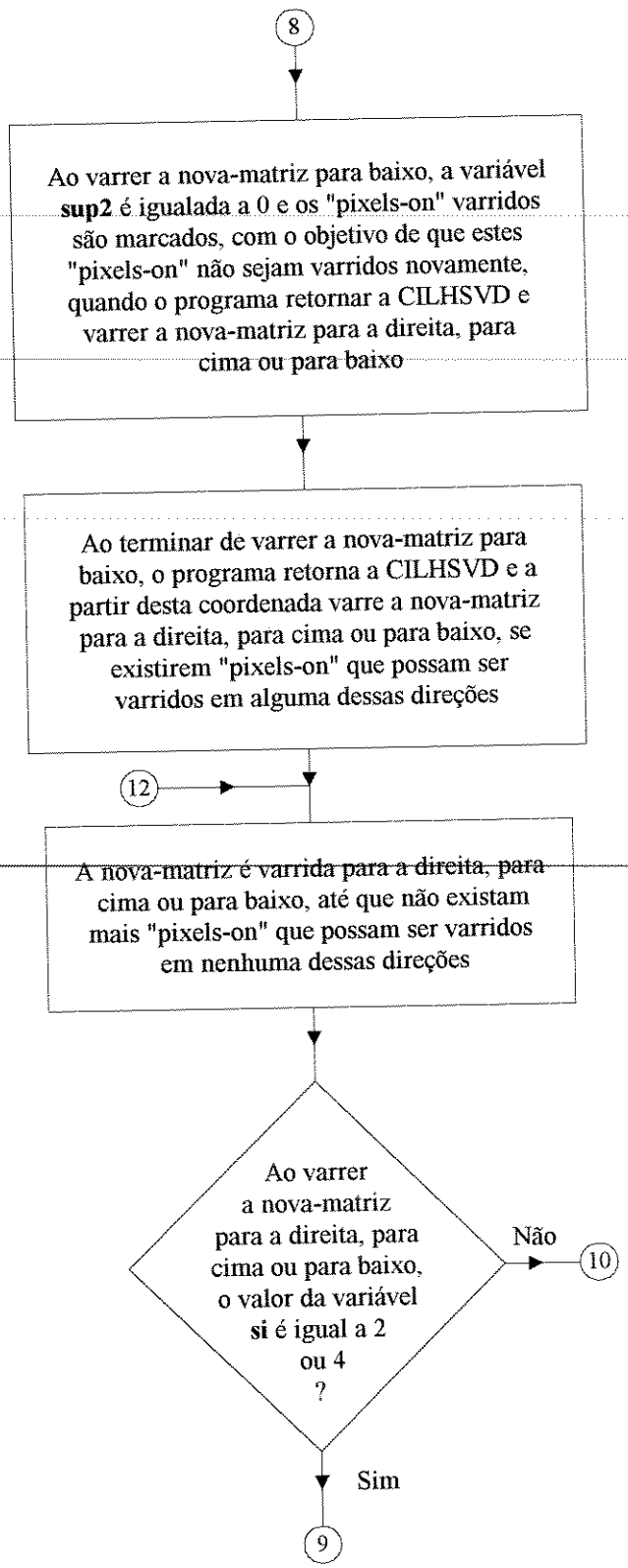
LHS - linha horizontal superior
CILHSVD - coordenada inicial da LHS varrida para a direita

Figura 5.11(e) - Quinta parte do fluxograma da função isolad-conec.



LHS - linha horizontal superior
CILHSVD - coordenada inicial da LHS varrida para a direita

Figura 5.11(f) - Sexta parte do fluxograma da função isolad-conec.



LHS - linha horizontal superior
CILHSVD - coordenada inicial da LHS varrida para a direita

Figura 5.11(g) - Sétima parte do fluxograma da função isolad-conec.

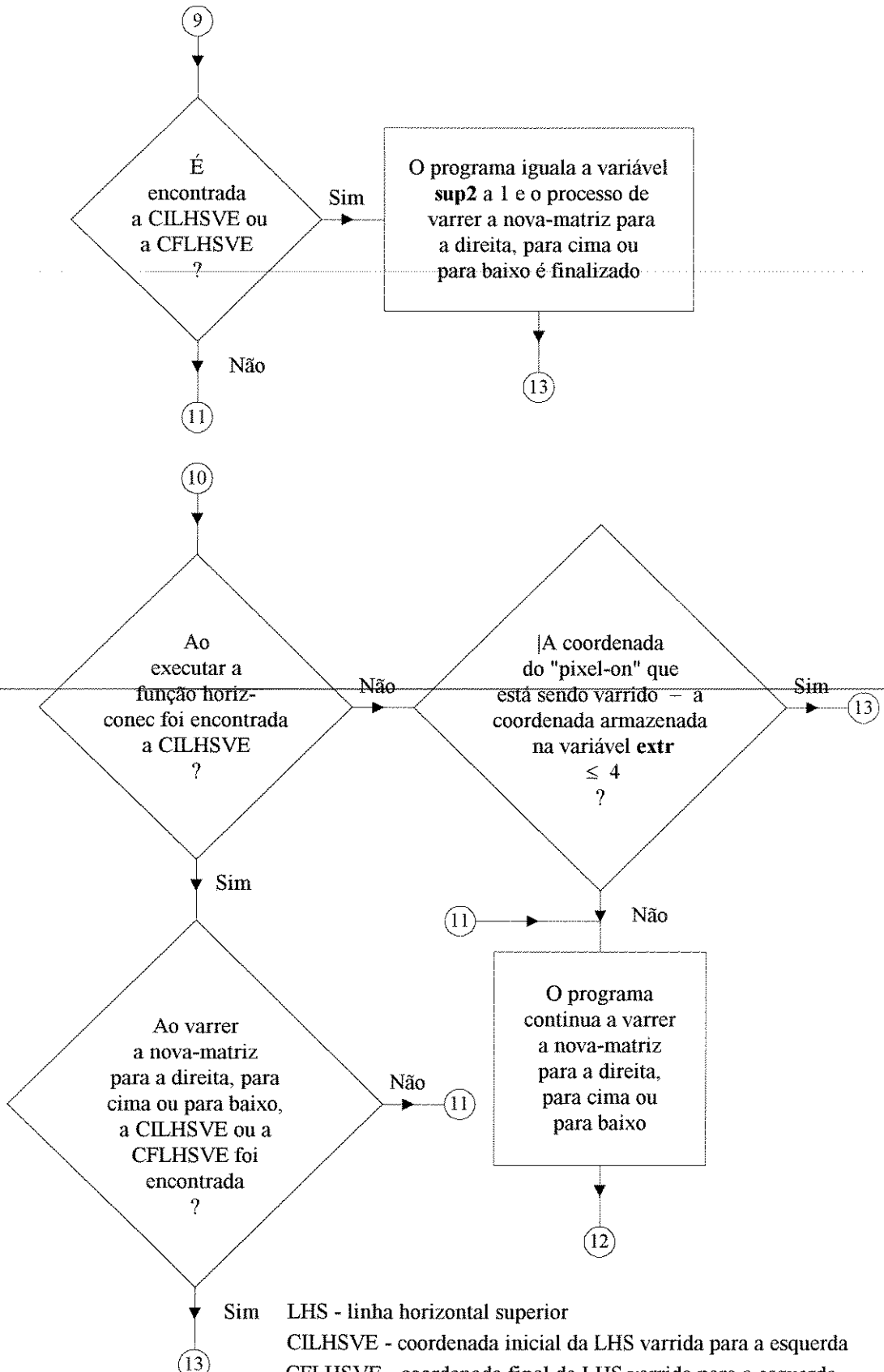
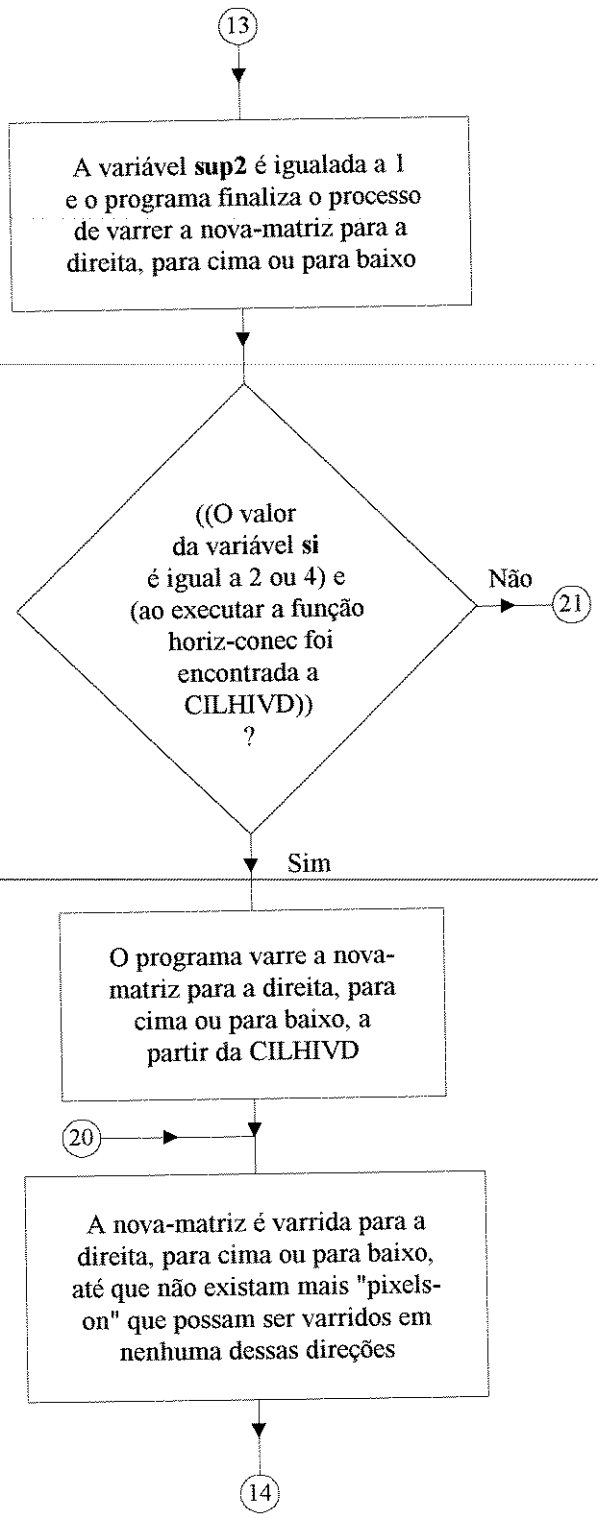


Figura 5.11(h) - Oitava parte do fluxograma da função isolad-conec.



LHI - linha horizontal inferior
CILHIVD - coordenada inicial da LHI varrida para a direita

Figura 5.11(i) - Nona parte do fluxograma da função isolad-conec.

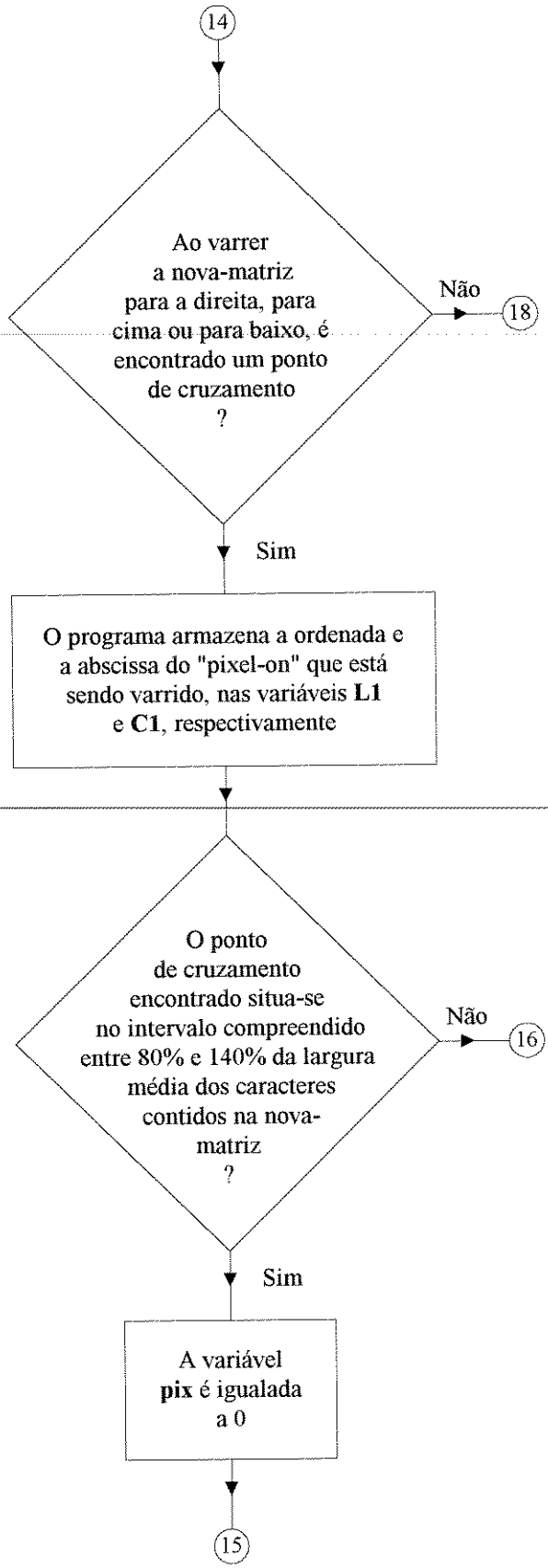


Figura 5.11(j) - Décima parte do fluxograma da função isolad-conec.

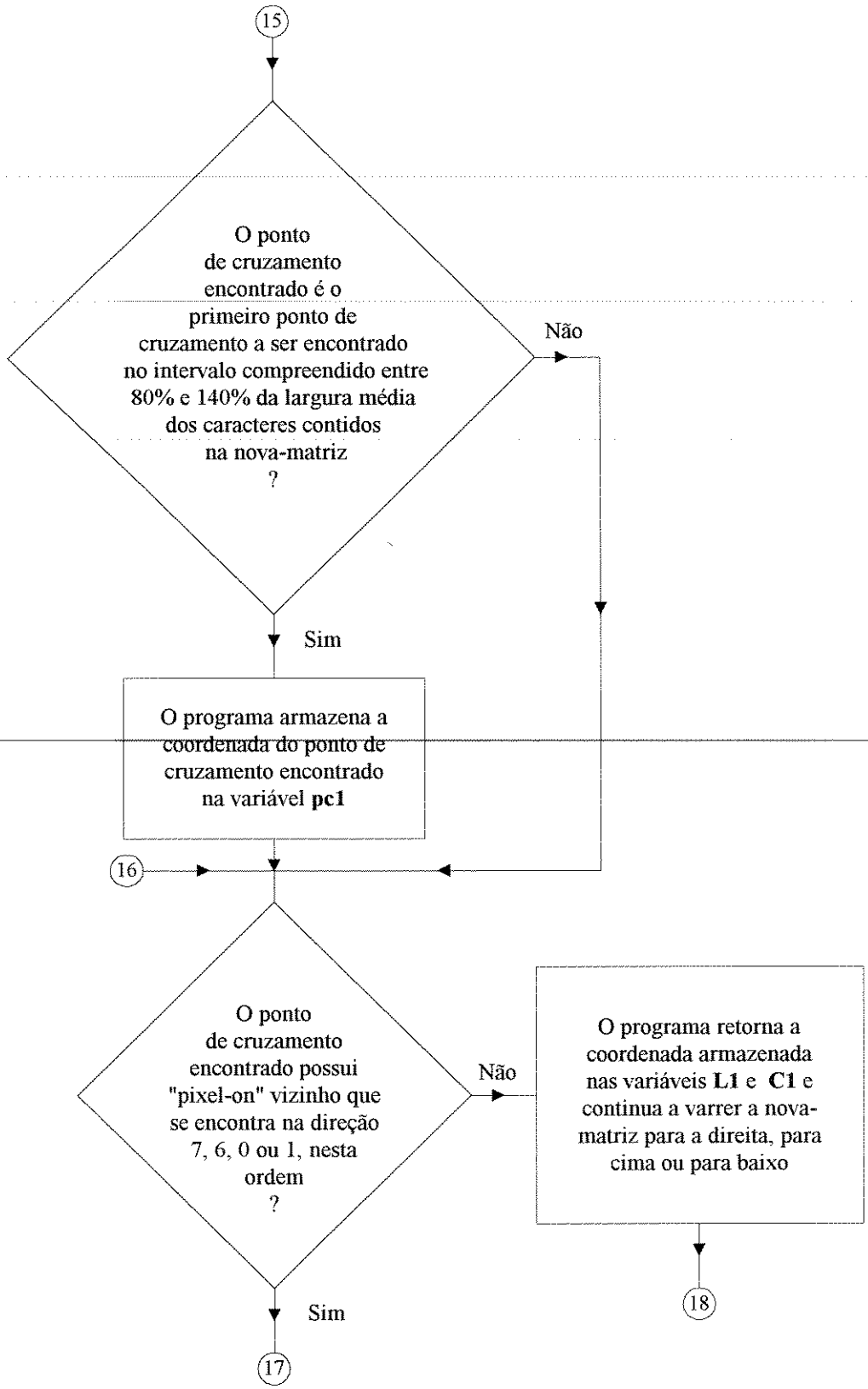


Figura 5.11(k) - Décima primeira parte do fluxograma da função isolad-conec.

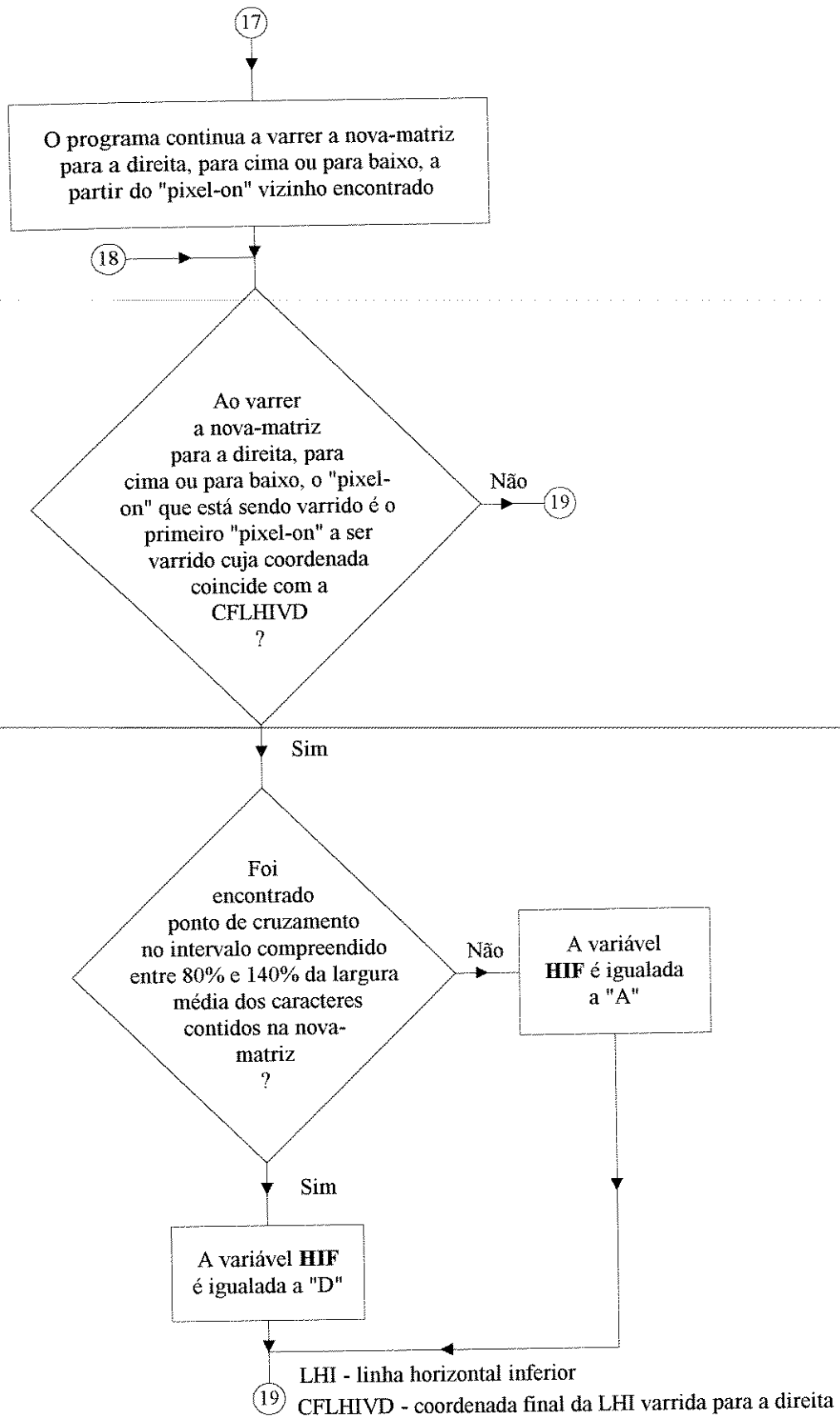


Figura 5.11(l) - Décima segunda parte do fluxograma da função isolad-conec.

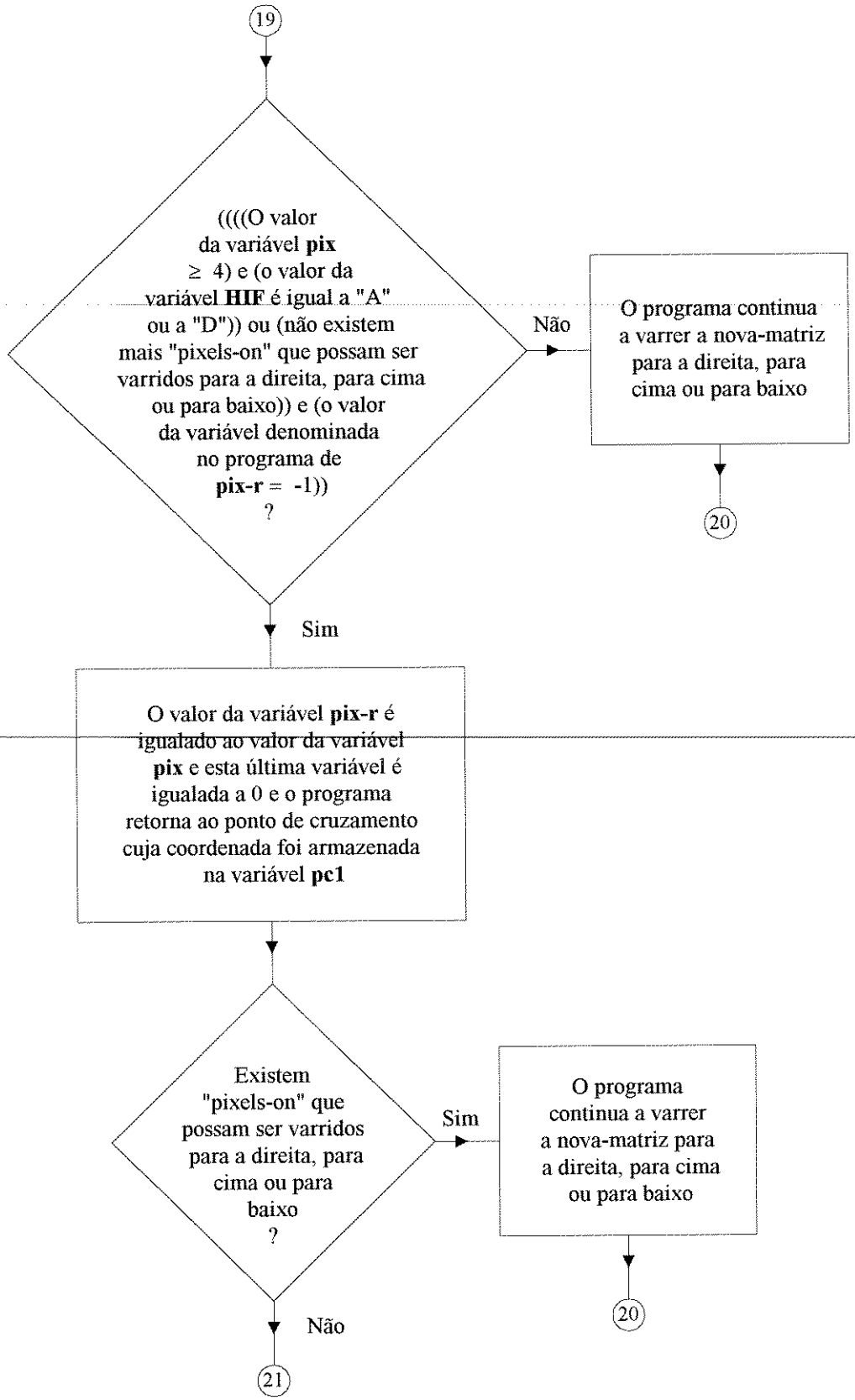


Figura 5.11(m) - Décima terceira parte do fluxograma da função isolad-conec.

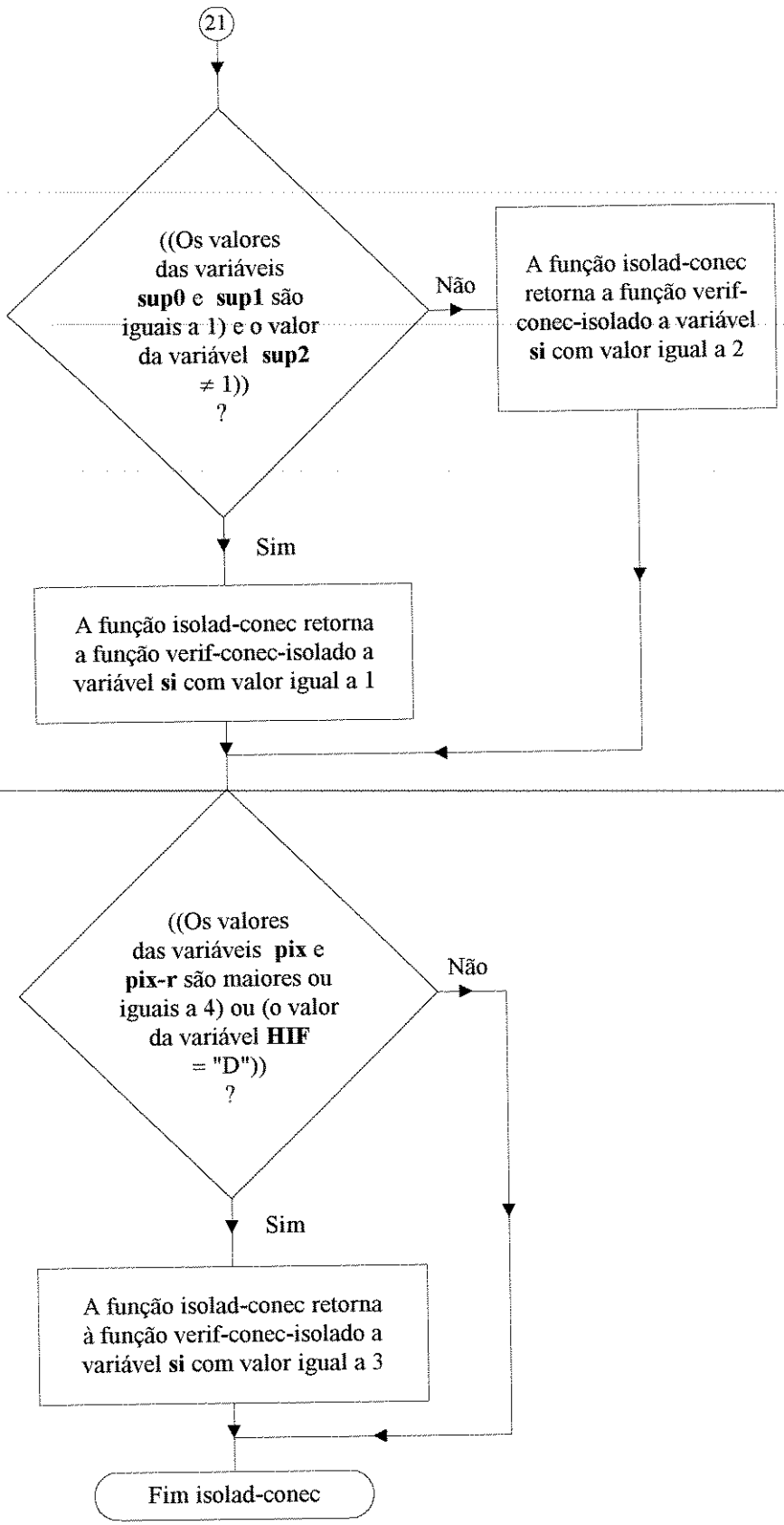


Figura 5.11(n) - Décima quarta parte do fluxograma da função isolad-conec.

Fluxograma da função conectados

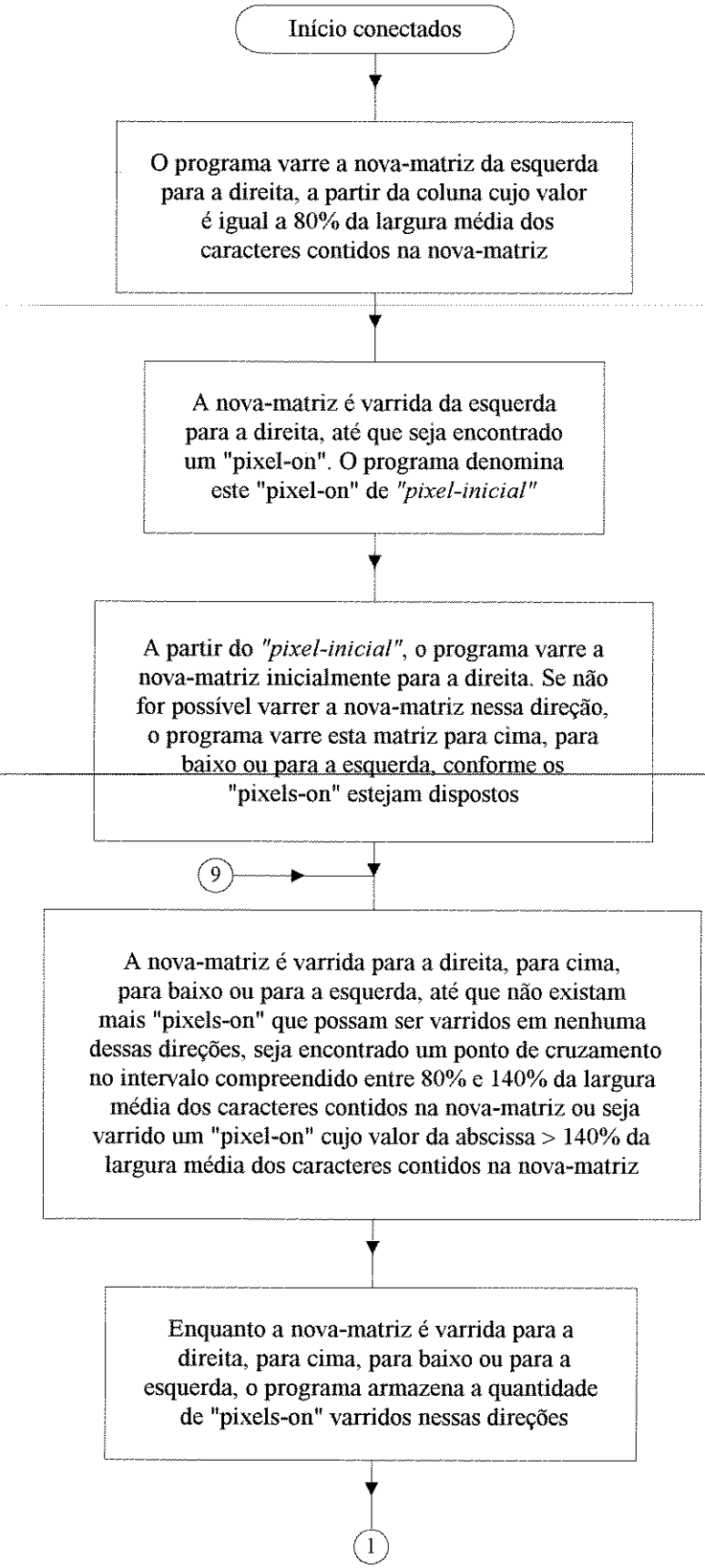


Figura 5.12(a) - Primeira parte do fluxograma da função conectados.

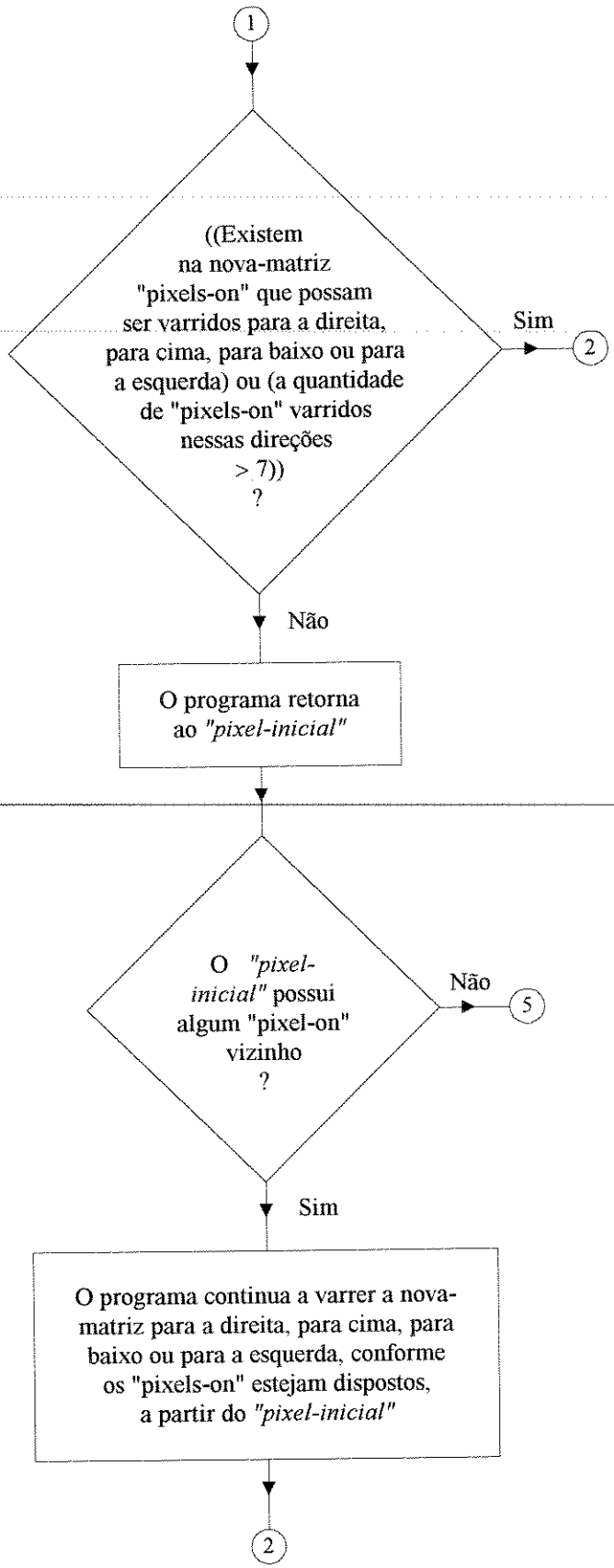


Figura 5.12(b) - Segunda parte do fluxograma da função conectados.

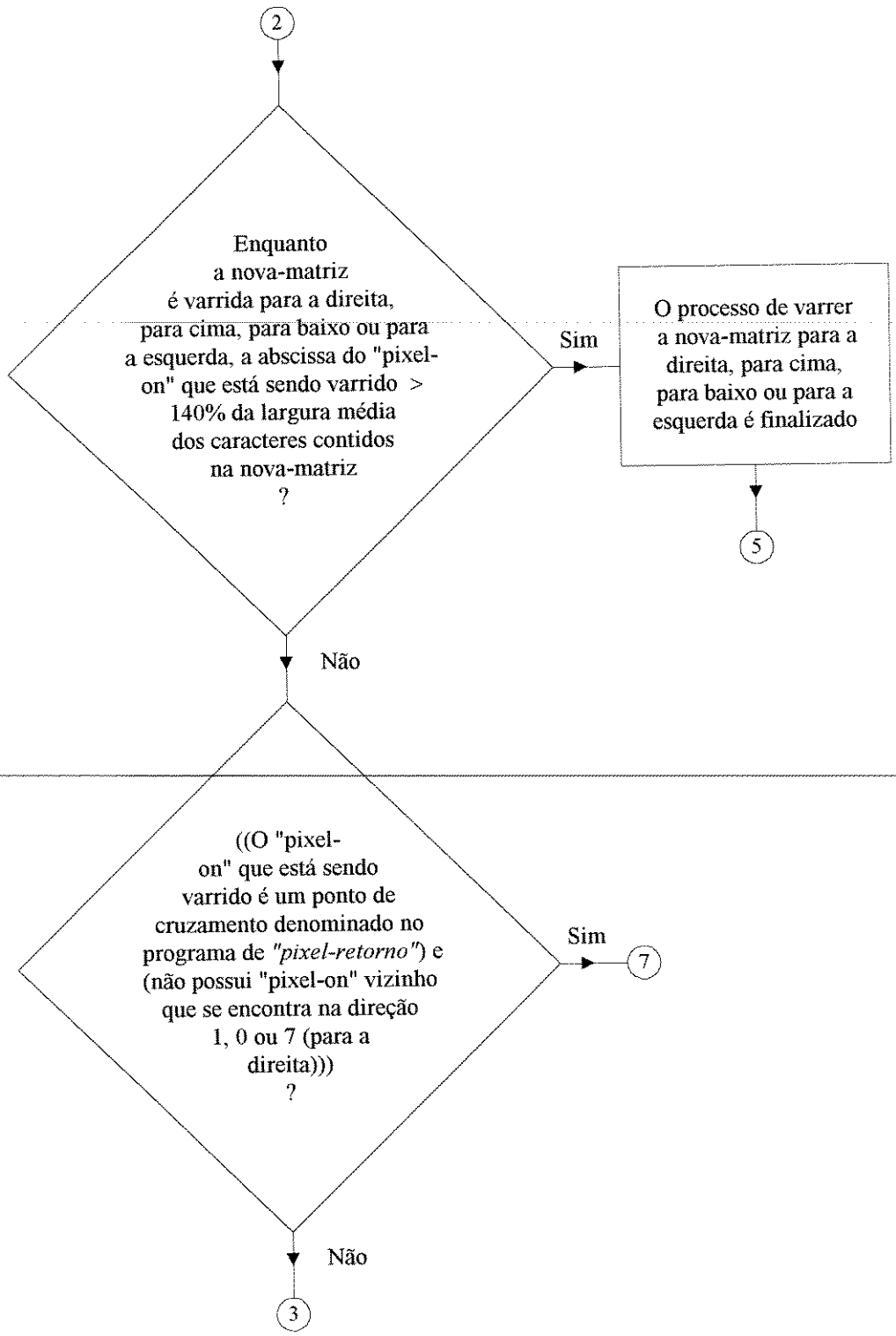


Figura 5.12(c) - Terceira parte do fluxograma da função conectados.

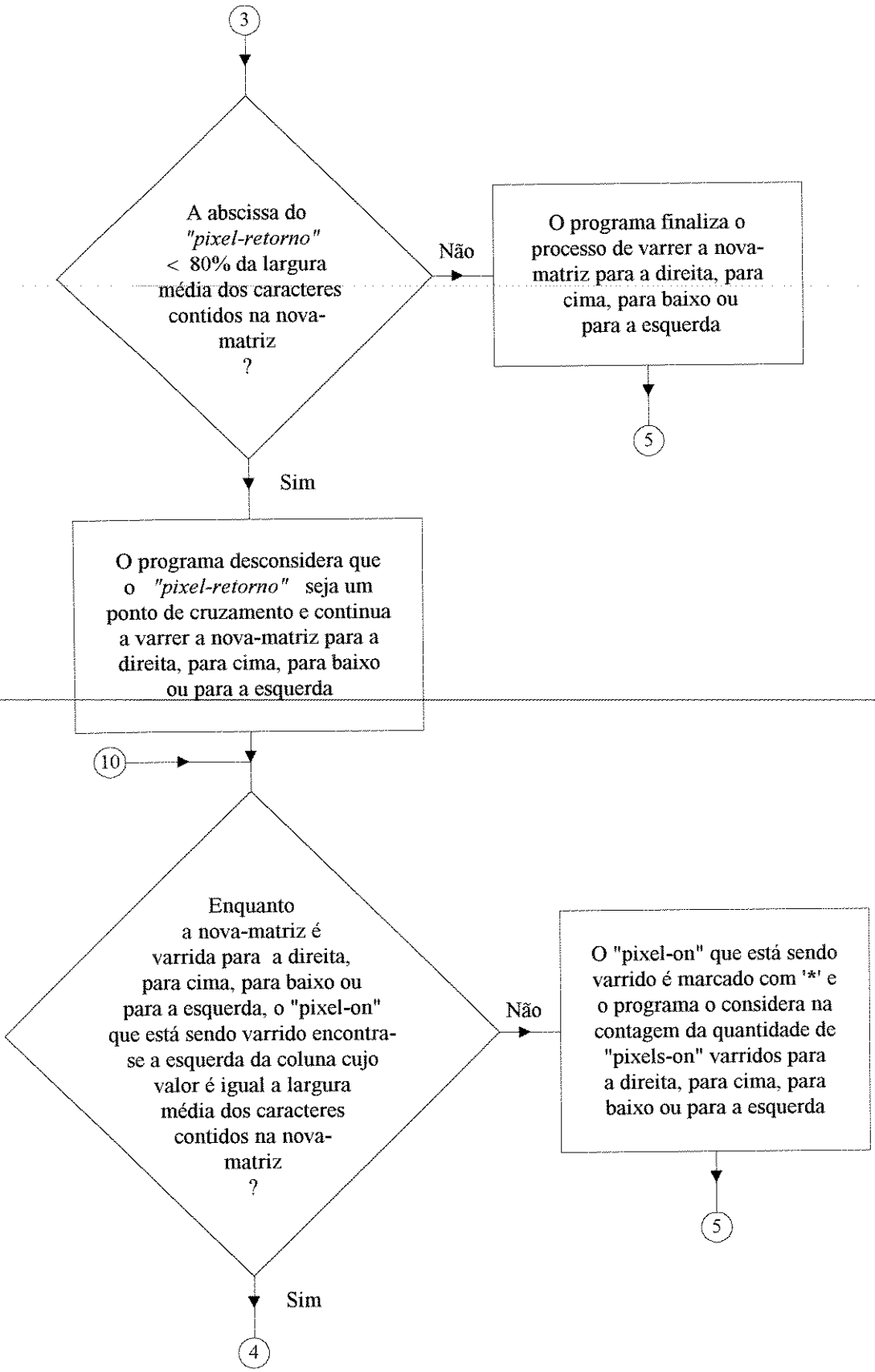
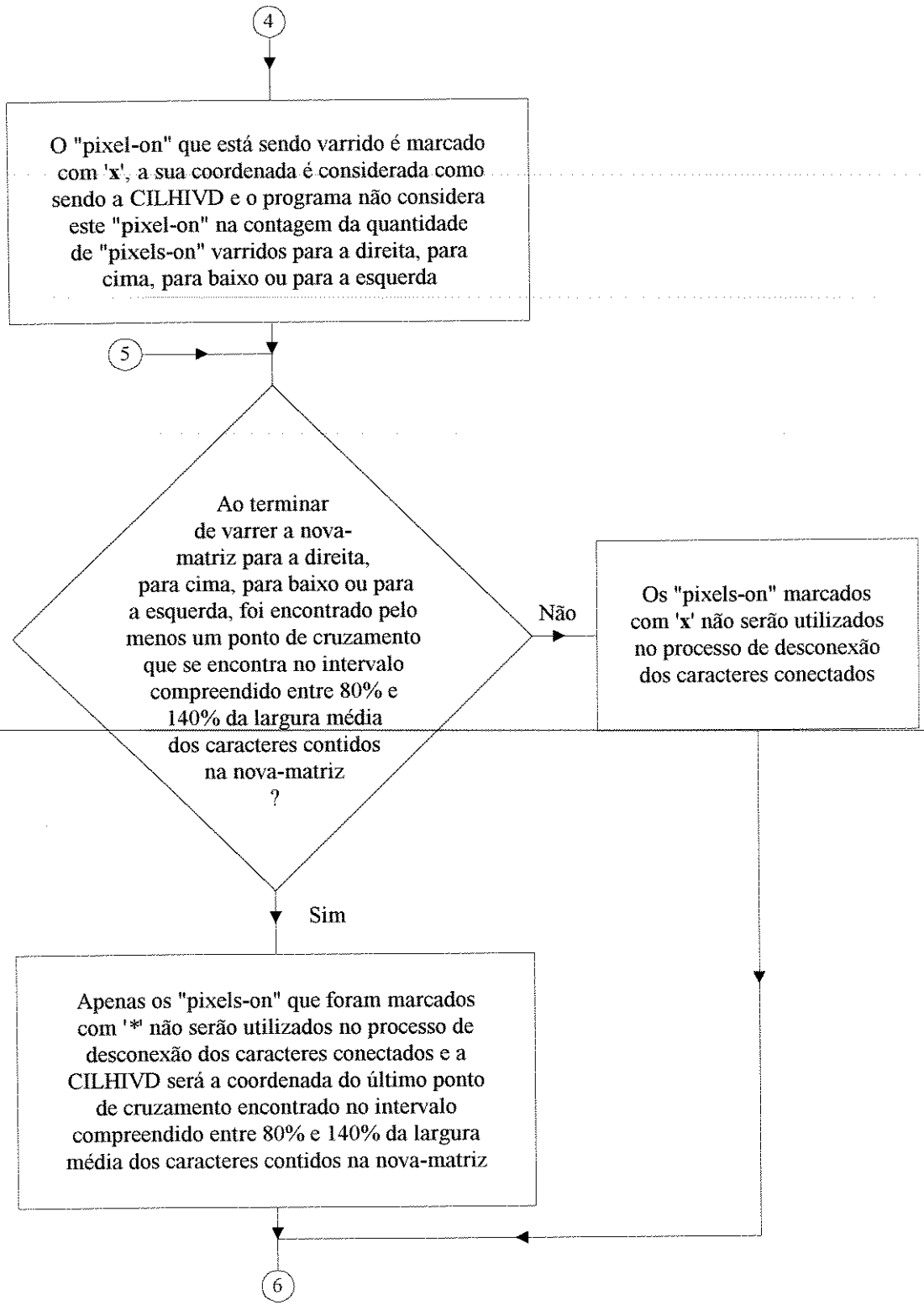


Figura 5.12(d) - Quarta parte do fluxograma da função conectados.



LHI - linha horizontal inferior
CILHIVD - coordenada inicial da LHI varrida para a direita

Figura 5.12(e) - Quinta parte do fluxograma da função conectados.

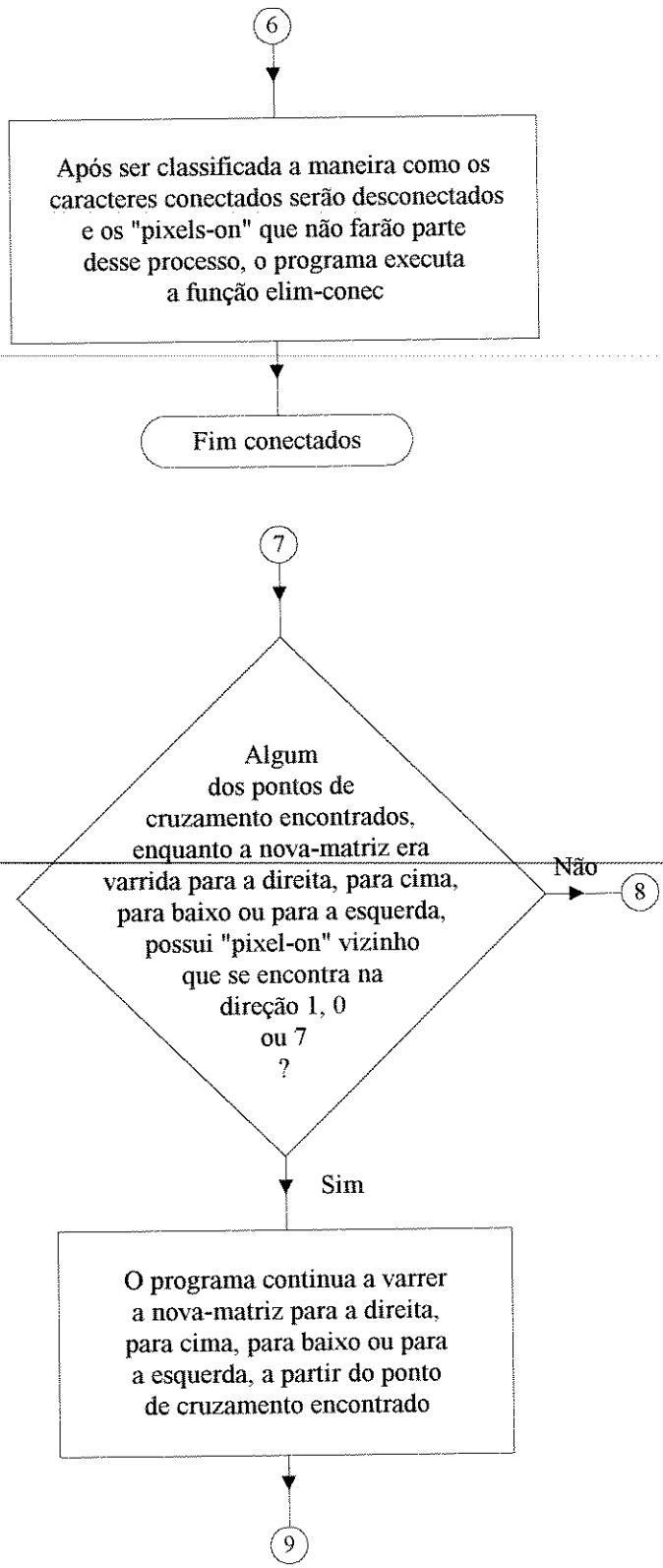


Figura 5.12(f) - Sexta parte do fluxograma da função conectados.

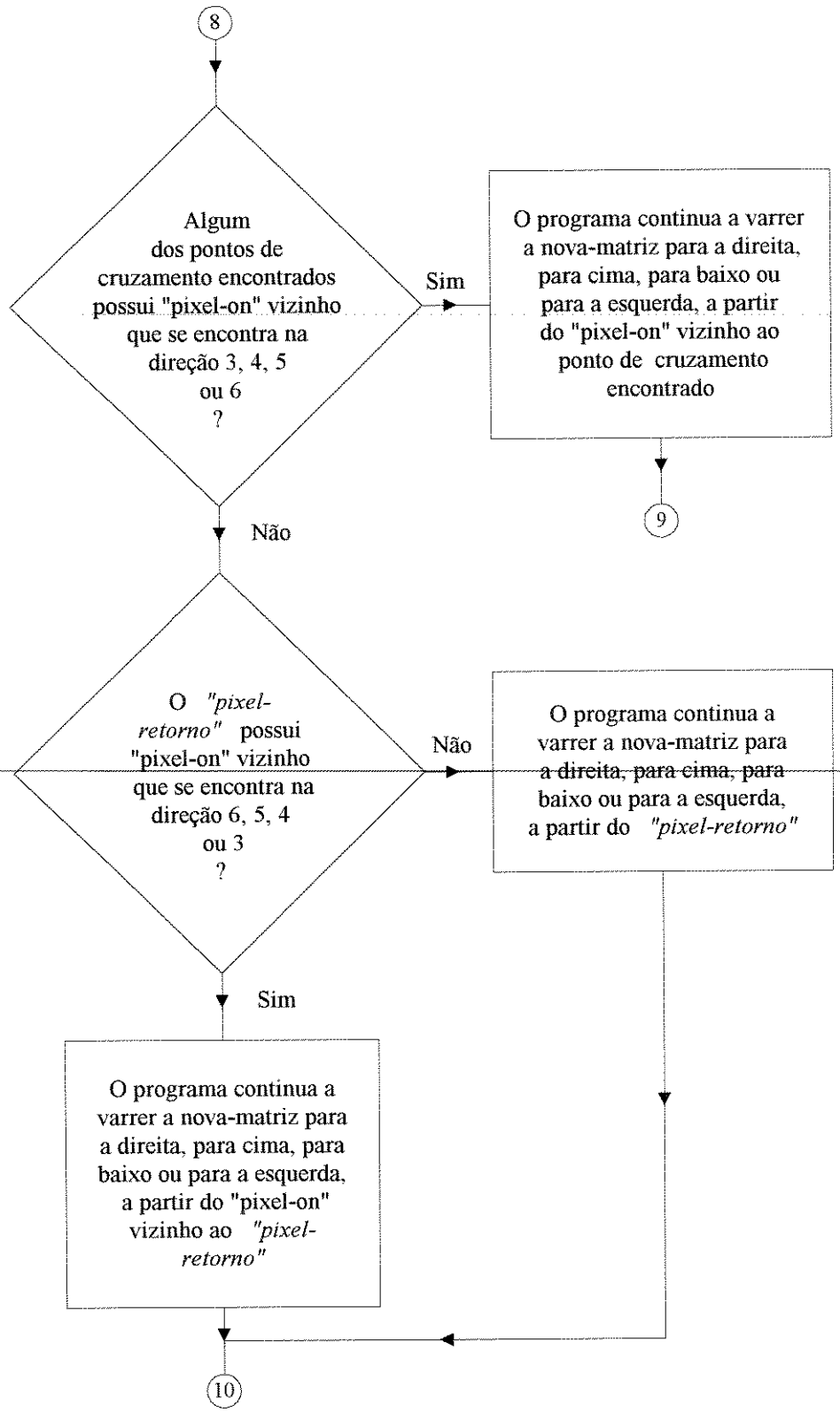
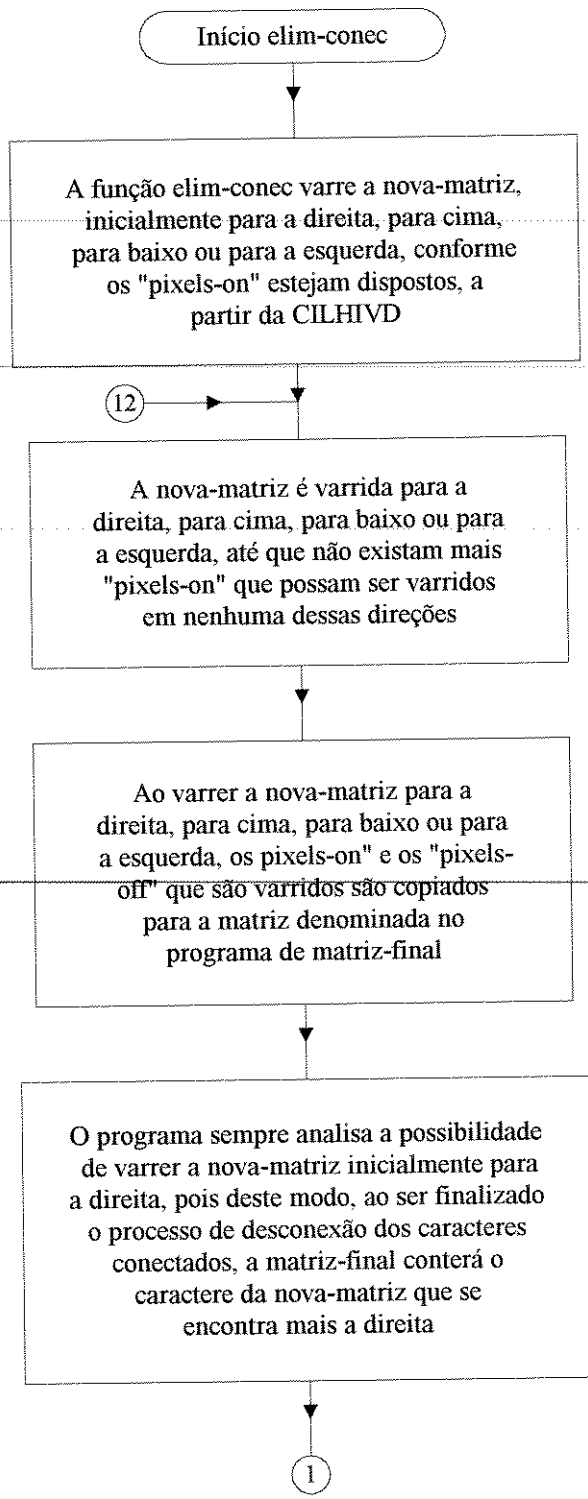


Figura 5.12(g) - Sétima parte do fluxograma da função conectados.

Fluxograma da função elim-conec



LHI - linha horizontal inferior
CILHIVD - coordenada inicial da LHI varrida para a direita

Figura 5.13(a) - Primeira parte do fluxograma da função elim-conec.

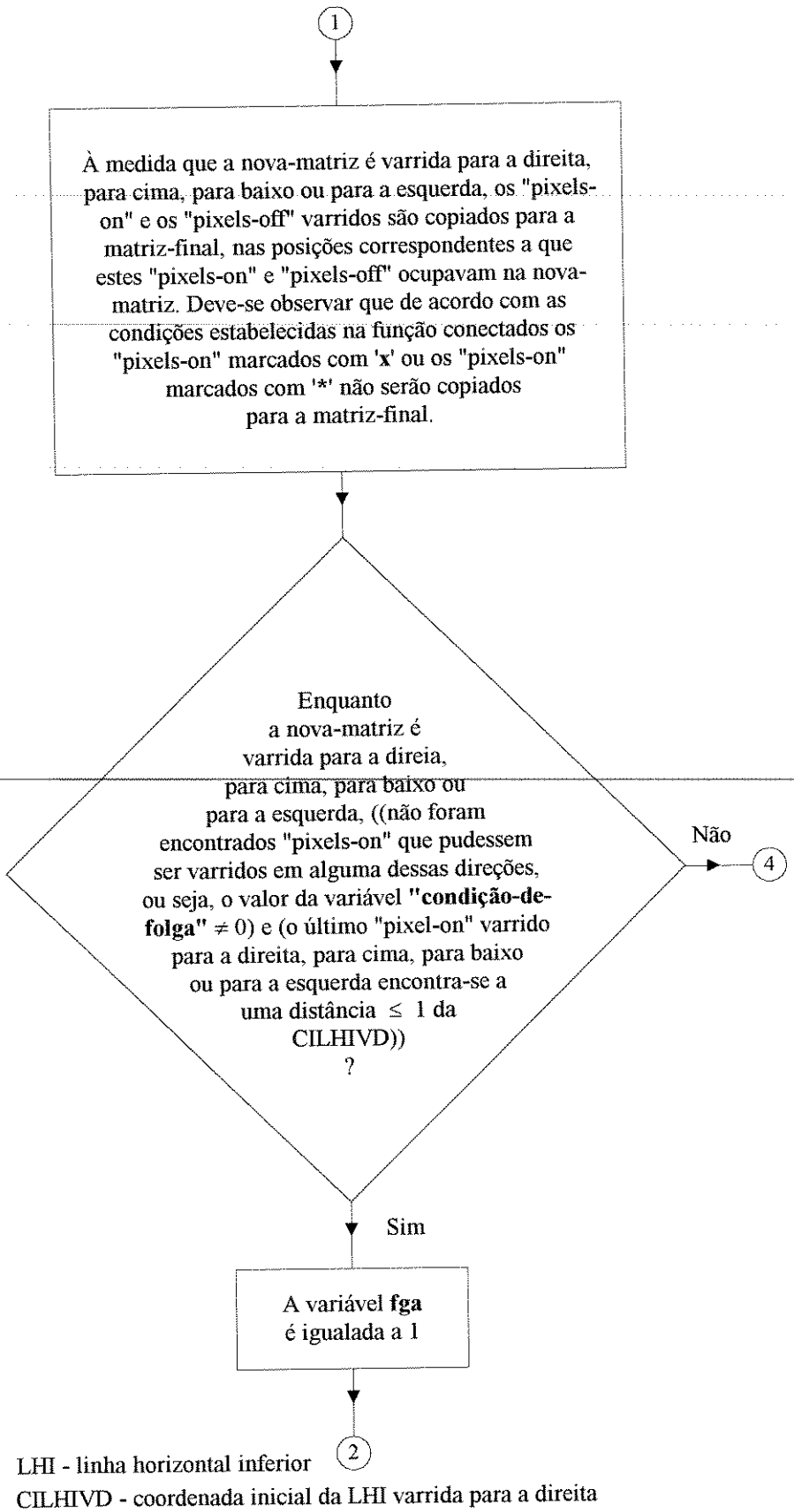


Figura 5.13(b) - Segunda parte do fluxograma da função elim-conec.

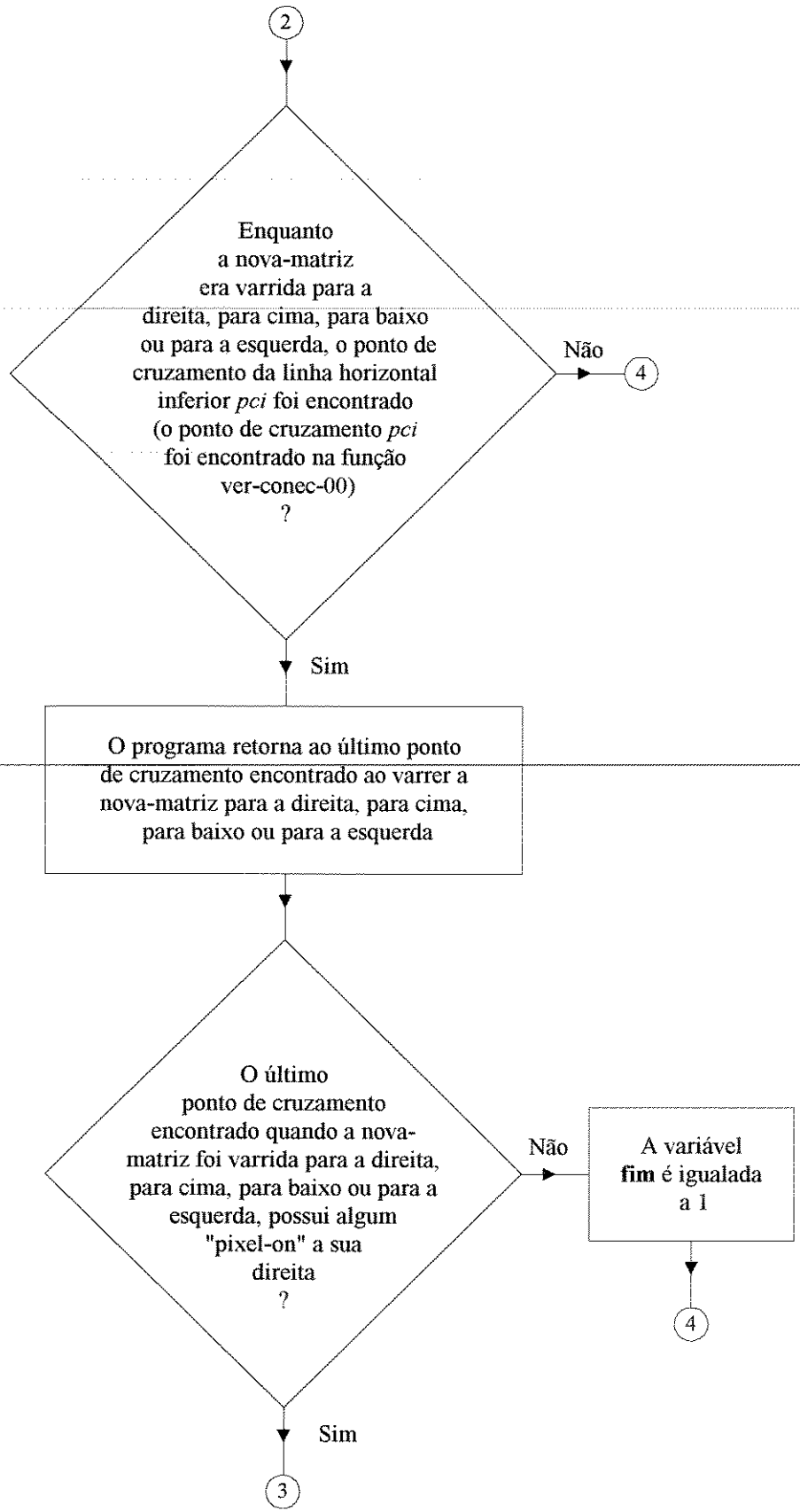


Figura 5.13(c) - Terceira parte do fluxograma da função elim-conec.

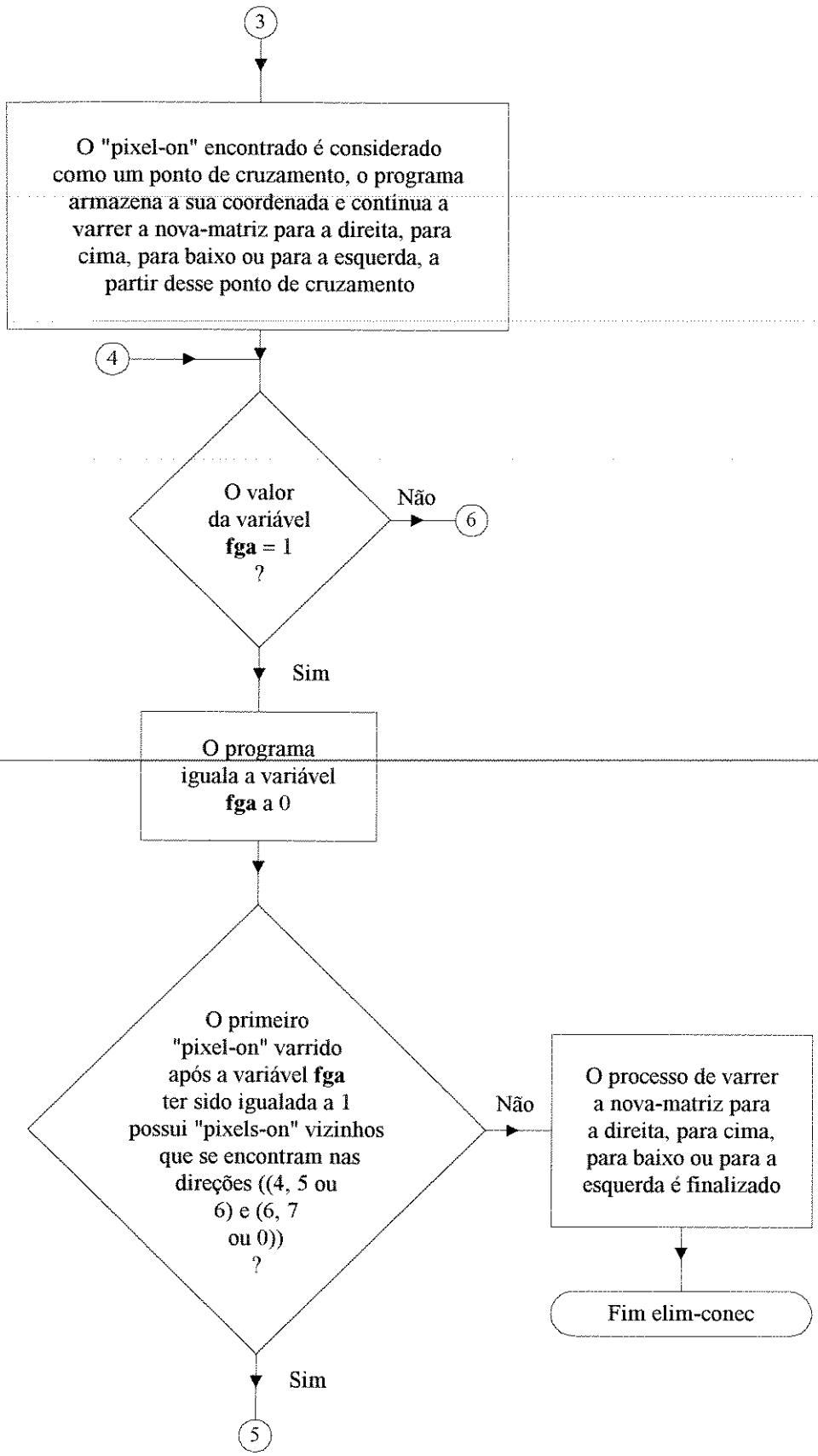


Figura 5.13(d) - Quarta parte do fluxograma da função elim-conec.

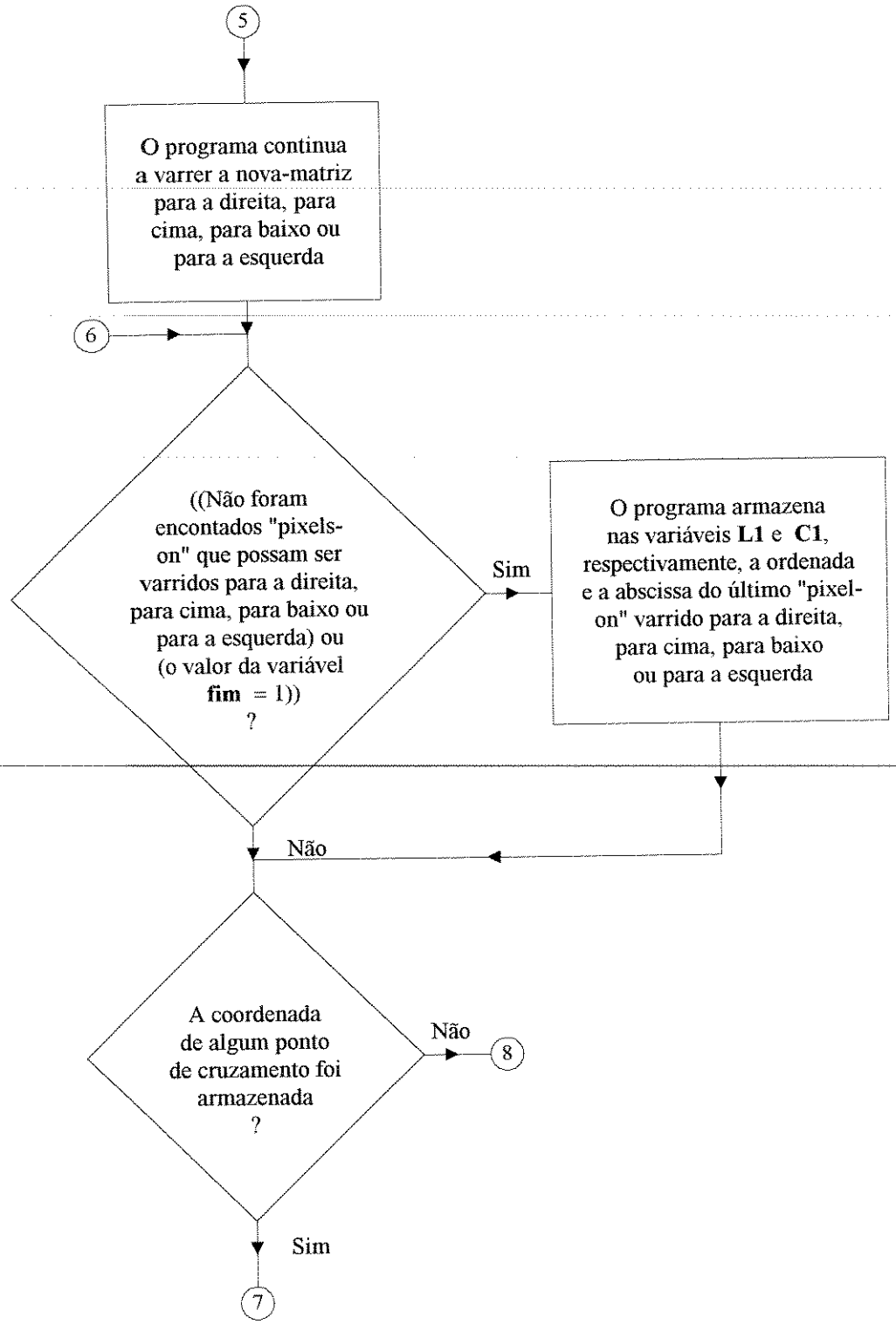


Figura 5.13(e) - Quinta parte do fluxograma da função elim-conec.

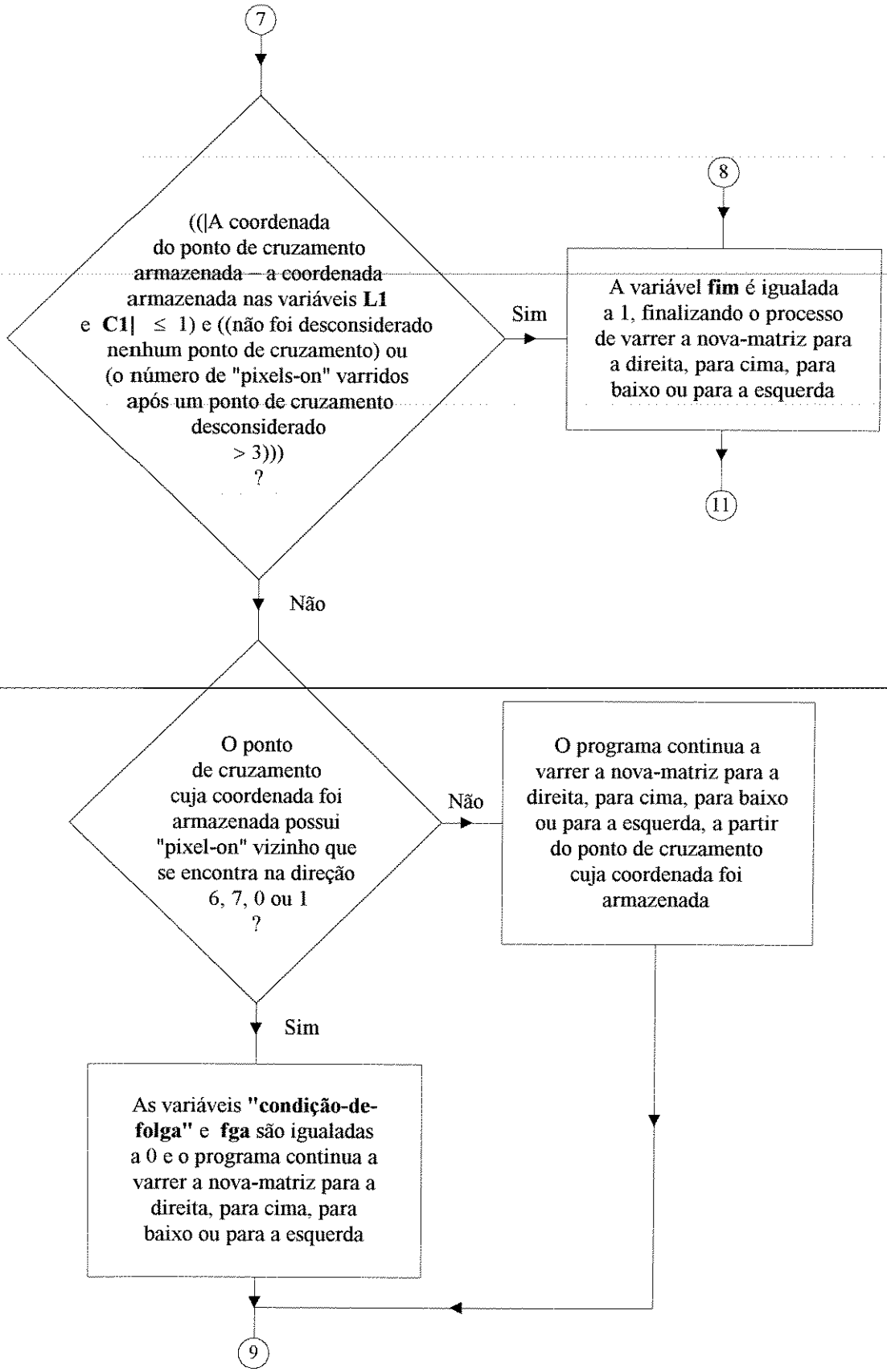


Figura 5.13(f) - Sexta parte do fluxograma da função elim-conec.

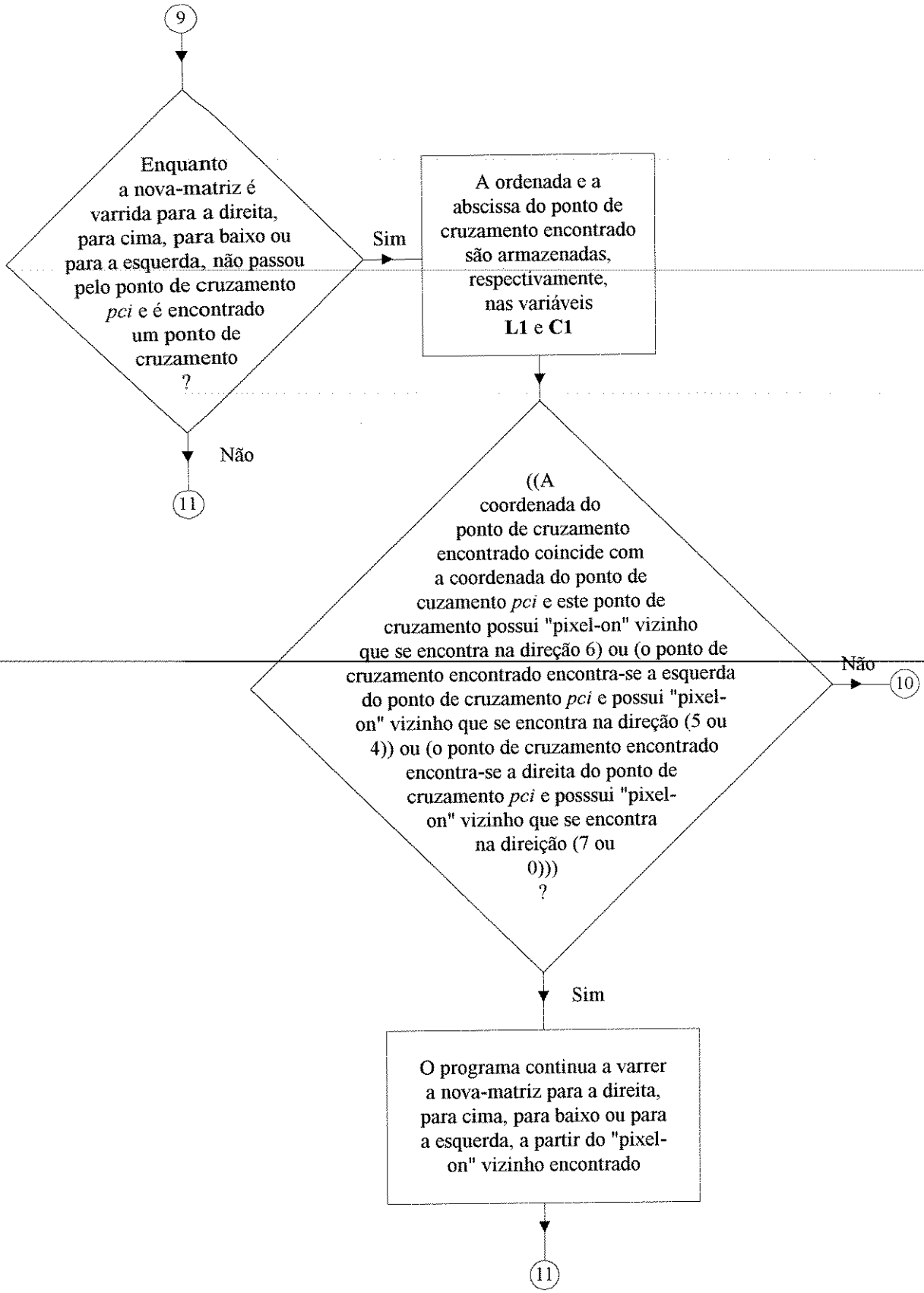


Figura 5.13(g) - Sétima parte do fluxograma da função elim-conec.

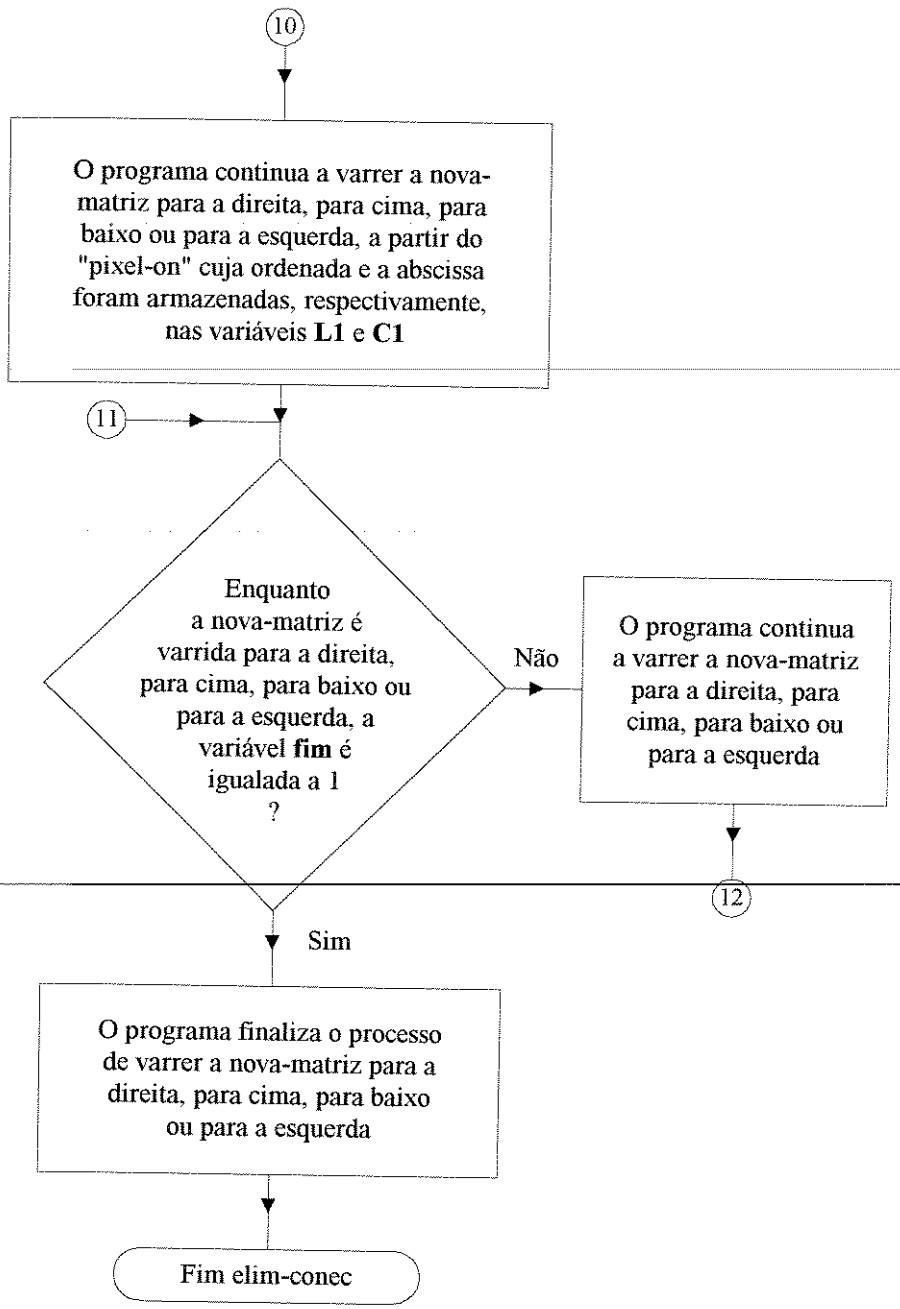


Figura 5.13(h) - Oitava parte do fluxograma da função elim-conec.

Fluxograma da função tarja-horiz-sup

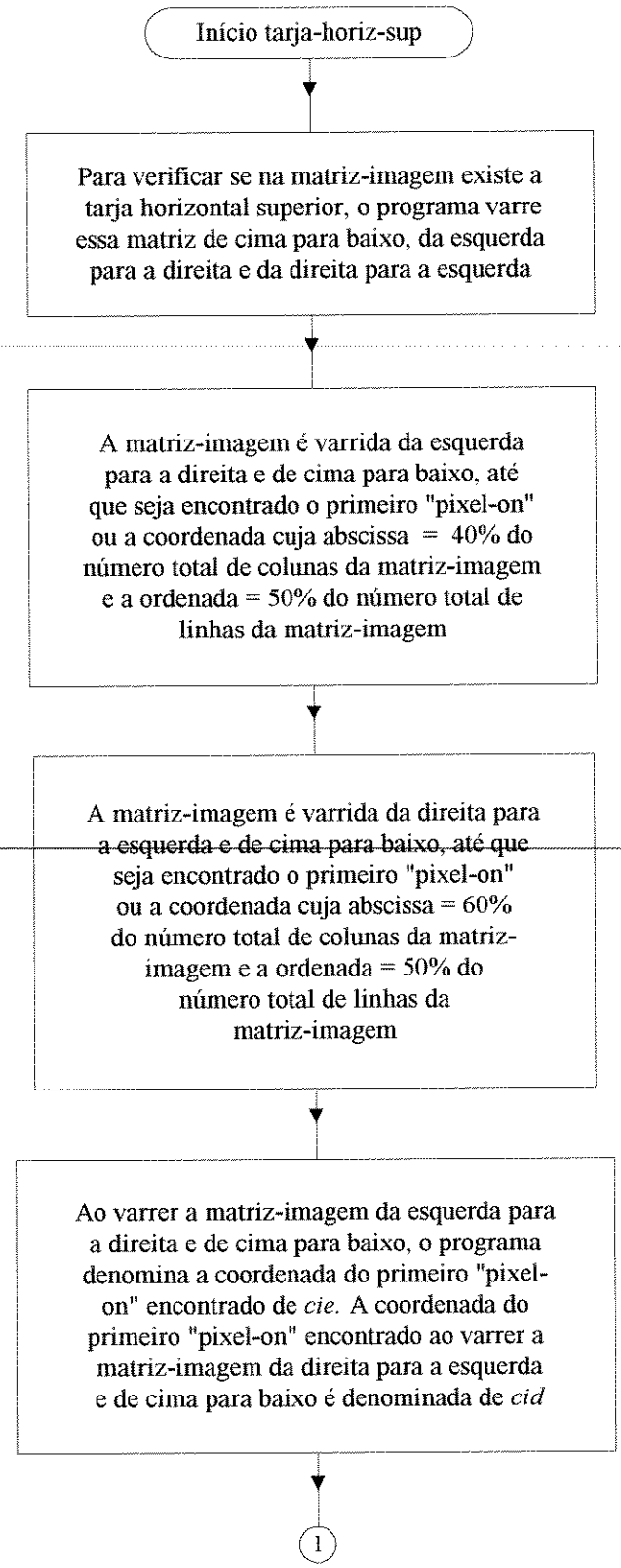


Figura 5.16(a) - Primeira parte do fluxograma da função tarja-horiz-sup.

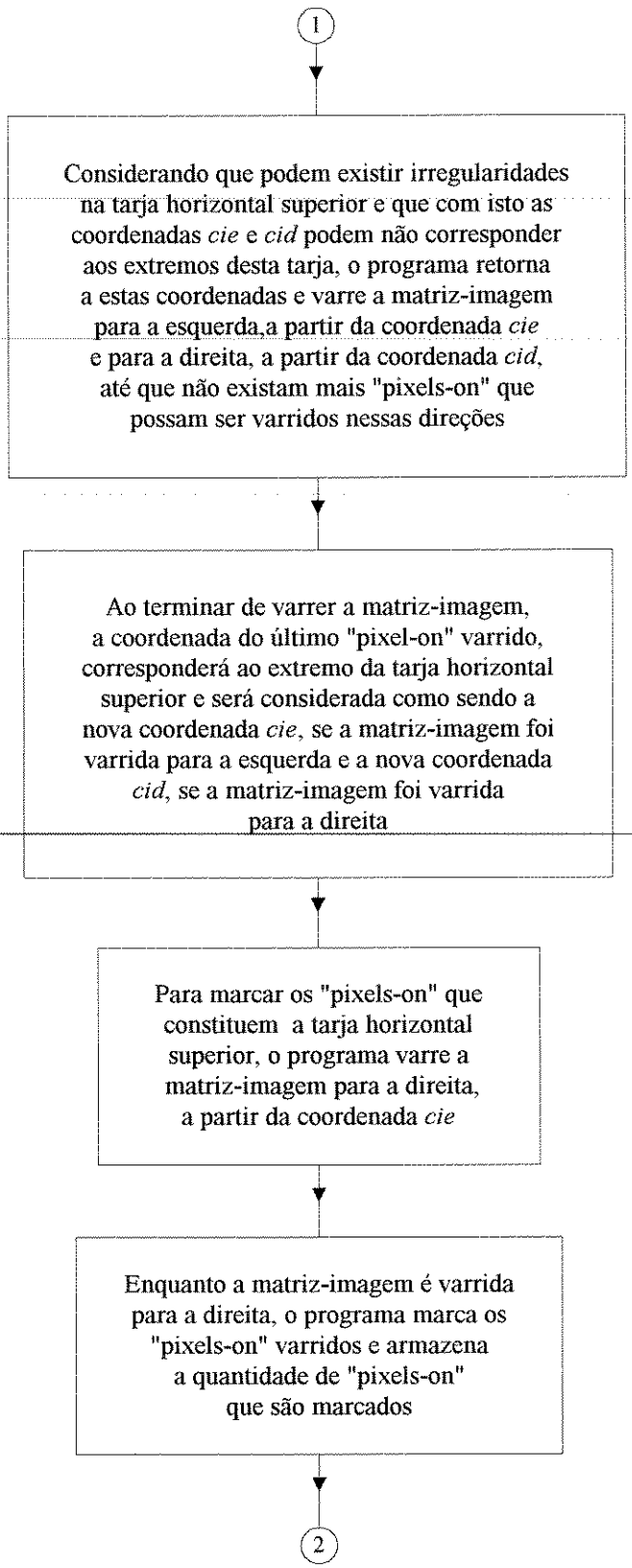


Figura 5.16(b) - Segunda parte do fluxograma da função tarja-horiz-sup.

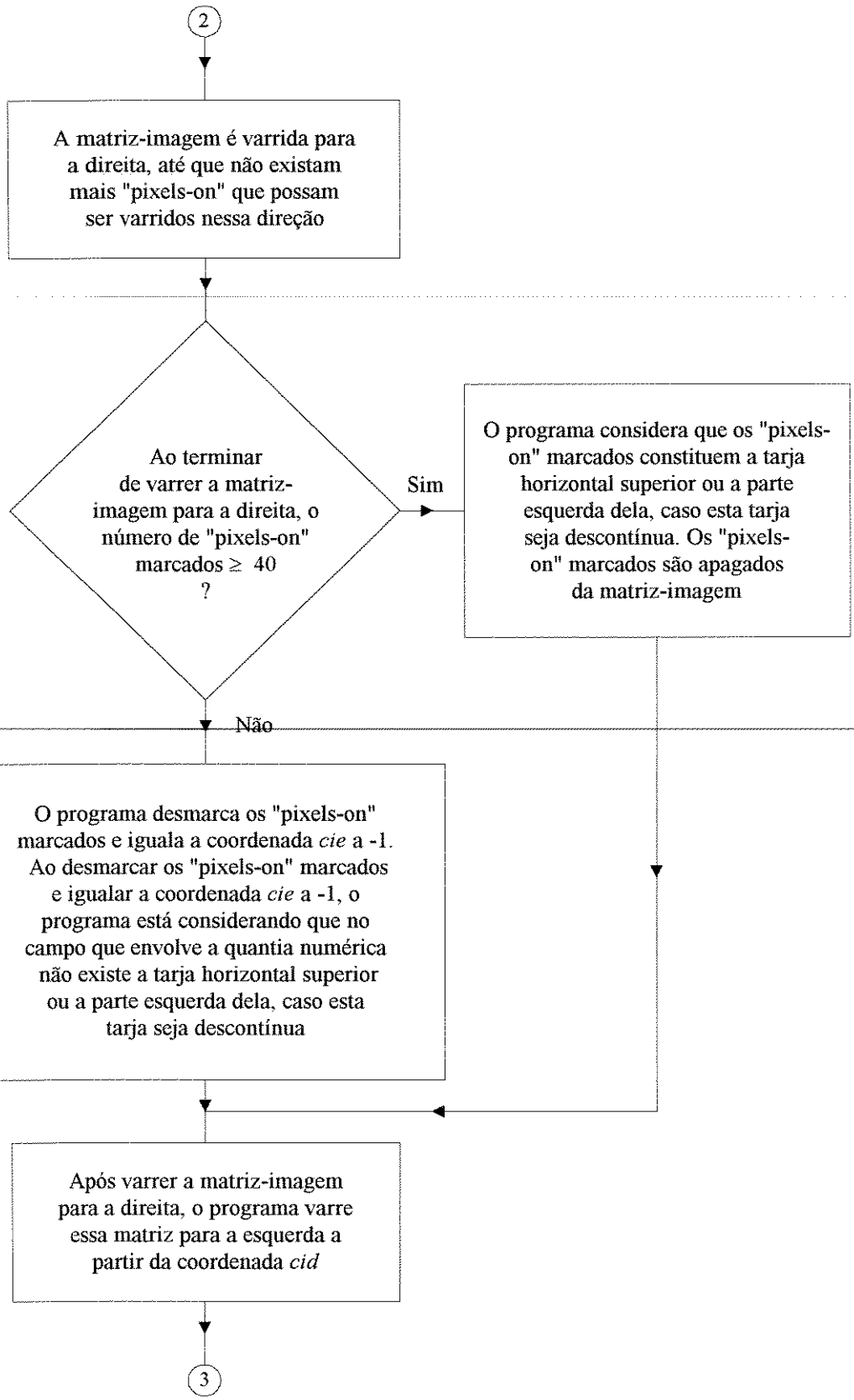


Figura 5.16(c) - Terceira parte do fluxograma da função tarja-horiz-sup.

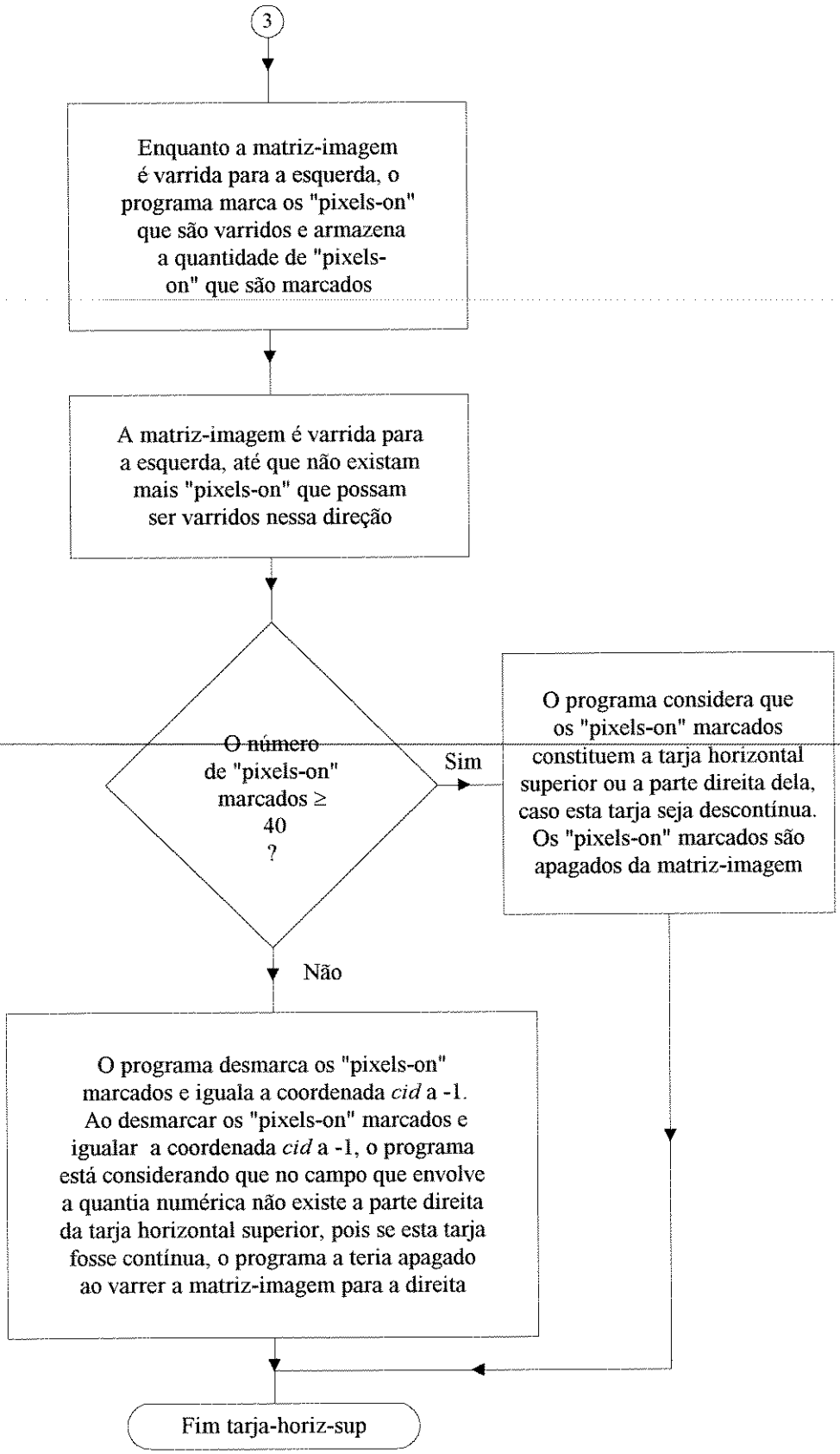


Figura 5.16(d) - Quarta parte do fluxograma da função tarja-horiz-sup.

Fluxograma da função tarja-horiz-inf

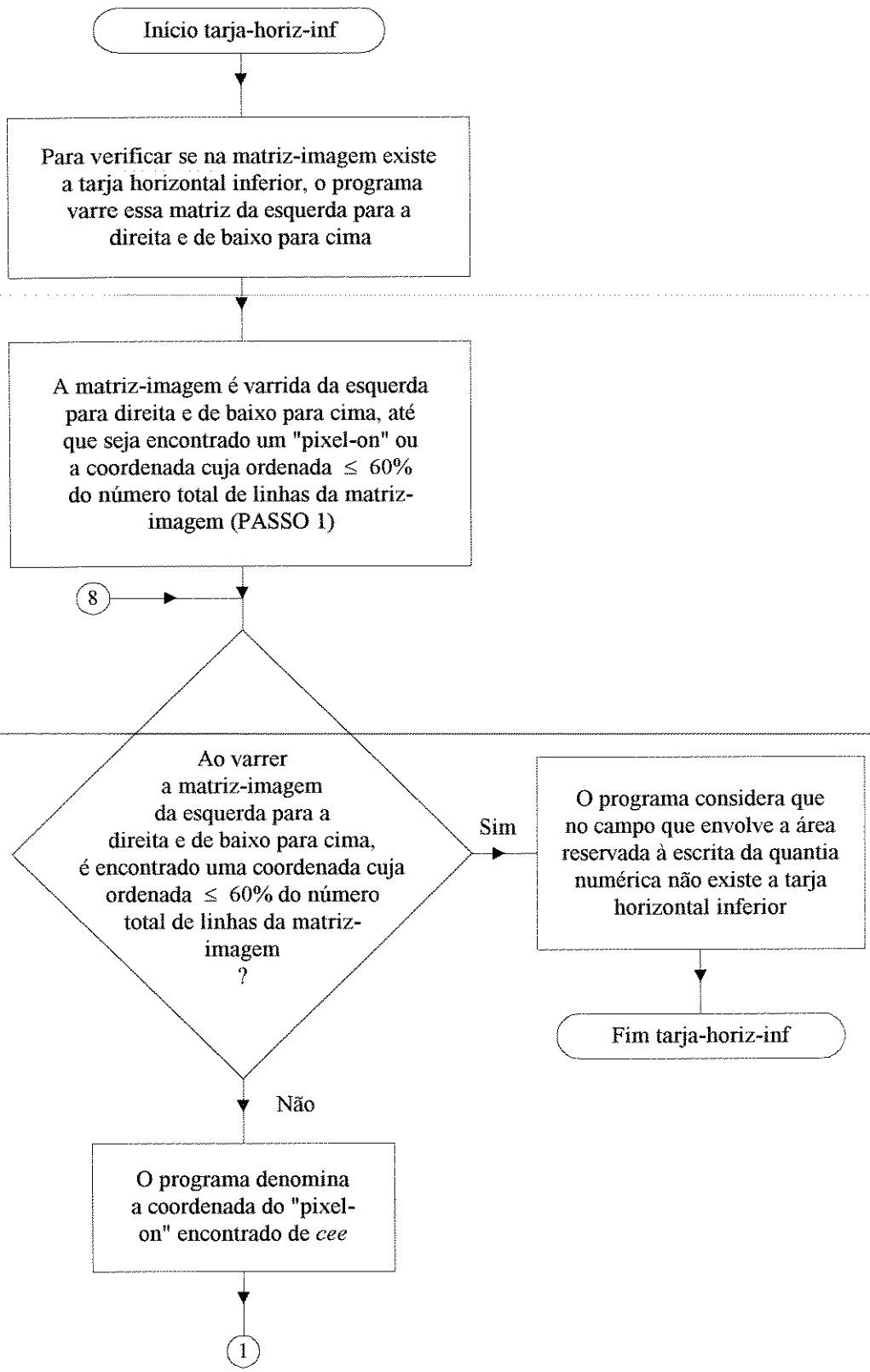


Figura 5.17(a) - Primeira parte do fluxograma da função tarja-horiz-inf.

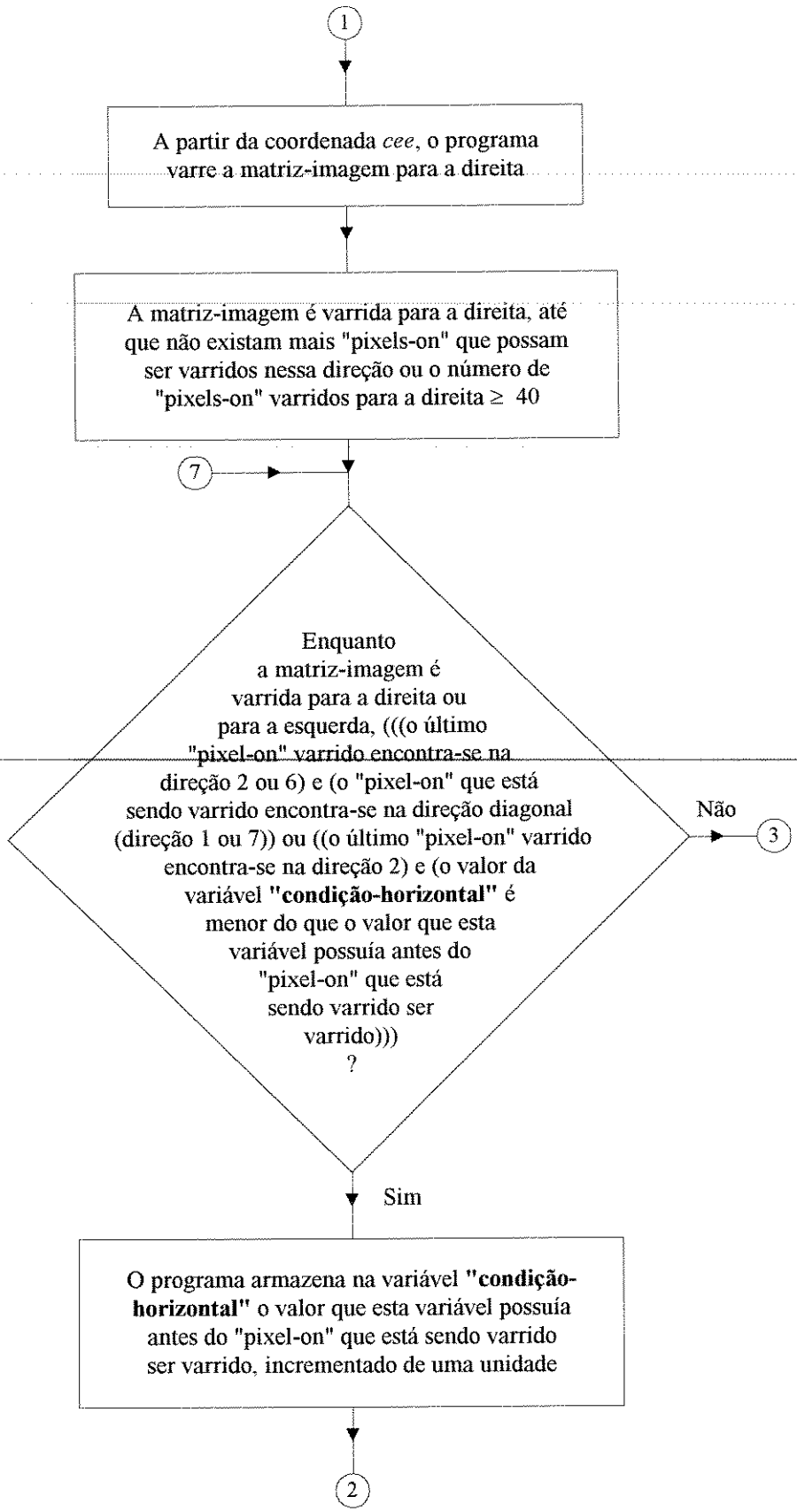


Figura 5.17(b) - Segunda parte do fluxograma da função tarja-horiz-inf.

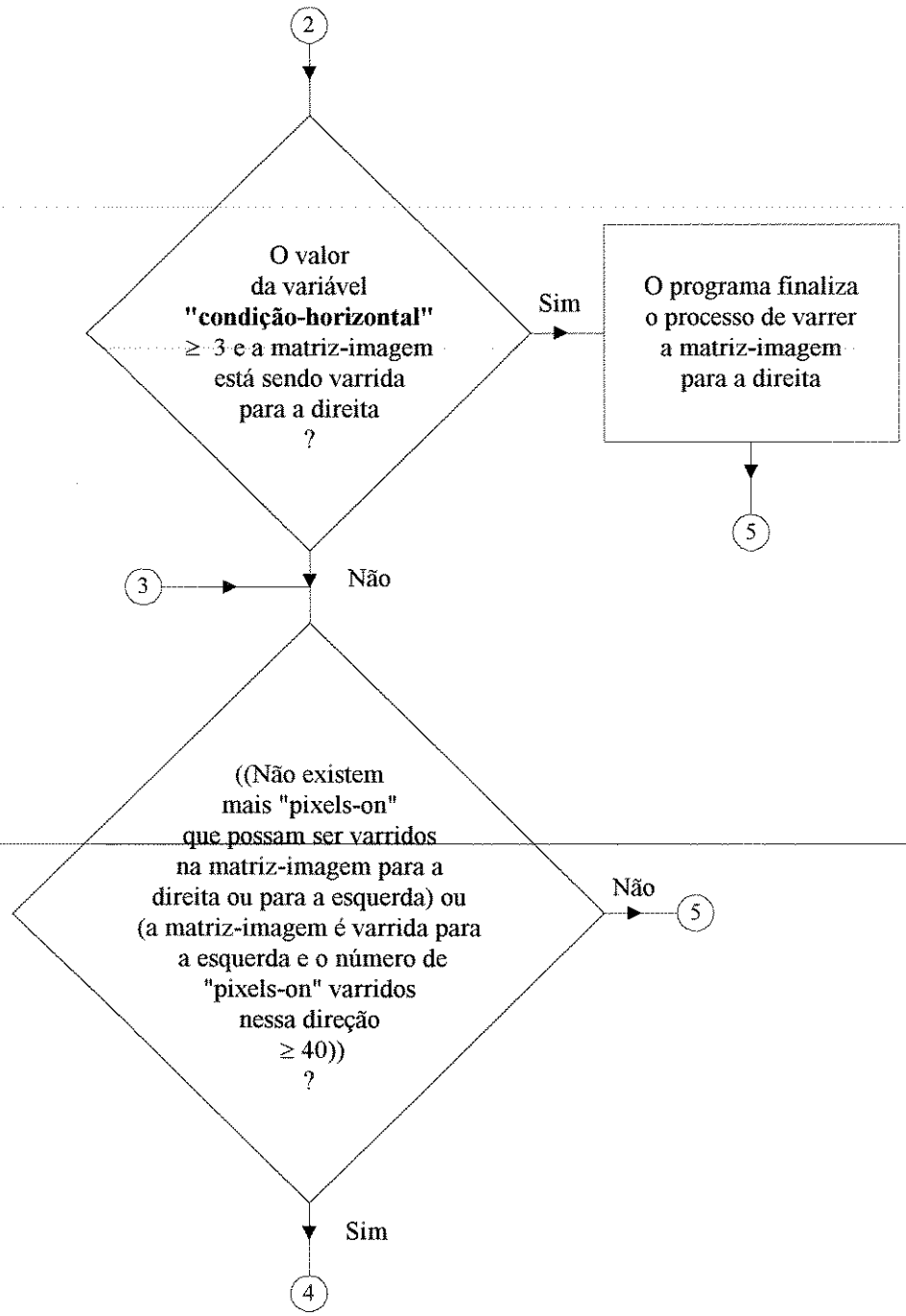


Figura 5.17(c) - Terceira parte do fluxograma da função tarja-horiz-inf.

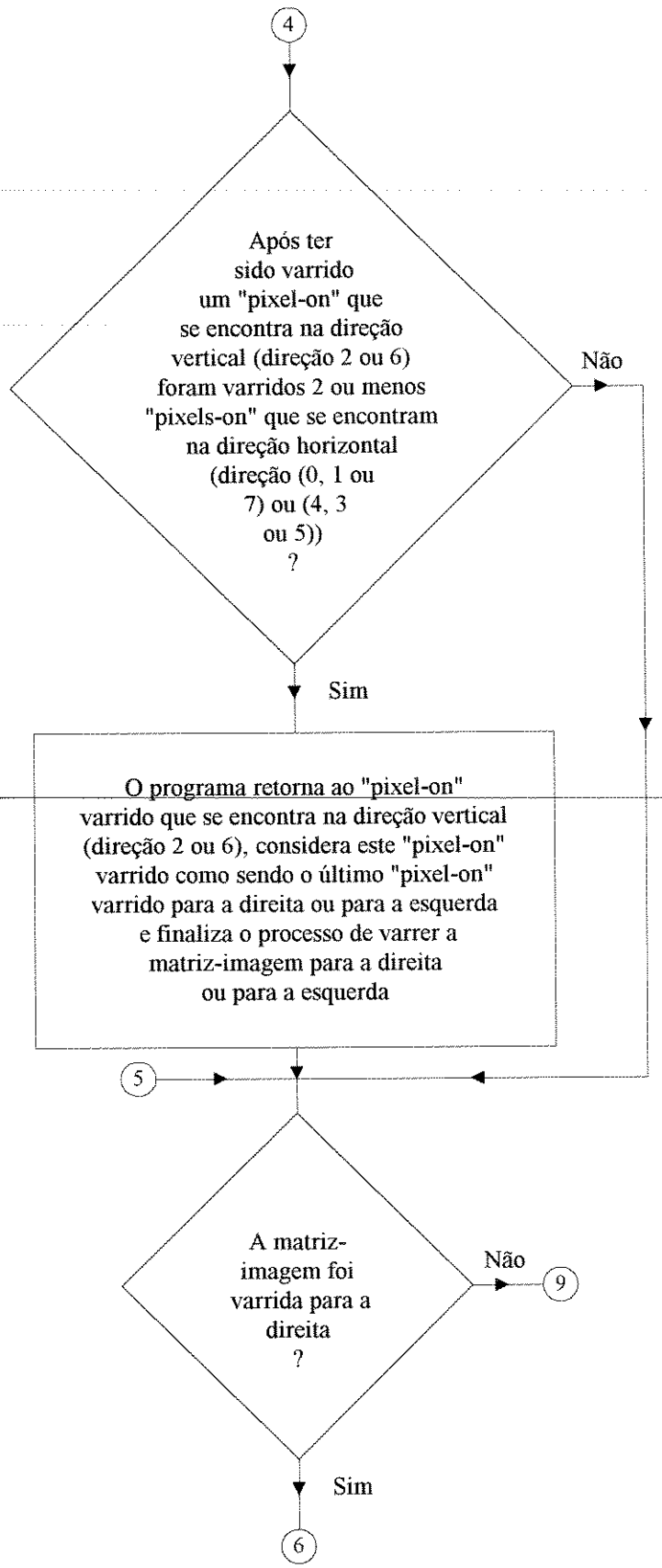


Figura 5.17(d) - Quarta parte do fluxograma da função tarja-horiz-inf.

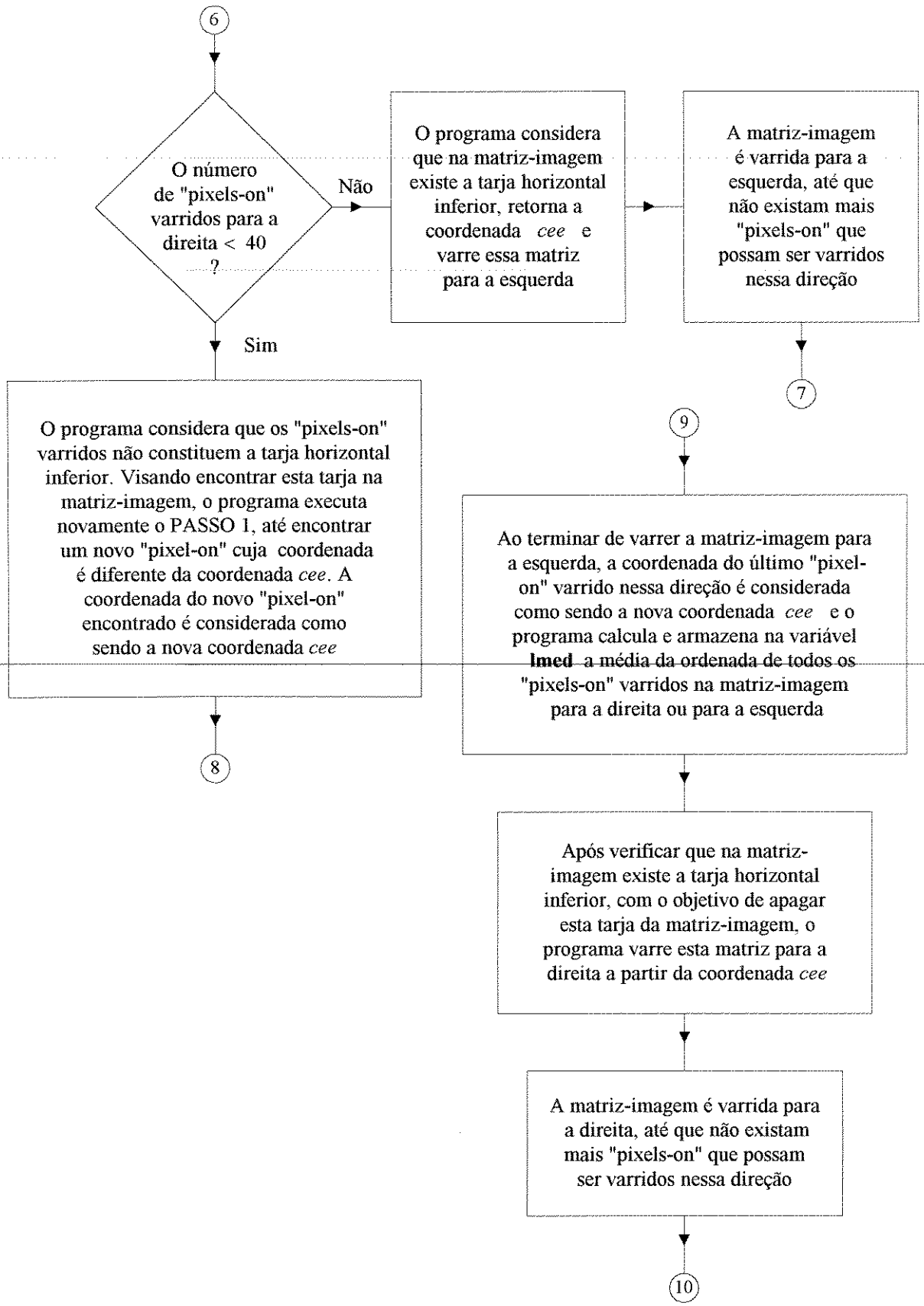


Figura 5.17(e) - Quinta parte do fluxograma da função tarja-horiz-inf.

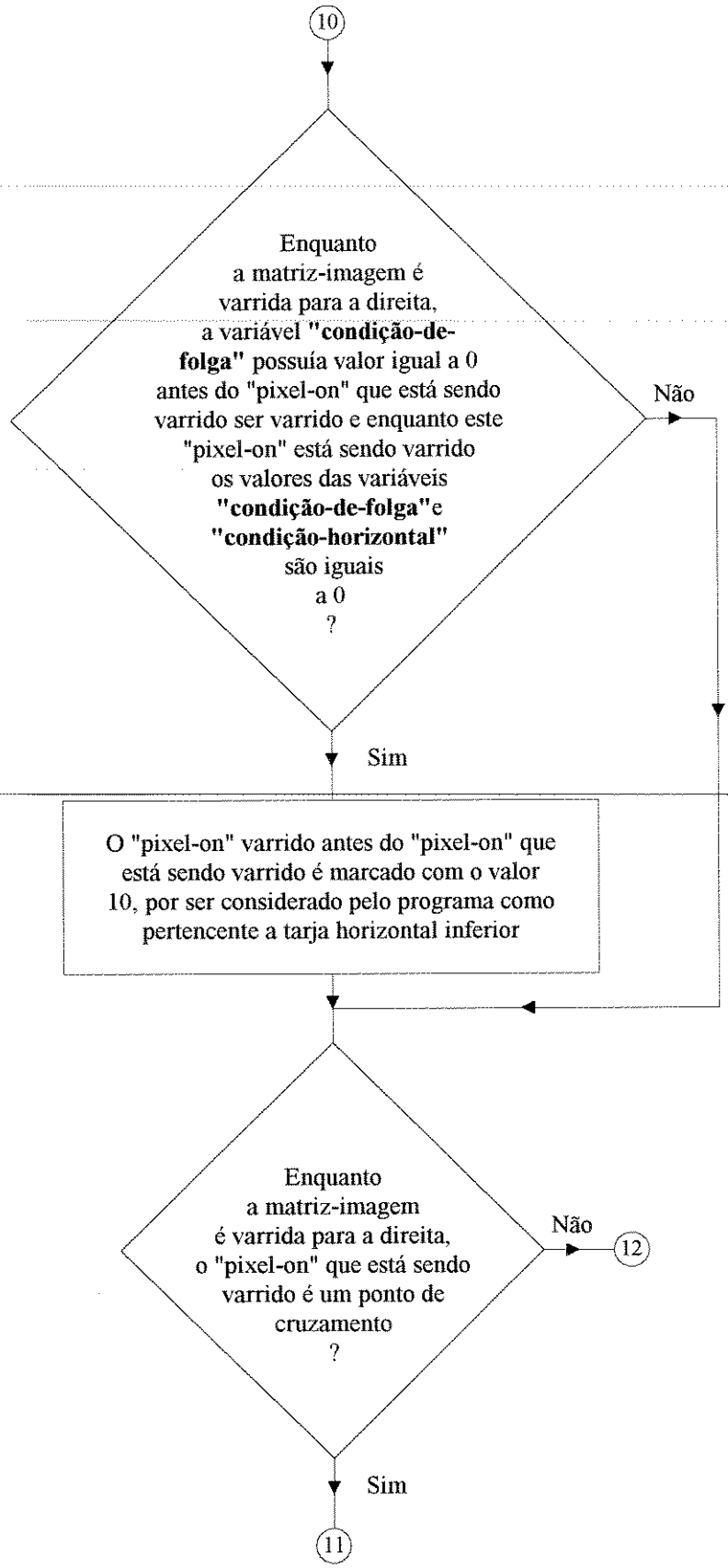


Figura 5.17(f) - Sexta parte do fluxograma da função tarja-horiz-inf.

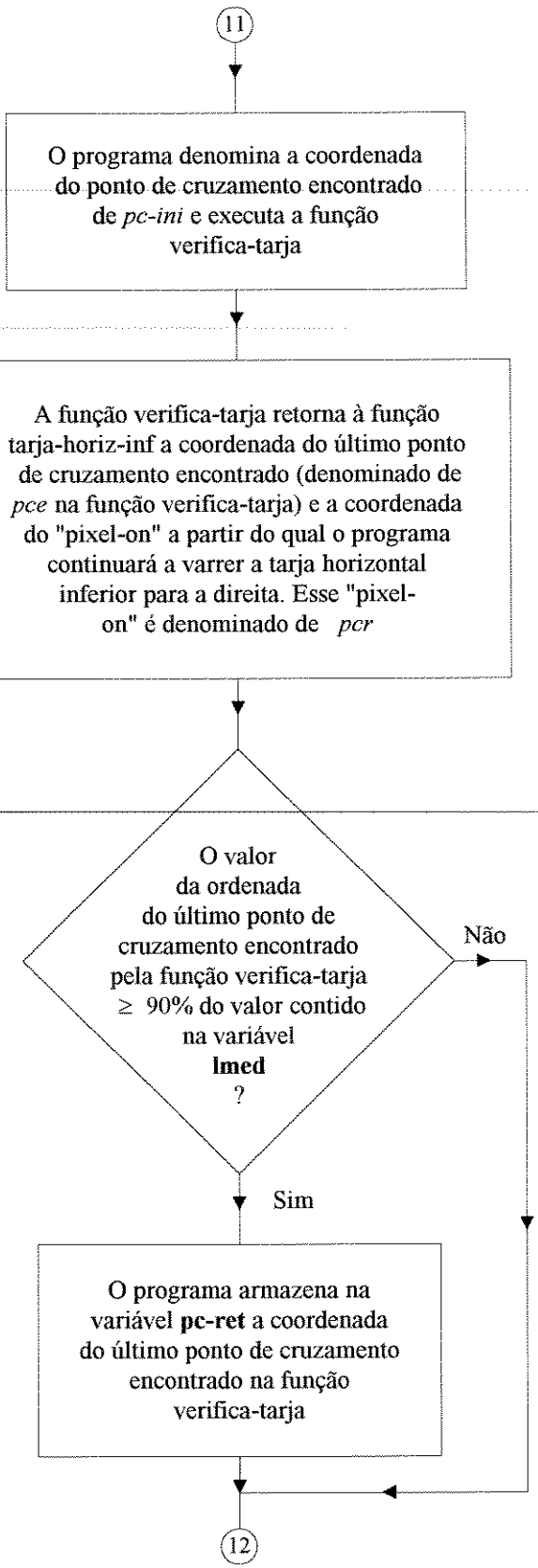


Figura 5.17(g) - Sétima parte do fluxograma da função tarja-horiz-inf.

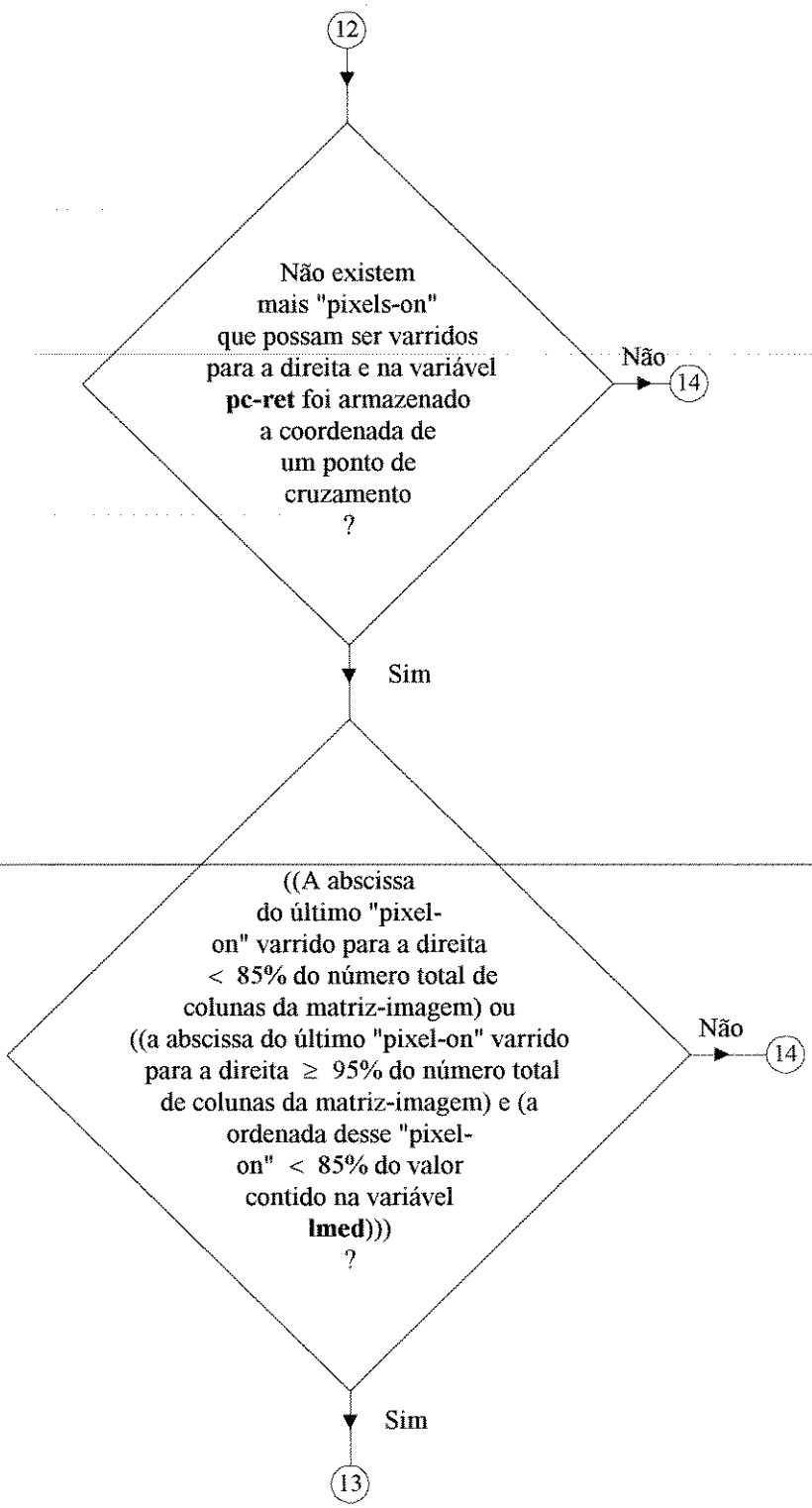


Figura 5.17(h) - Oitava parte do fluxograma da função tarja-horiz-inf.

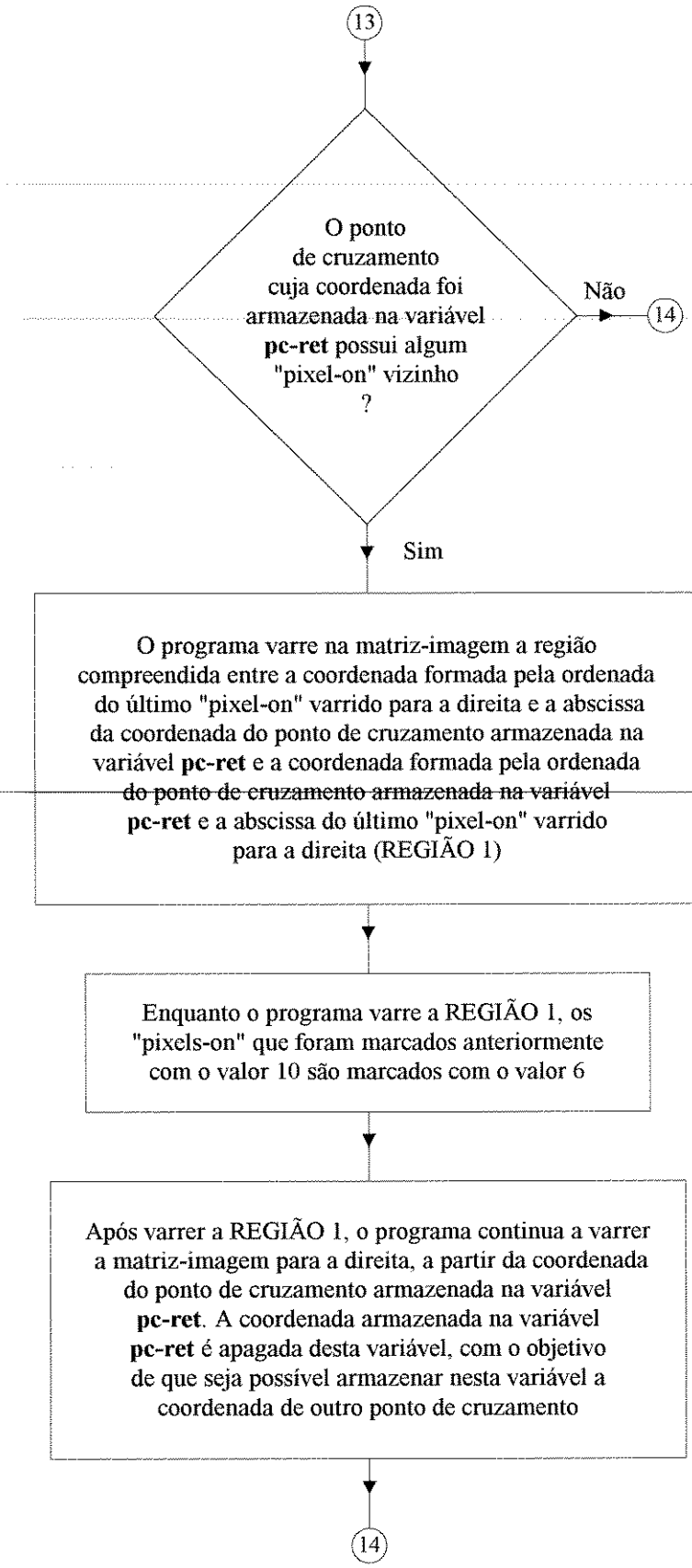


Figura 5.17(i) - Nona parte do fluxograma da função tarja-horiz-inf.

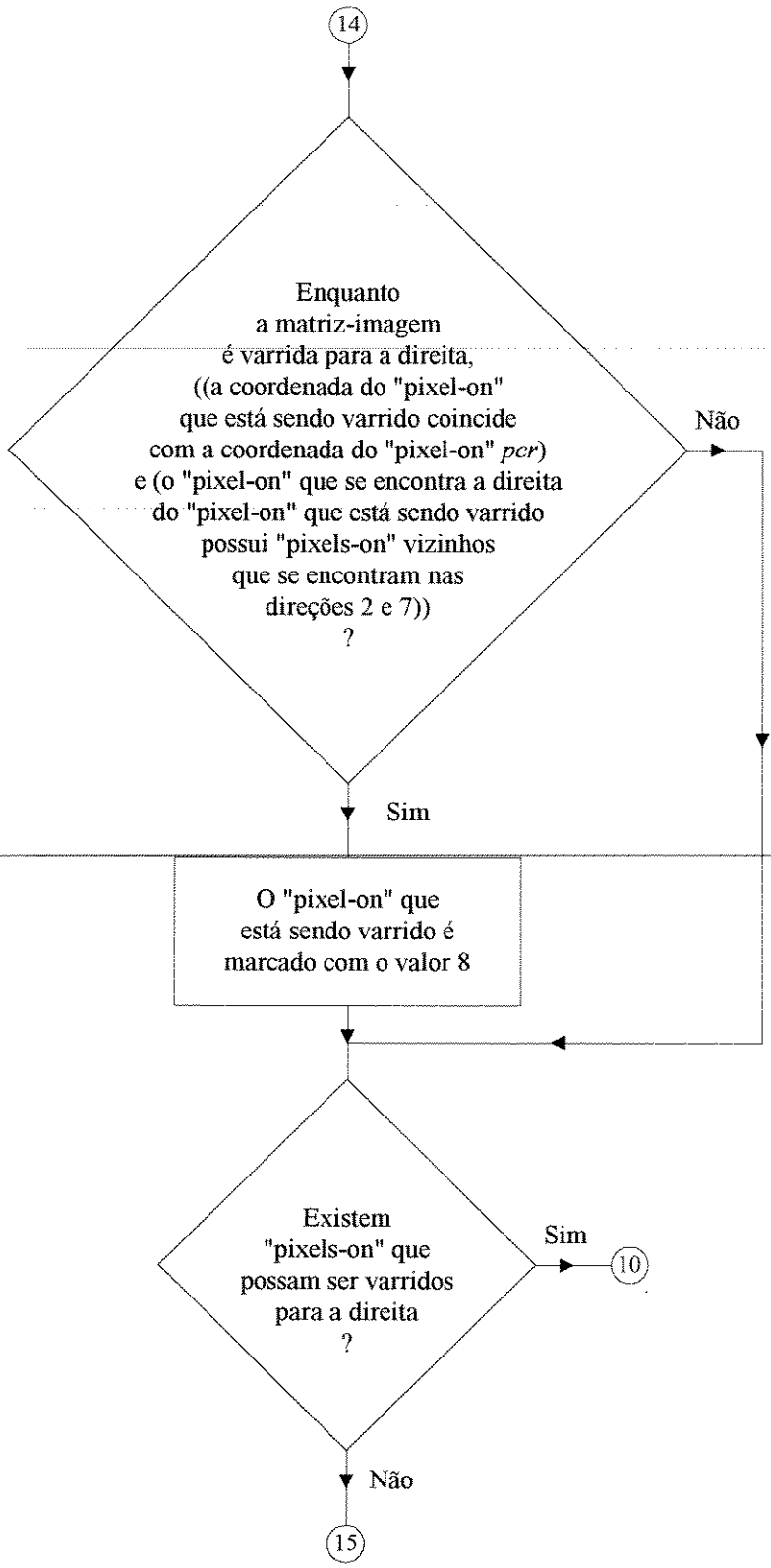


Figura 5.17(j) - Décima parte do fluxograma da função tarja-horiz-inf.

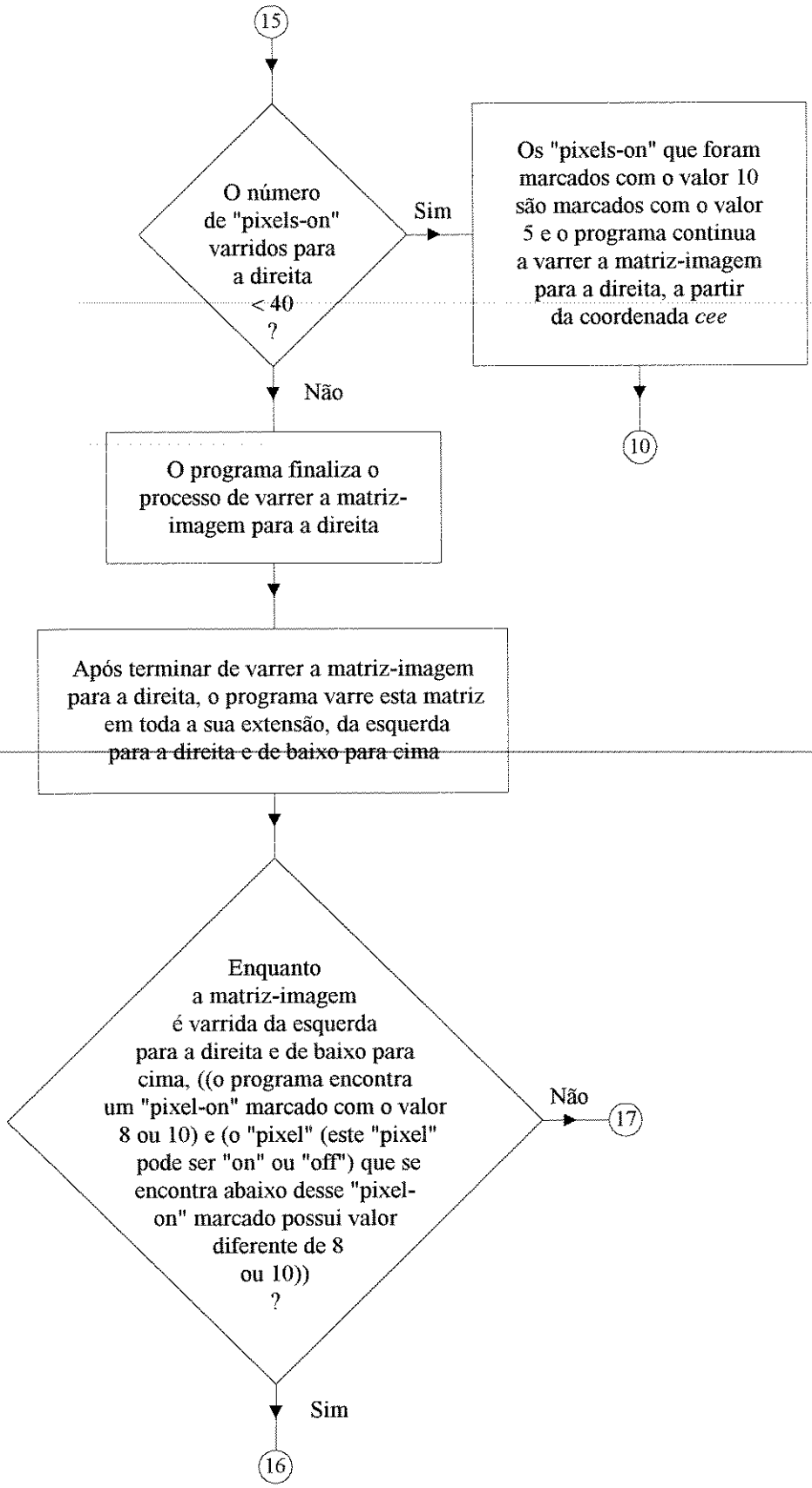


Figura 5.17(k) - Décima primeira parte do fluxograma da função tarja-horiz-inf.

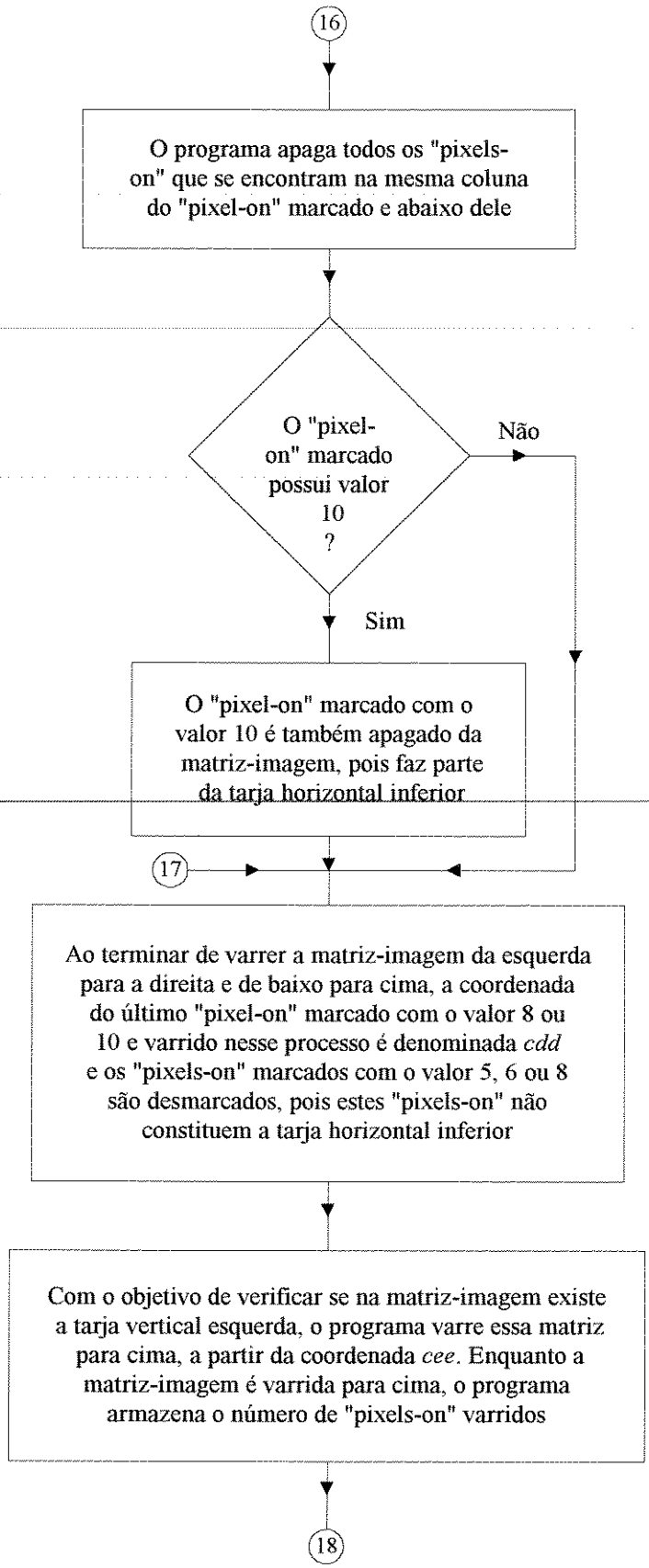


Figura 5.17(l) - Décima segunda parte do fluxograma da função tarja-horiz-inf.

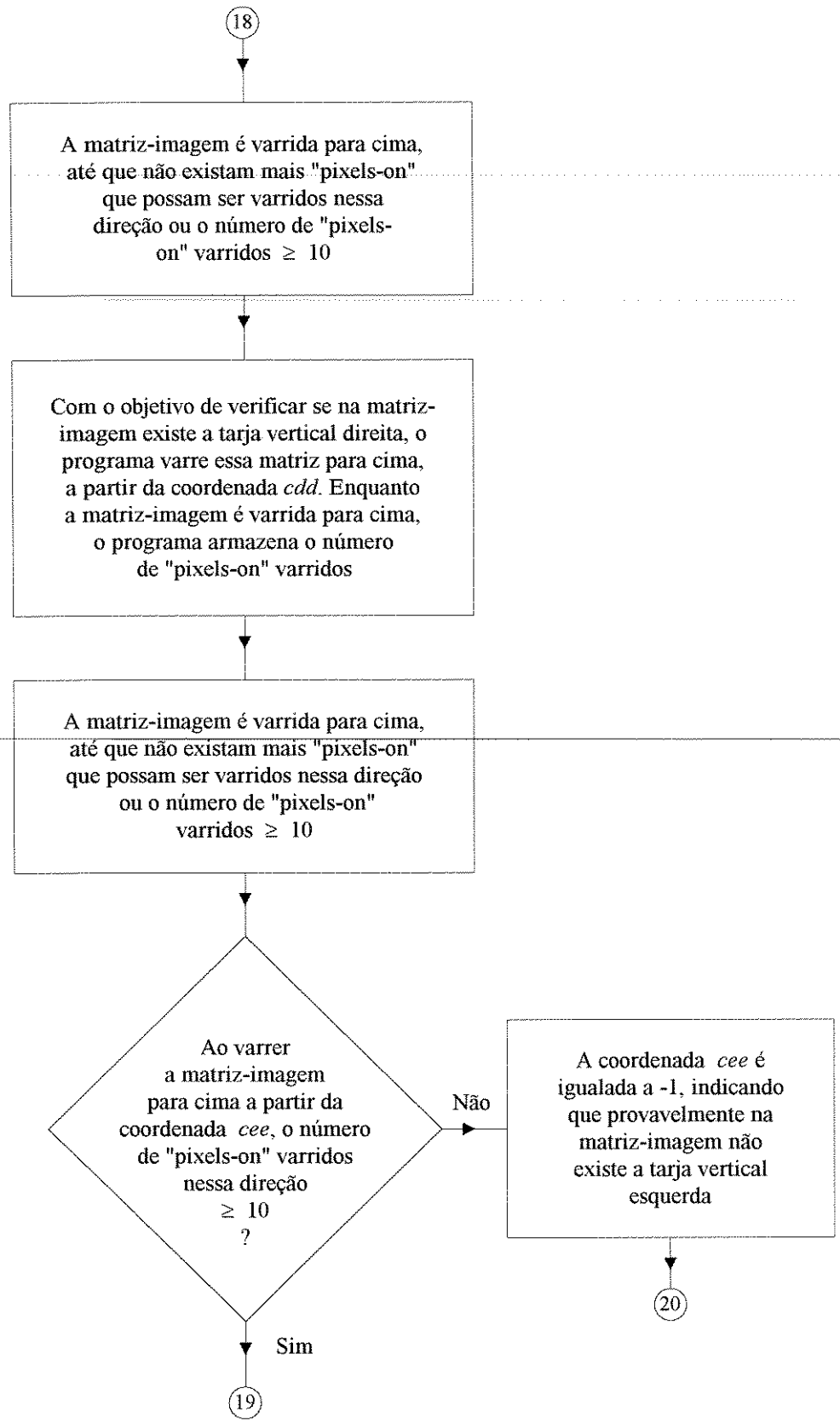


Figura 5.17(m) - Décima terceira parte do fluxograma da função tarja-horiz-inf.

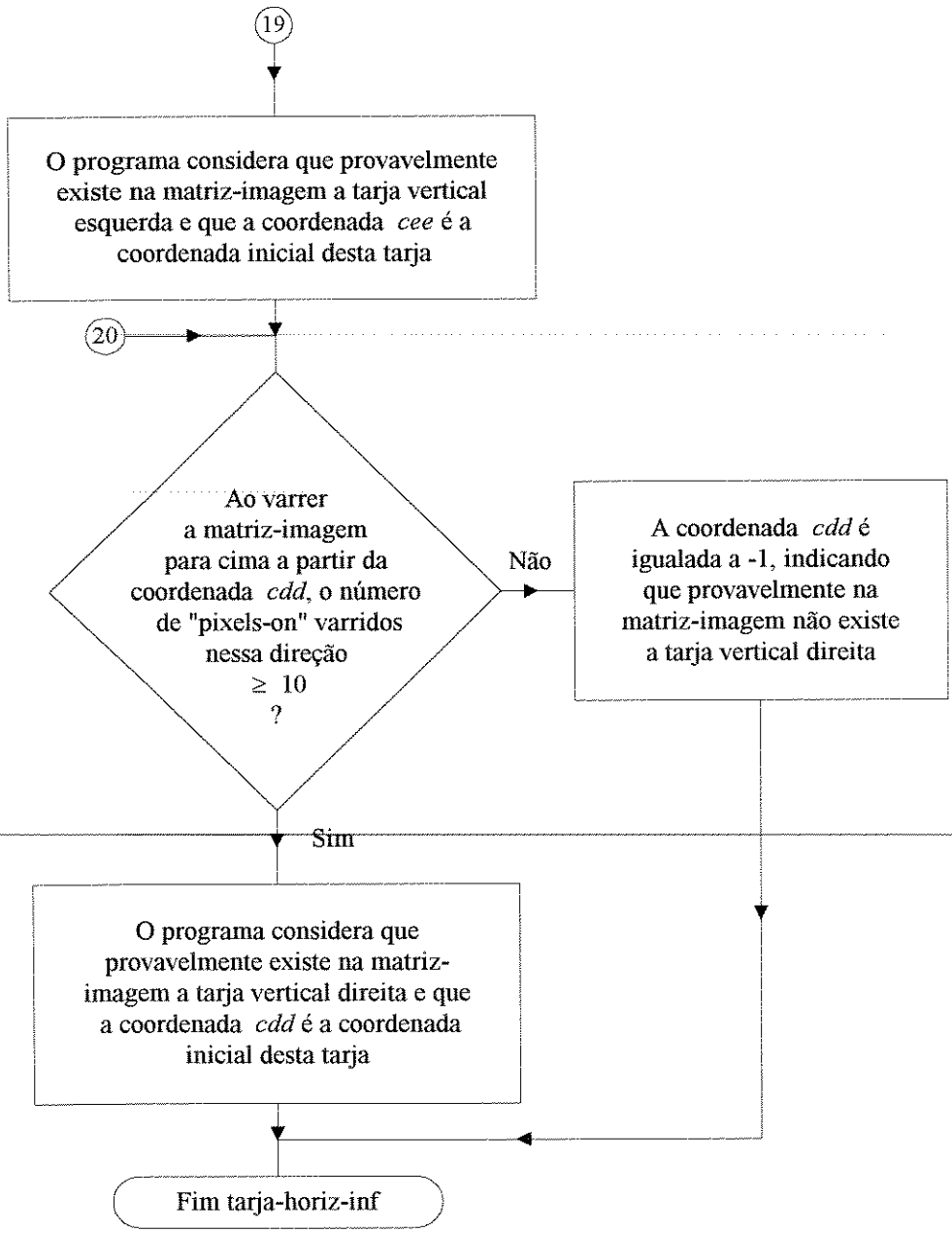


Figura 5.17(n) - Décima quarta parte do fluxograma da função tarja-horiz-inf.

Fluxograma da função verifica-tarja

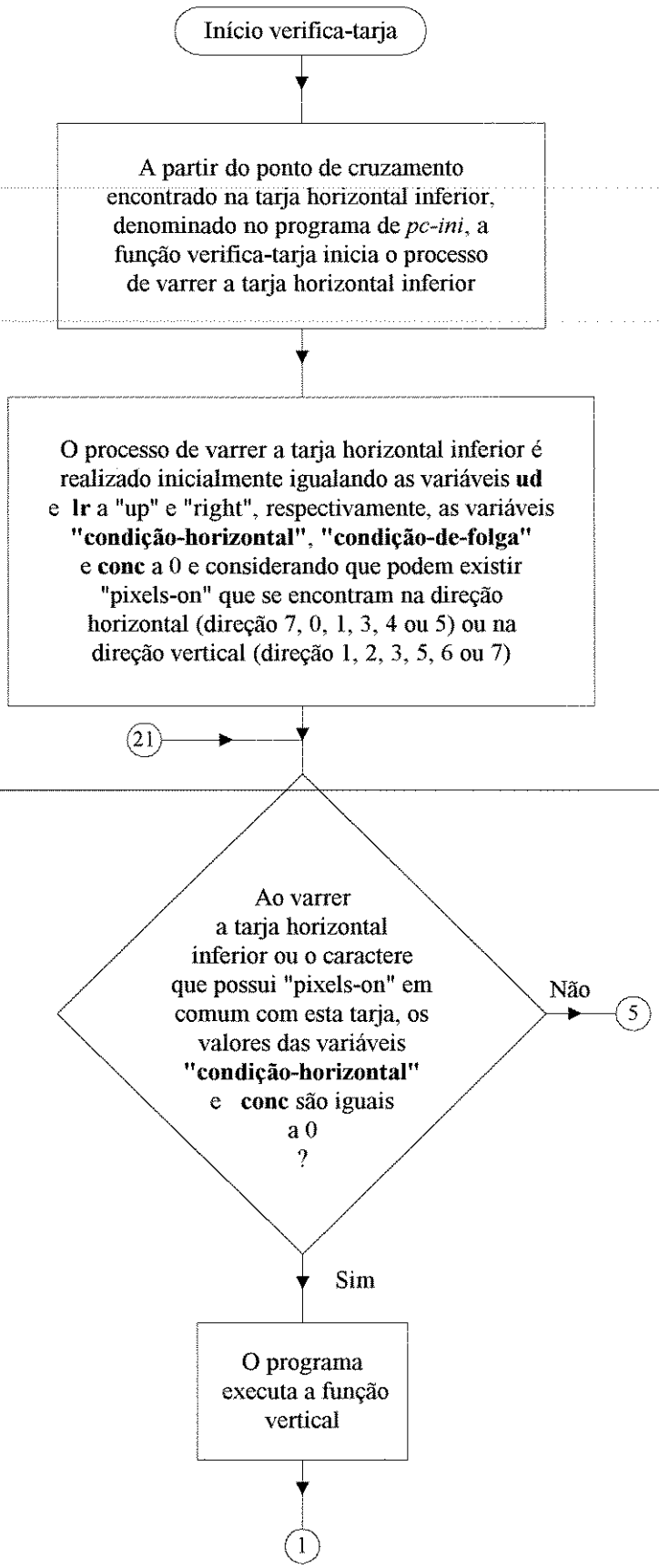


Figura 5.18(a) - Primeira parte do fluxograma da função verifica-tarja.

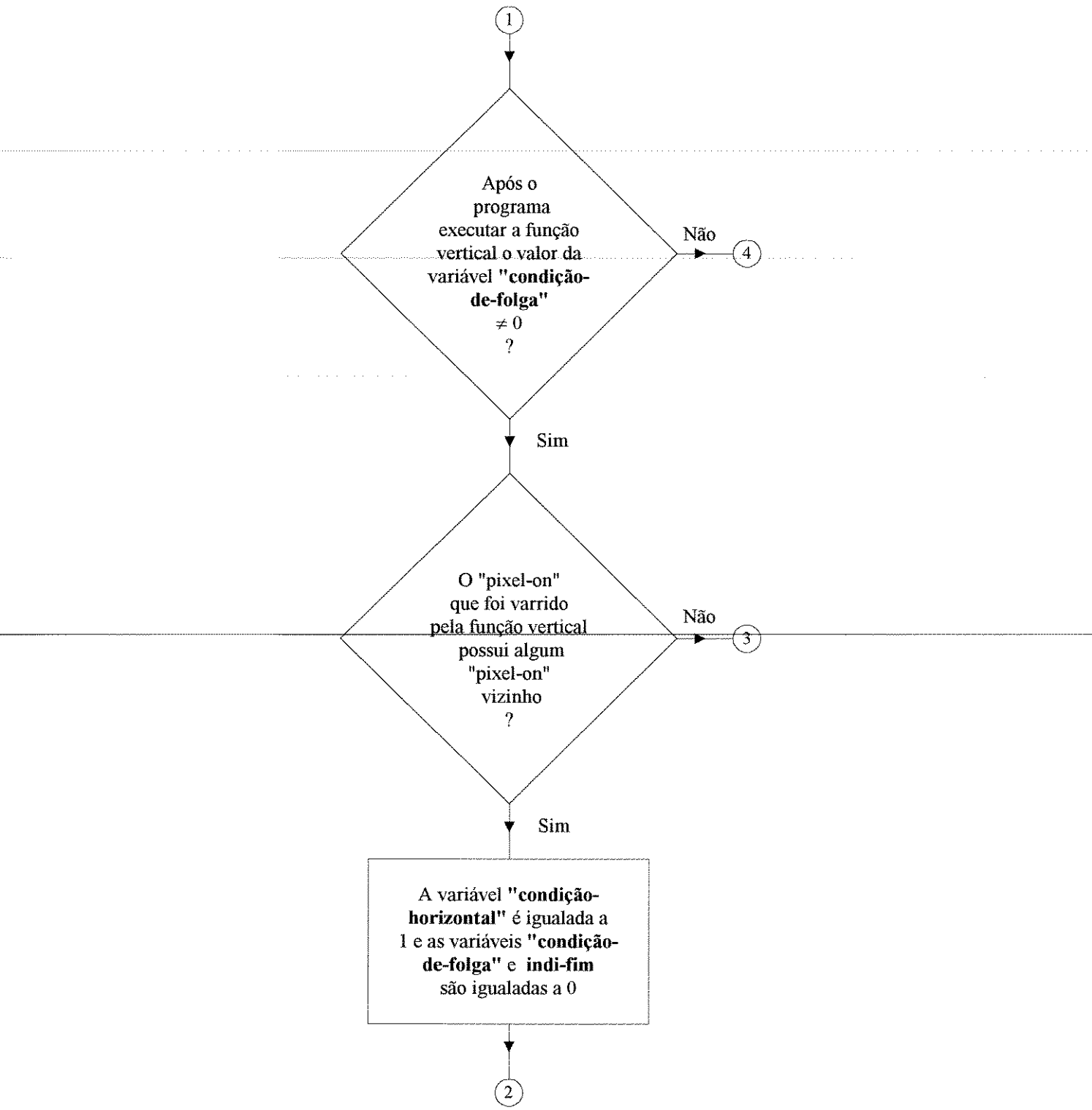


Figura 5.18(b) - Segunda parte do fluxograma da função verifica-tarja.

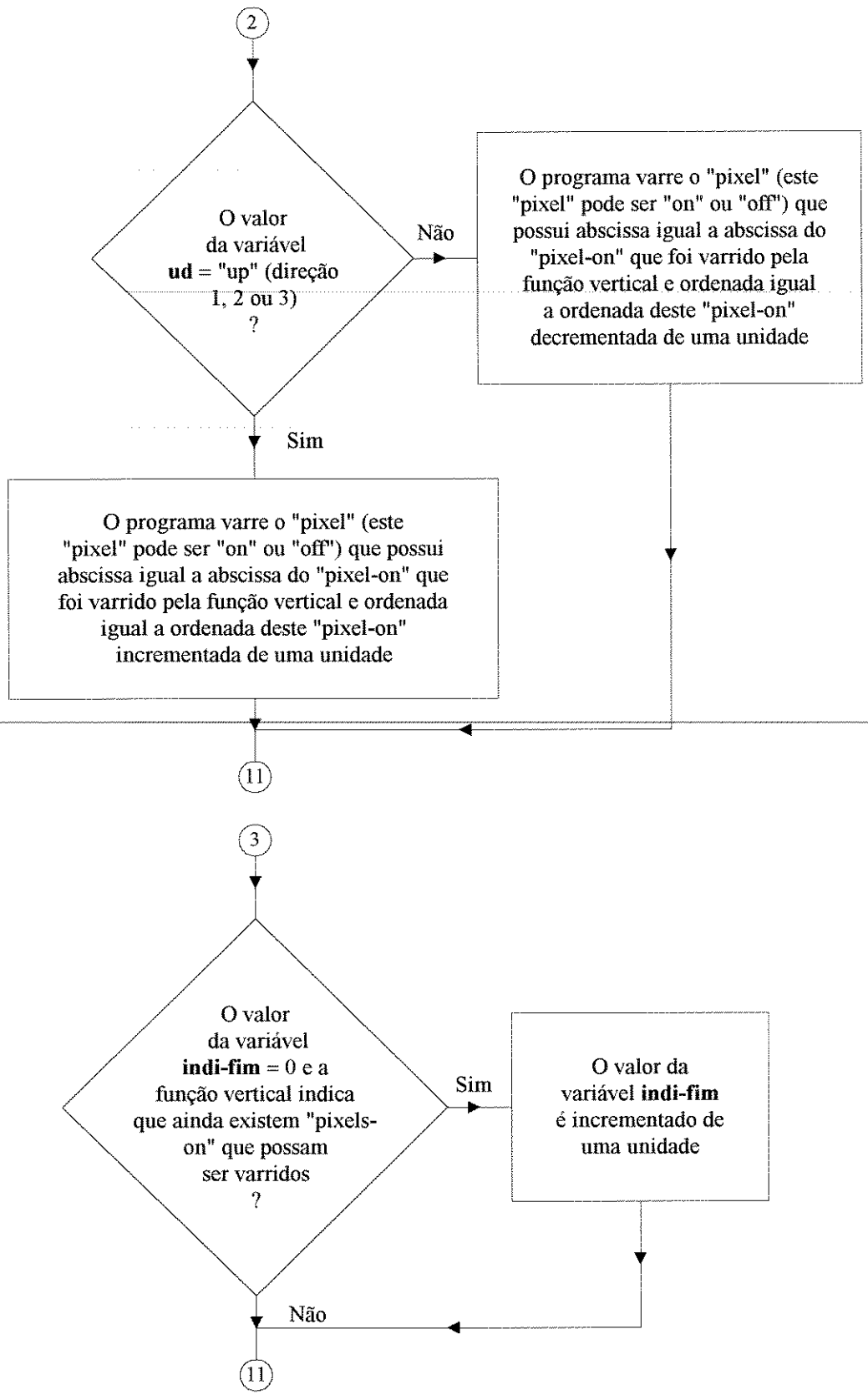


Figura 5.18(c) - Terceira parte do fluxograma da função verifica-tarja.

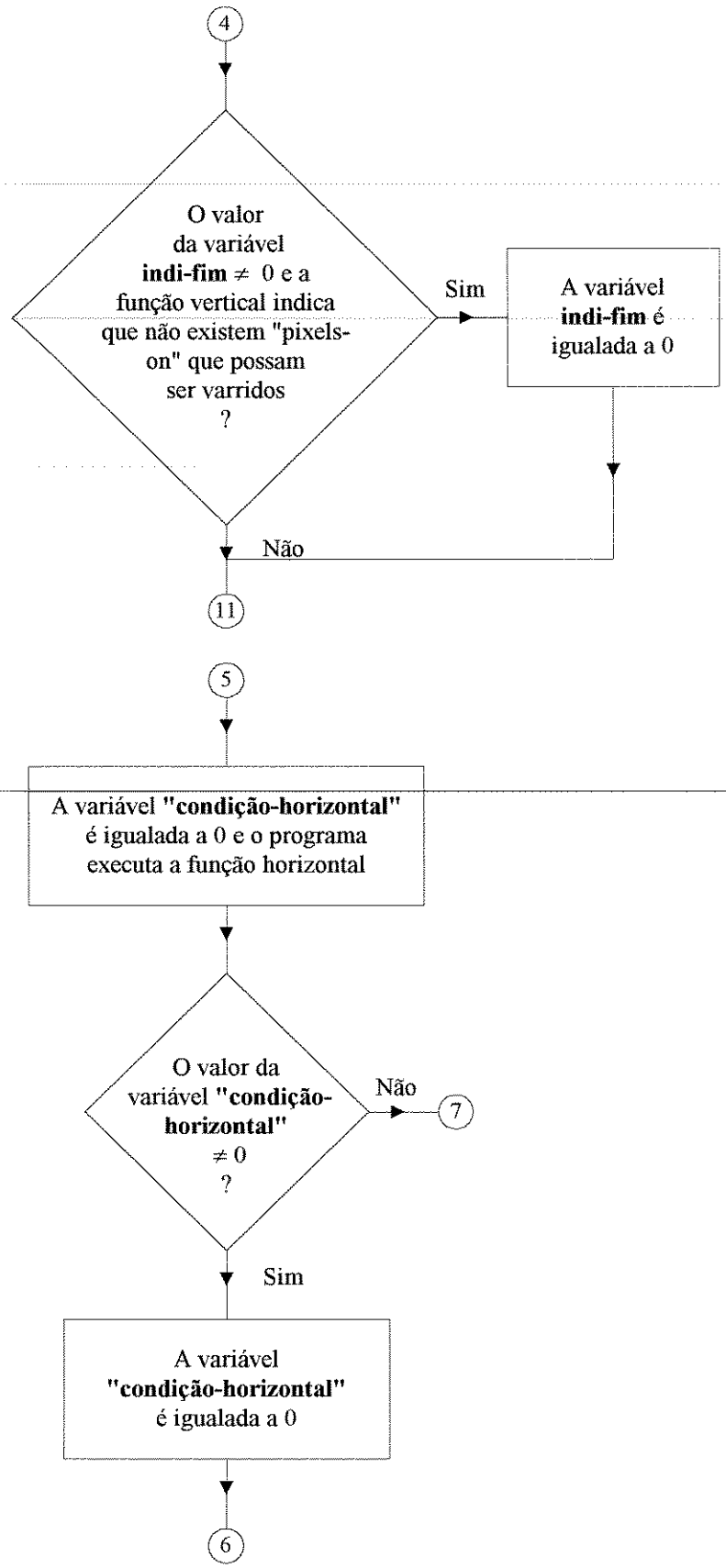


Figura 5.18(d) - Quarta parte do fluxograma da função verifica-tarja.

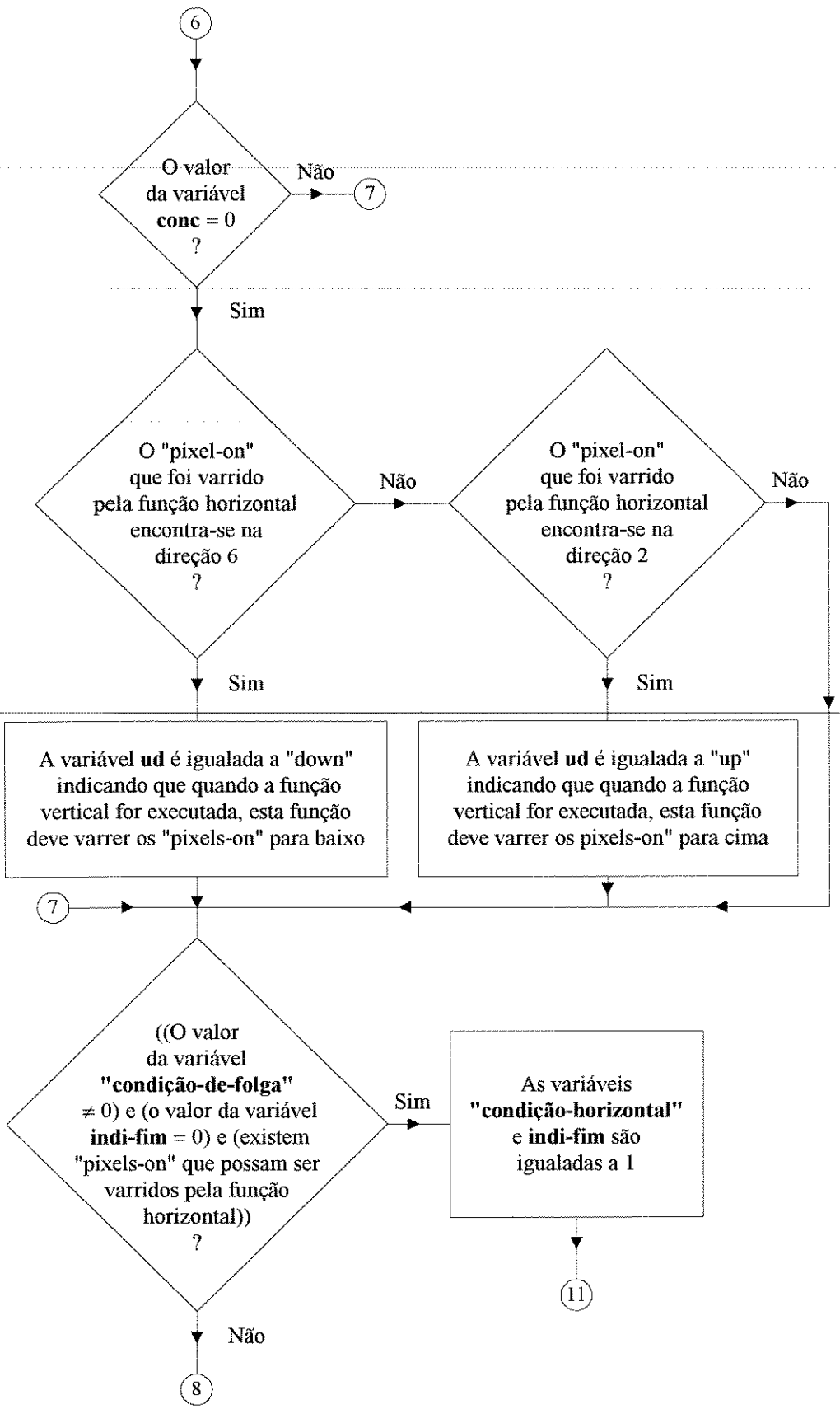


Figura 5.18(e) - Quinta parte do fluxograma da função verifica-tarja.

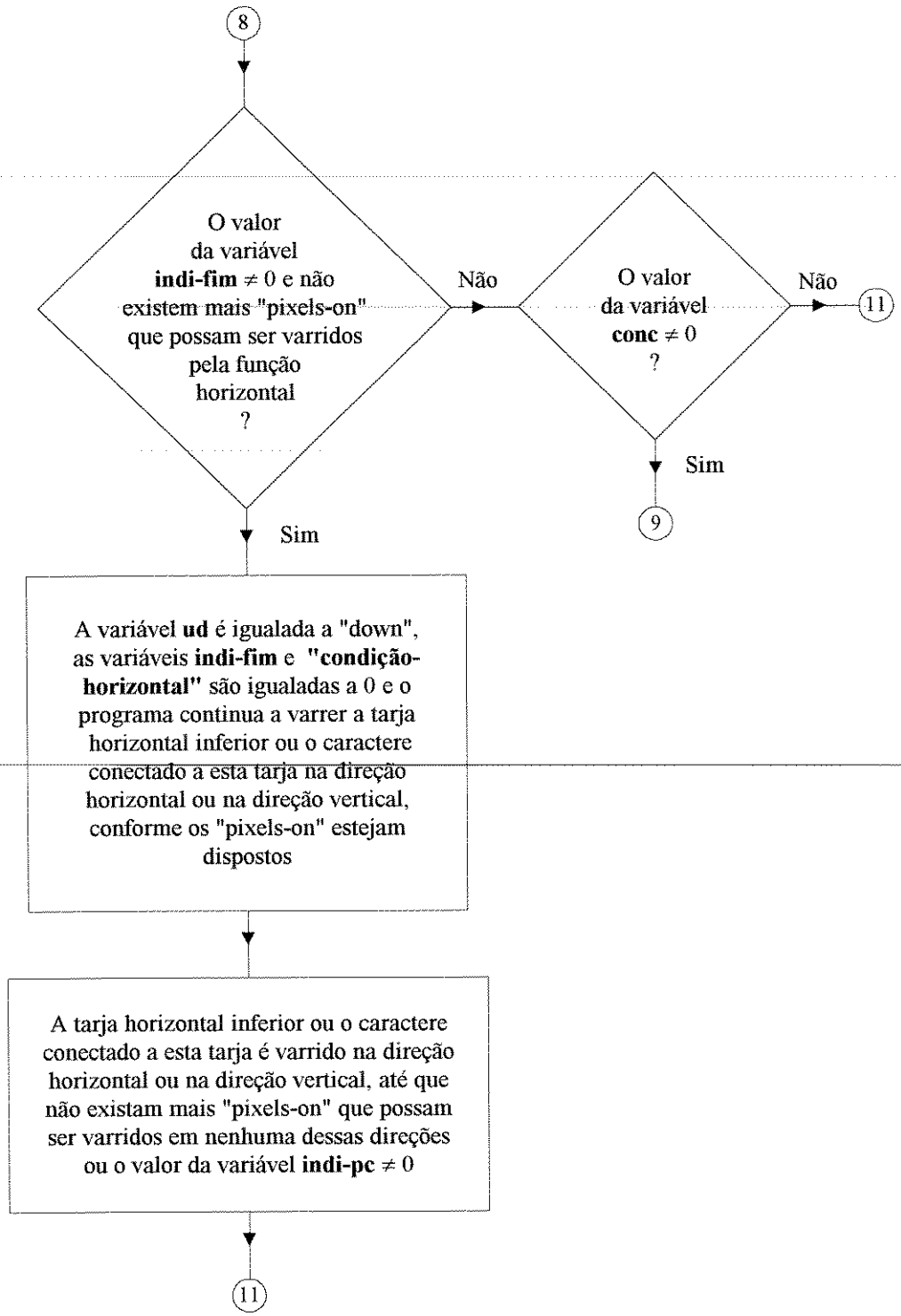


Figura 5.18(f) - Sexta parte do fluxograma da função verifica-tarja.

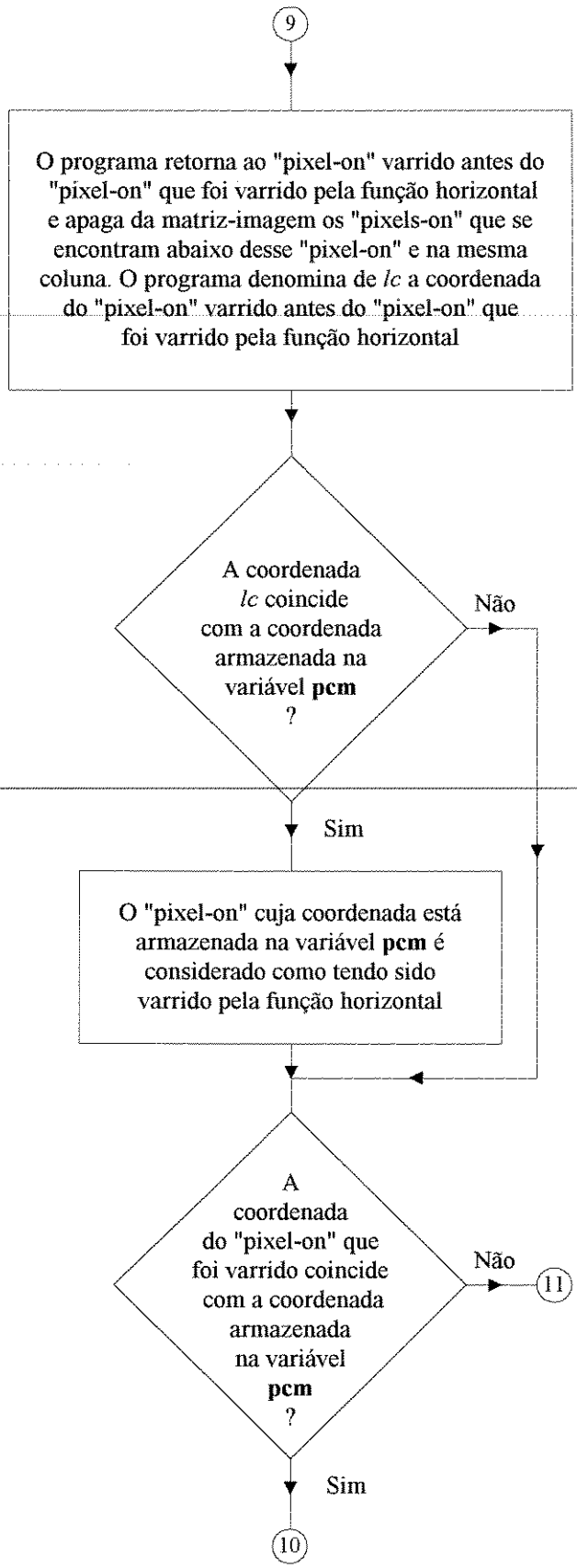


Figura 5.18(g) - Sétima parte do fluxograma da função verifica-tarja.

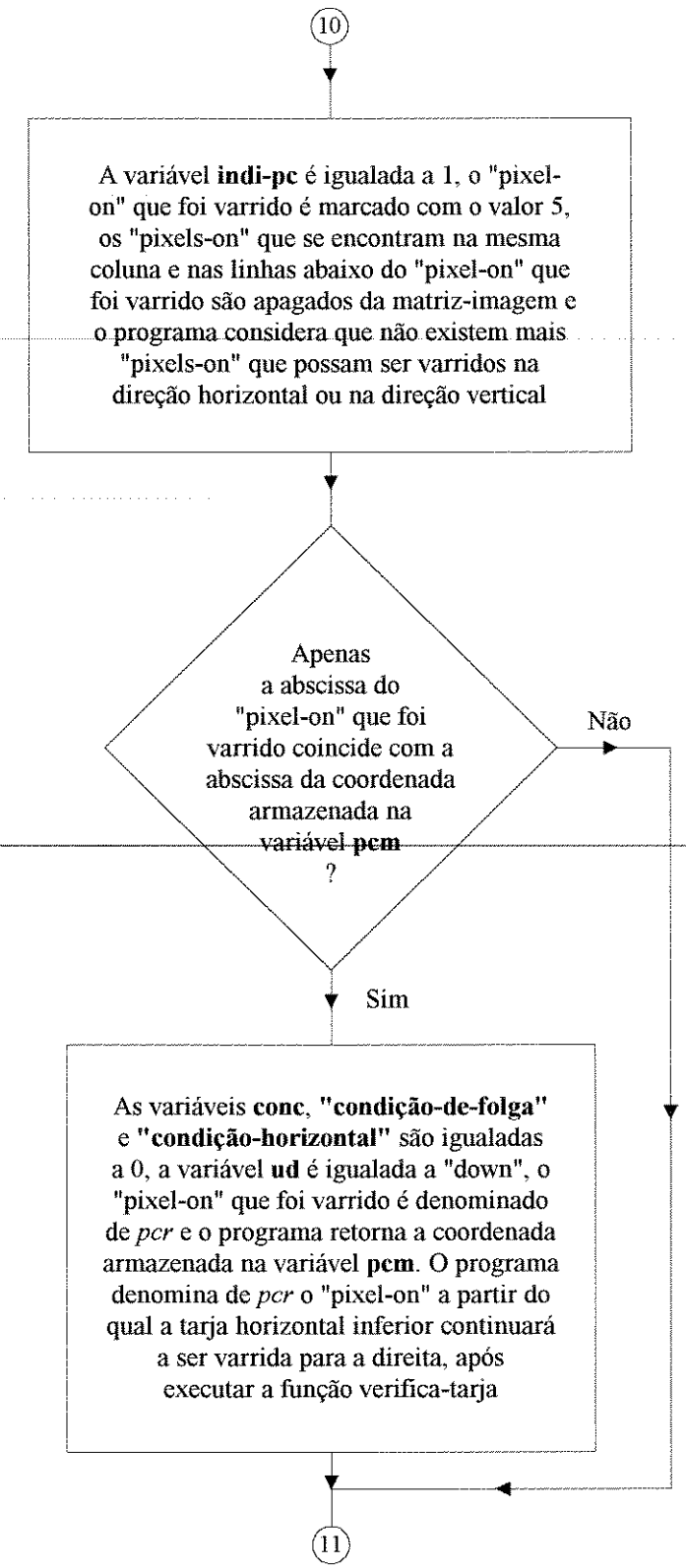


Figura 5.18(h) - Oitava parte do fluxograma da função verifica-tarja.

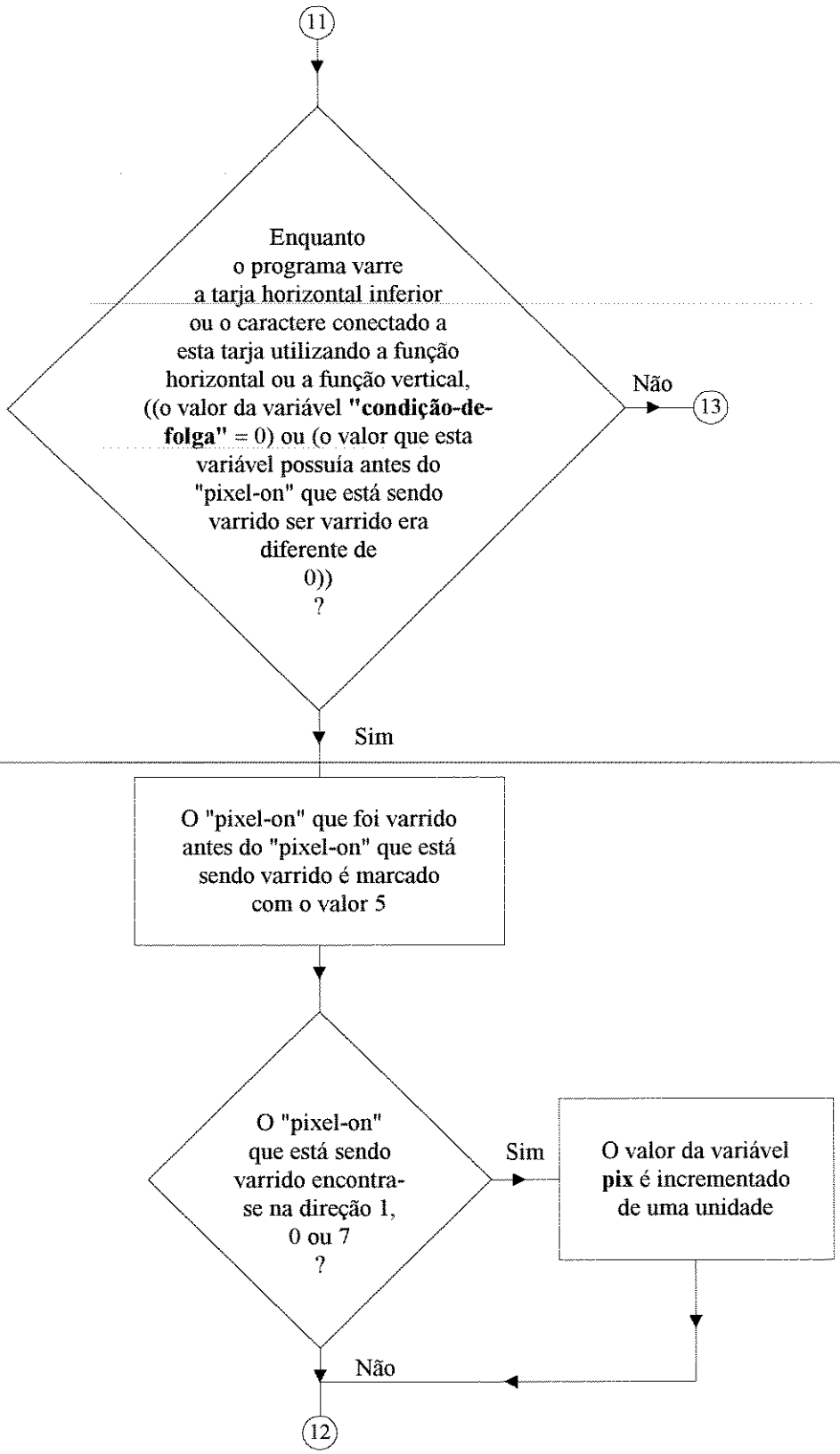


Figura 5.18(i) - Nona parte do fluxograma da função verifica-tarja.

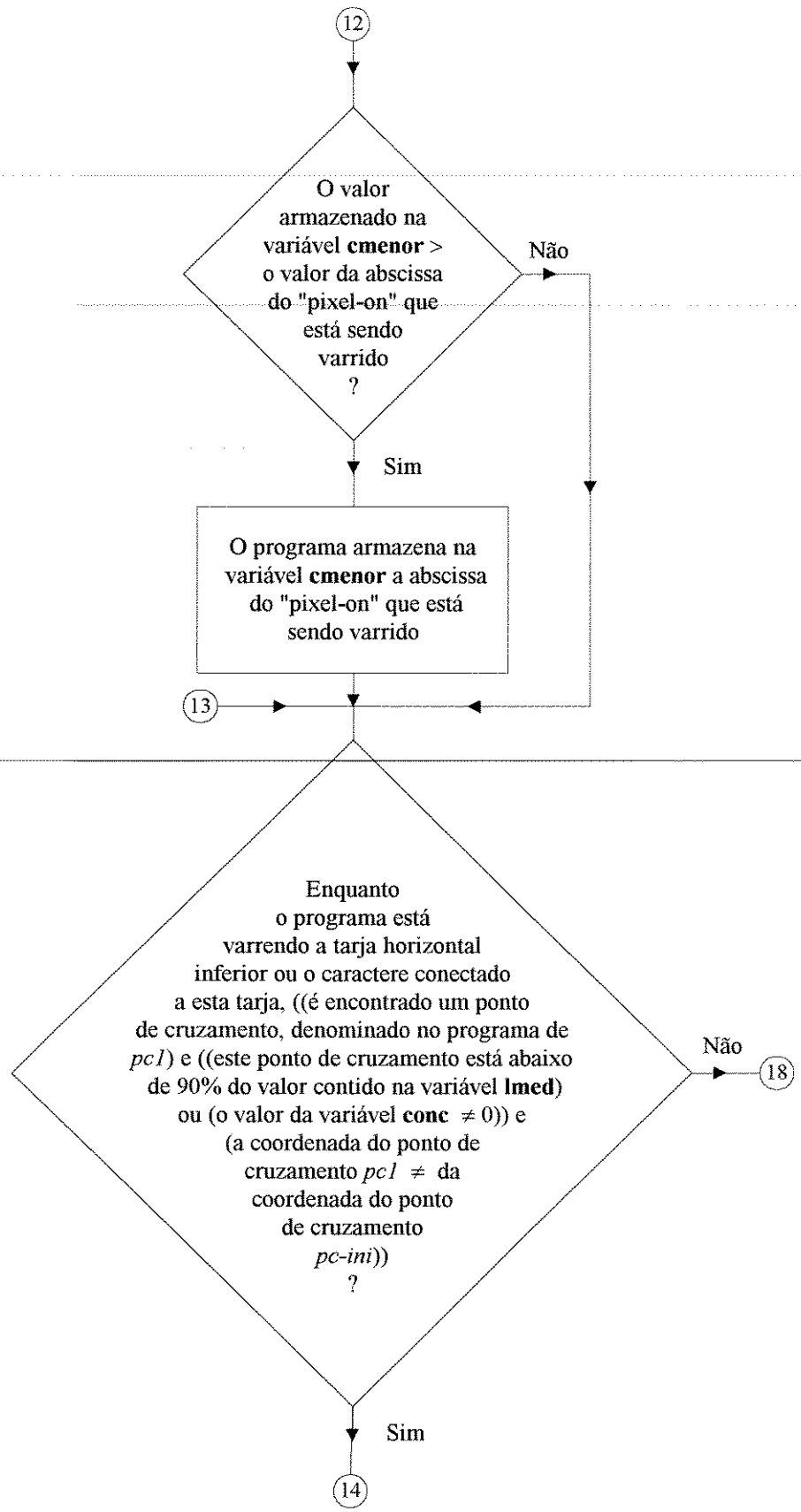


Figura 5.18(j) - Décima parte do fluxograma da função verifica-tarja.

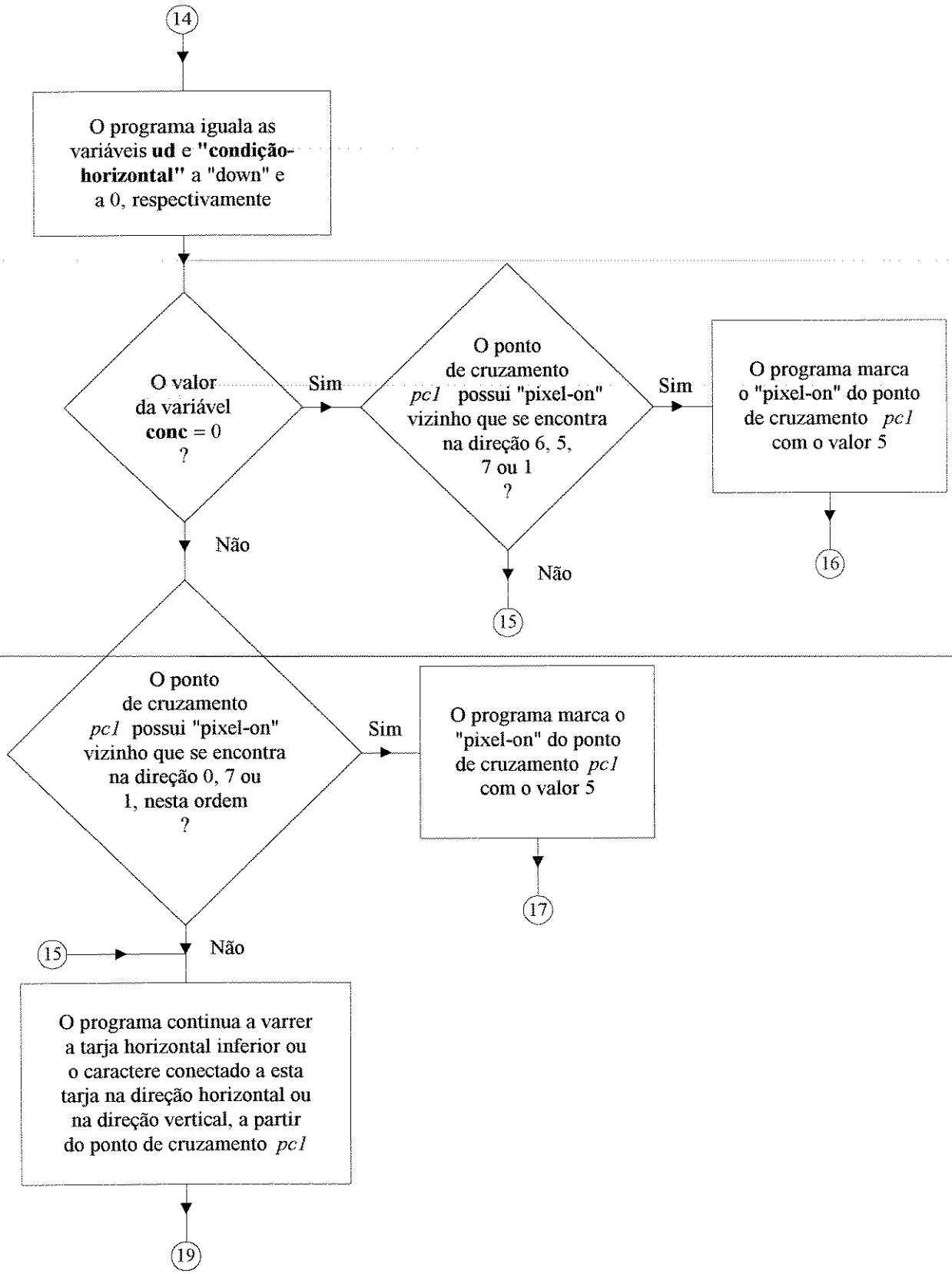


Figura 5.18(k) - Décima primeira parte do fluxograma da função verifica-tarja.

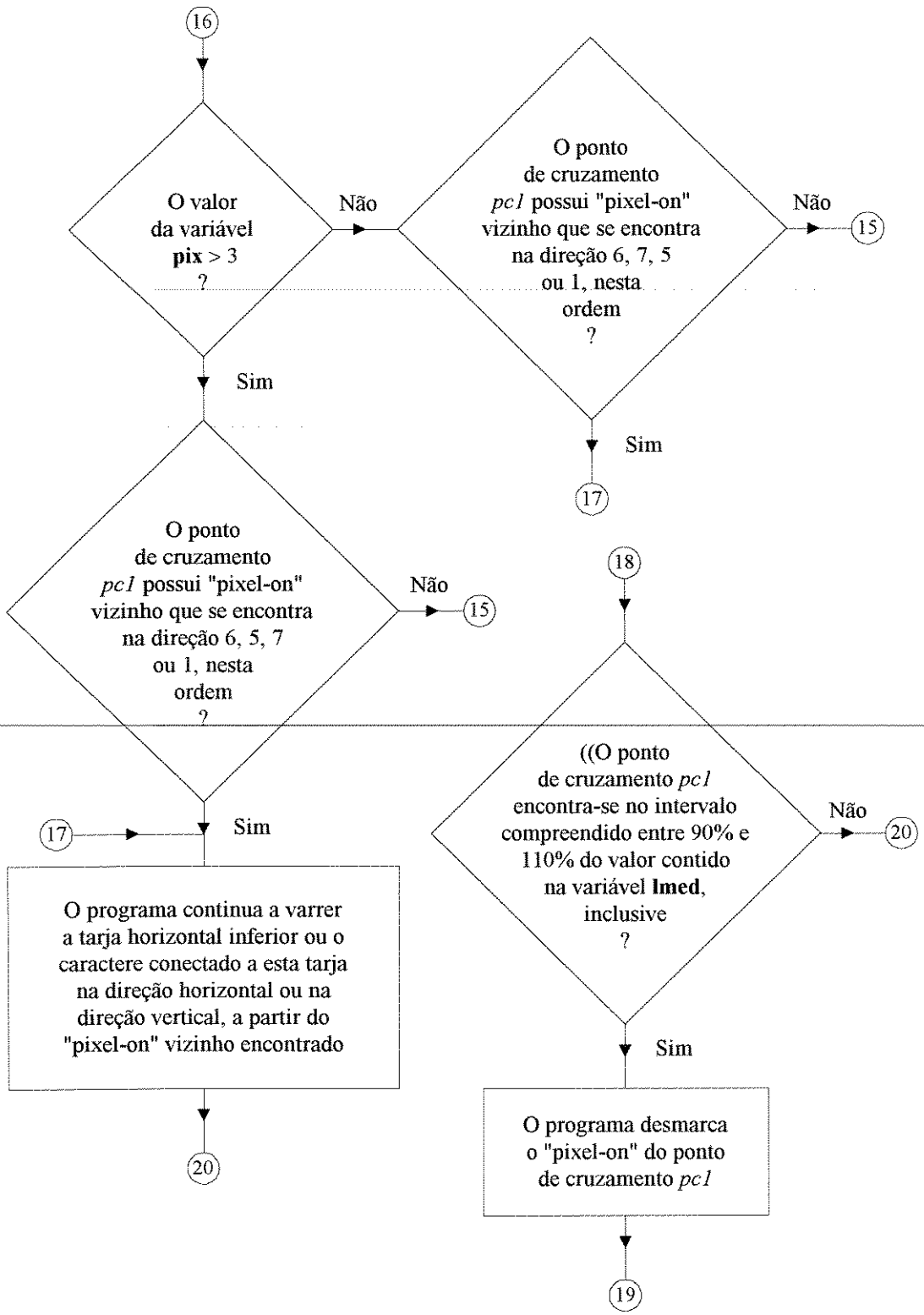


Figura 5.18(l) - Décima segunda parte do fluxograma da função verifica-tarja.

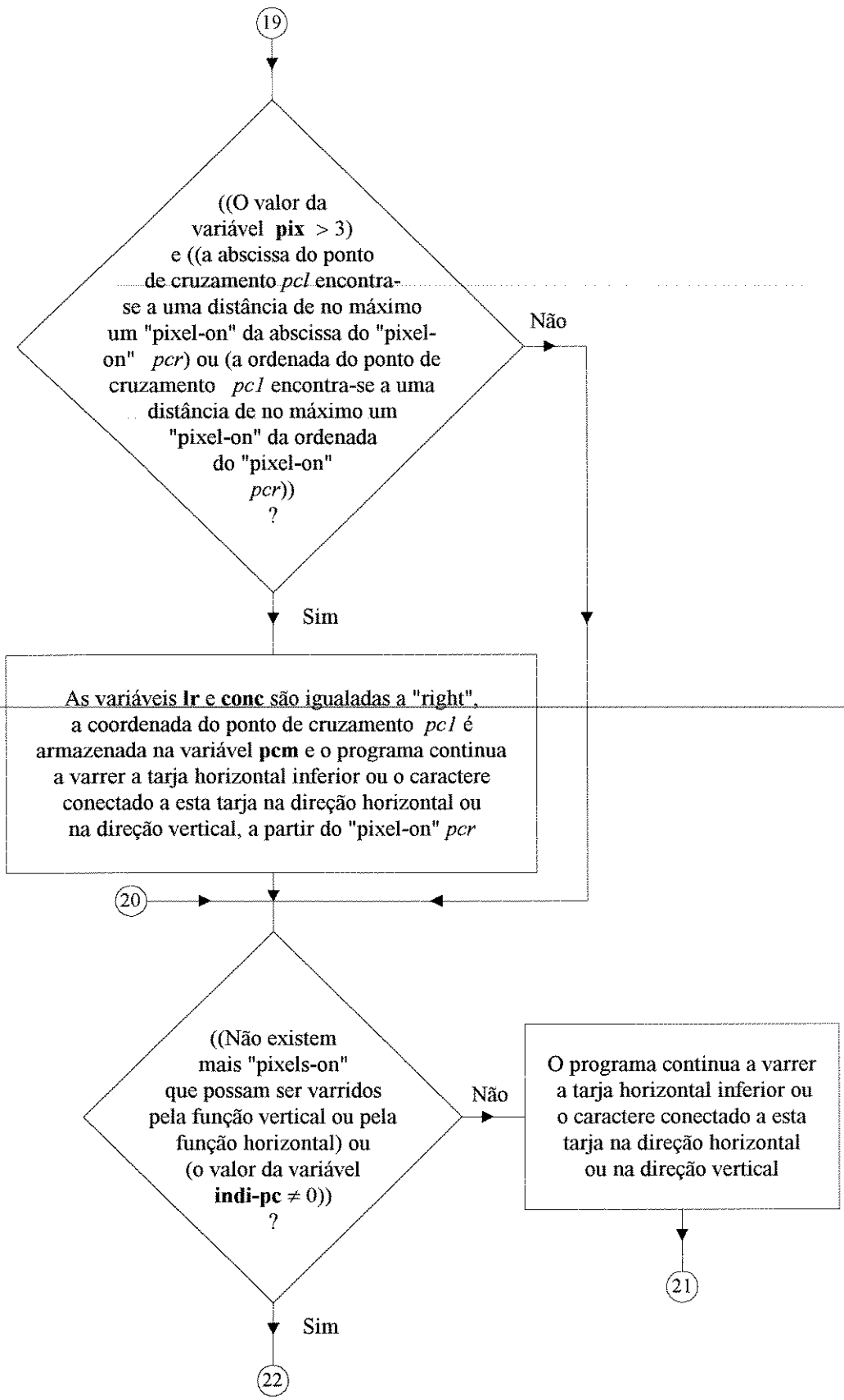


Figura 5.18(m) - Décima terceira parte do fluxograma da função verifica-tarja.

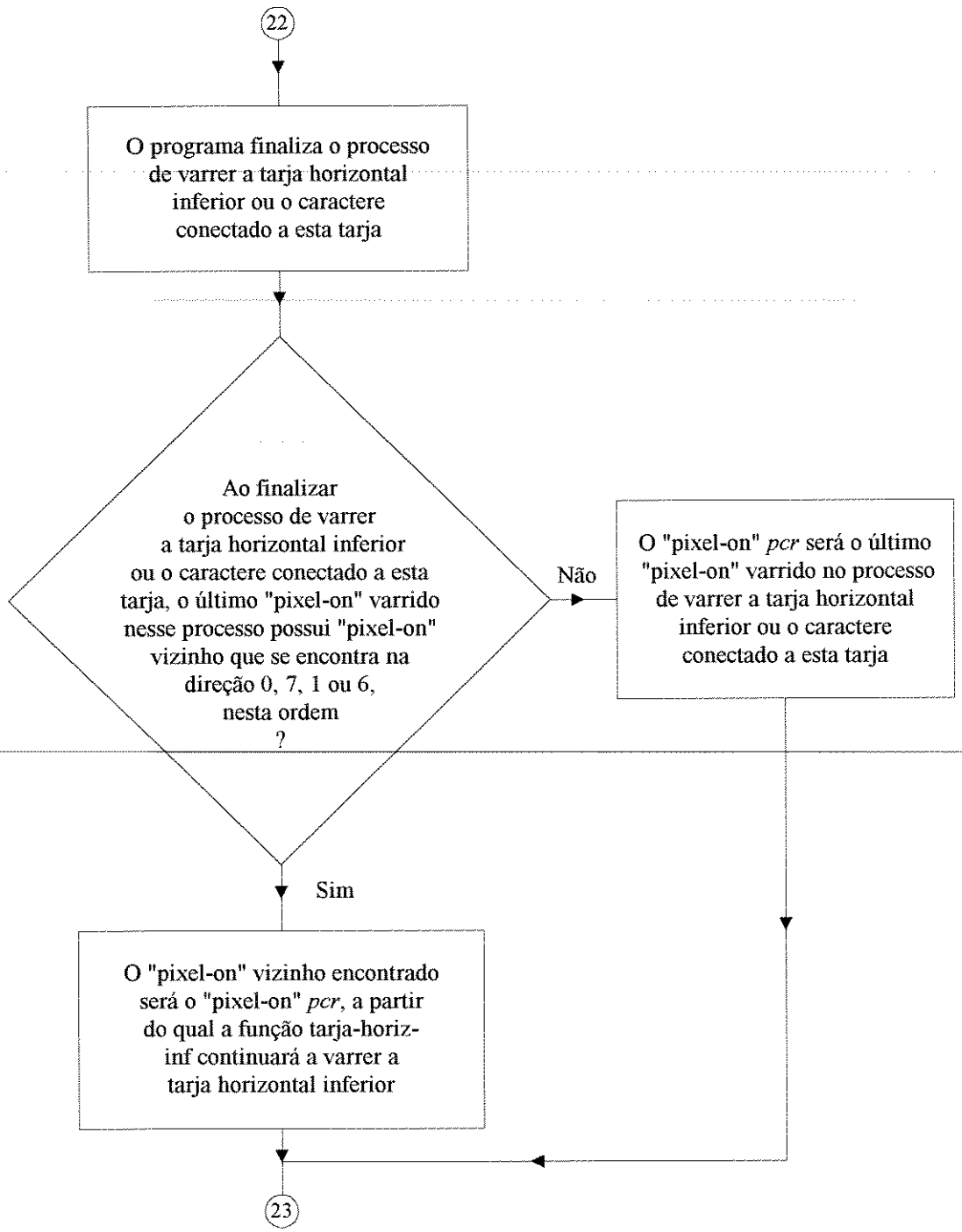


Figura 5.18(n) - Décima quarta parte do fluxograma da função verifica-tarja.

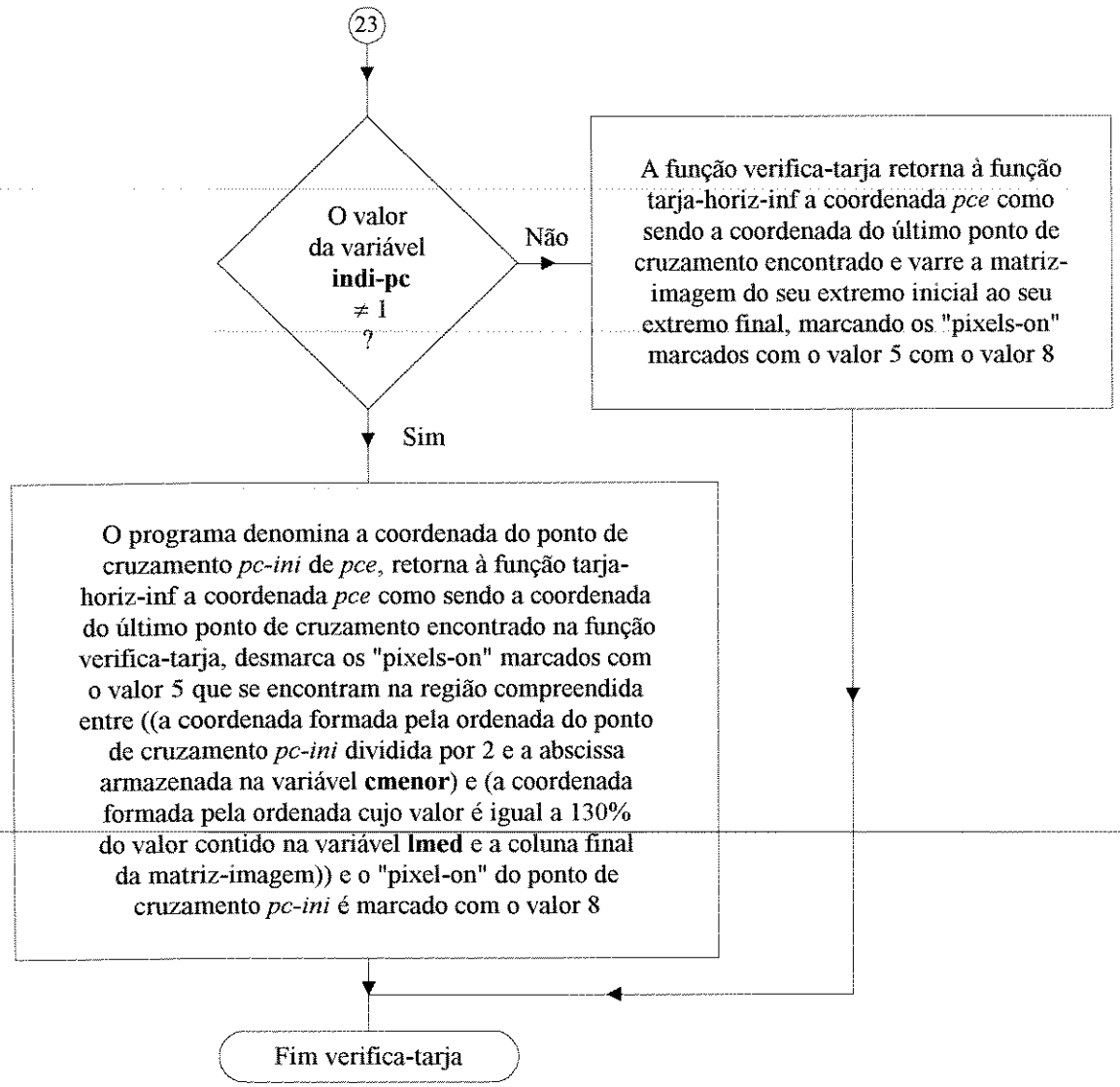


Figura 5.18(o) - Décima quinta parte do fluxograma da função verifica-tarja.

Fluxograma da função tarja-vertical

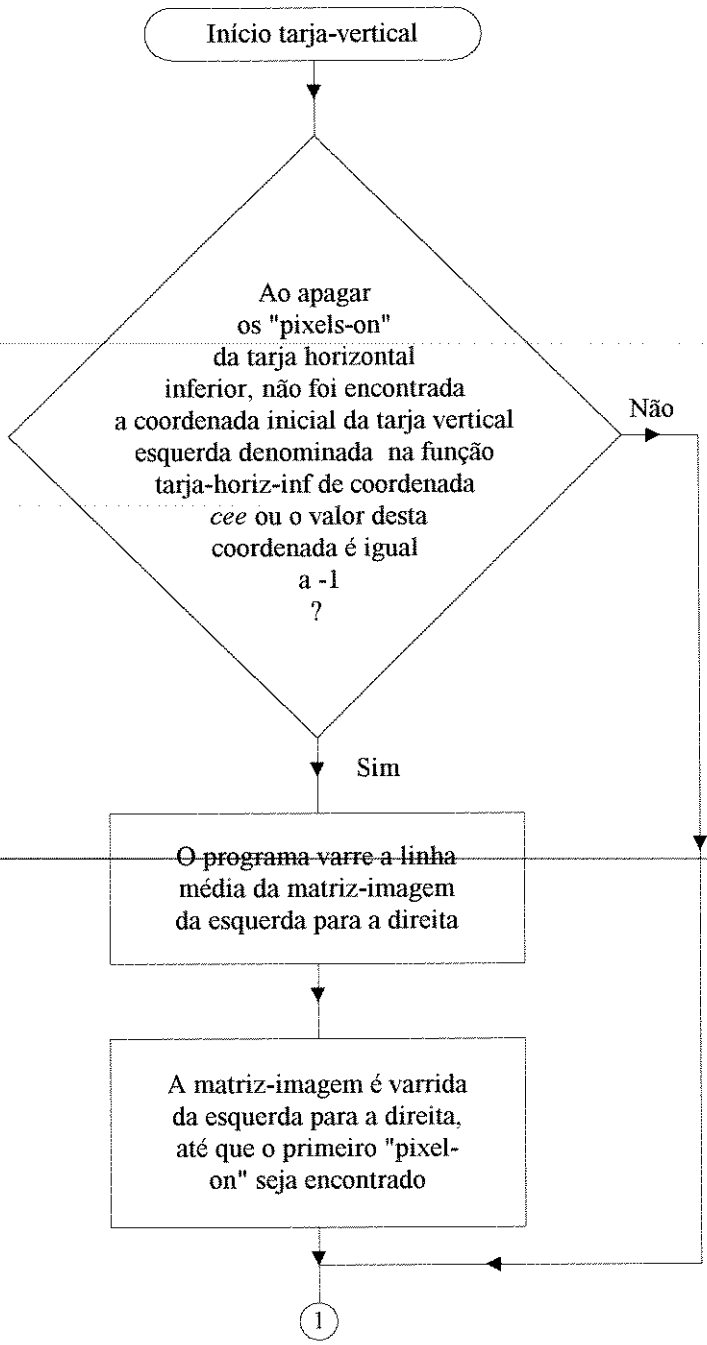


Figura 5.19(a) - Primeira parte do fluxograma da função tarja-vertical.

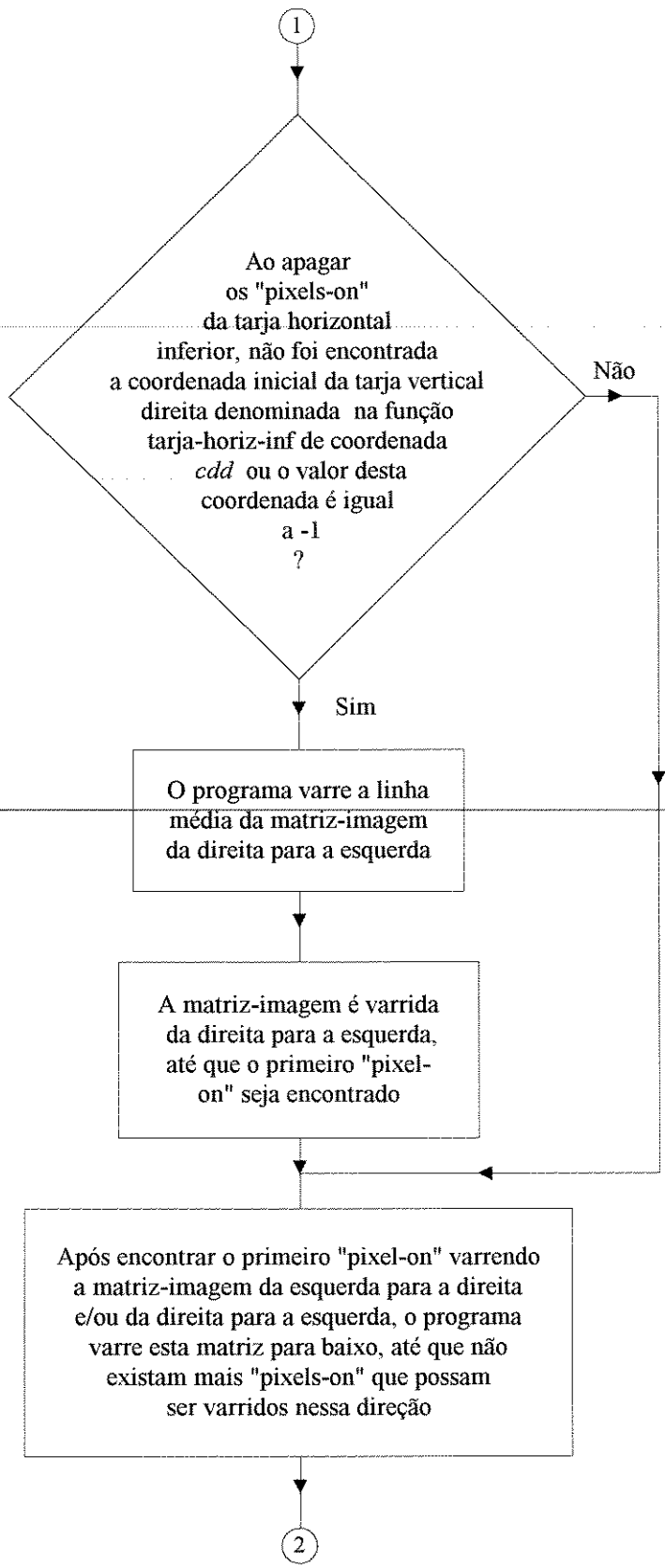


Figura 5.19(b) - Segunda parte do fluxograma da função tarja-vertical.

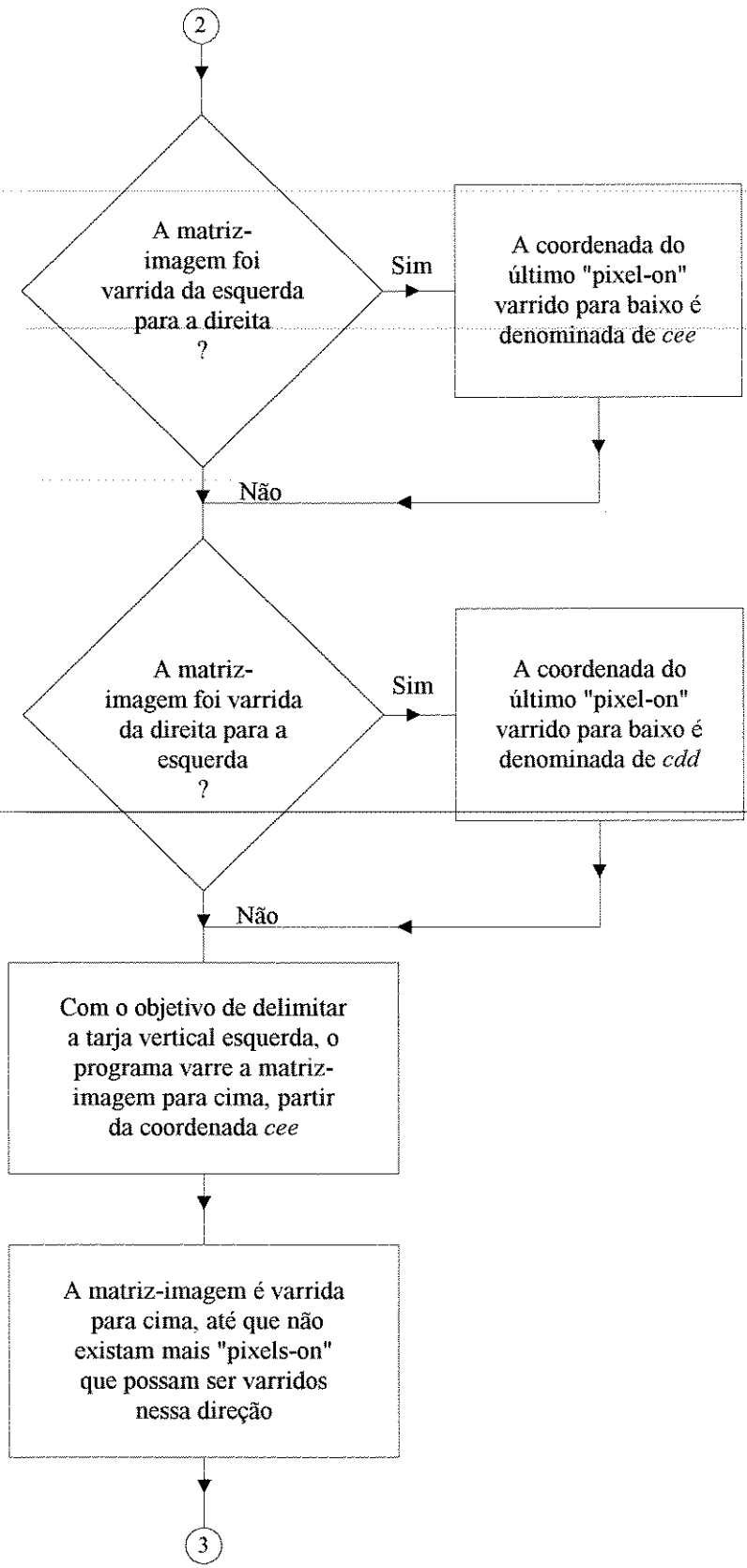


Figura 5.19(c) - Terceira parte do fluxograma da função tarja-vertical.

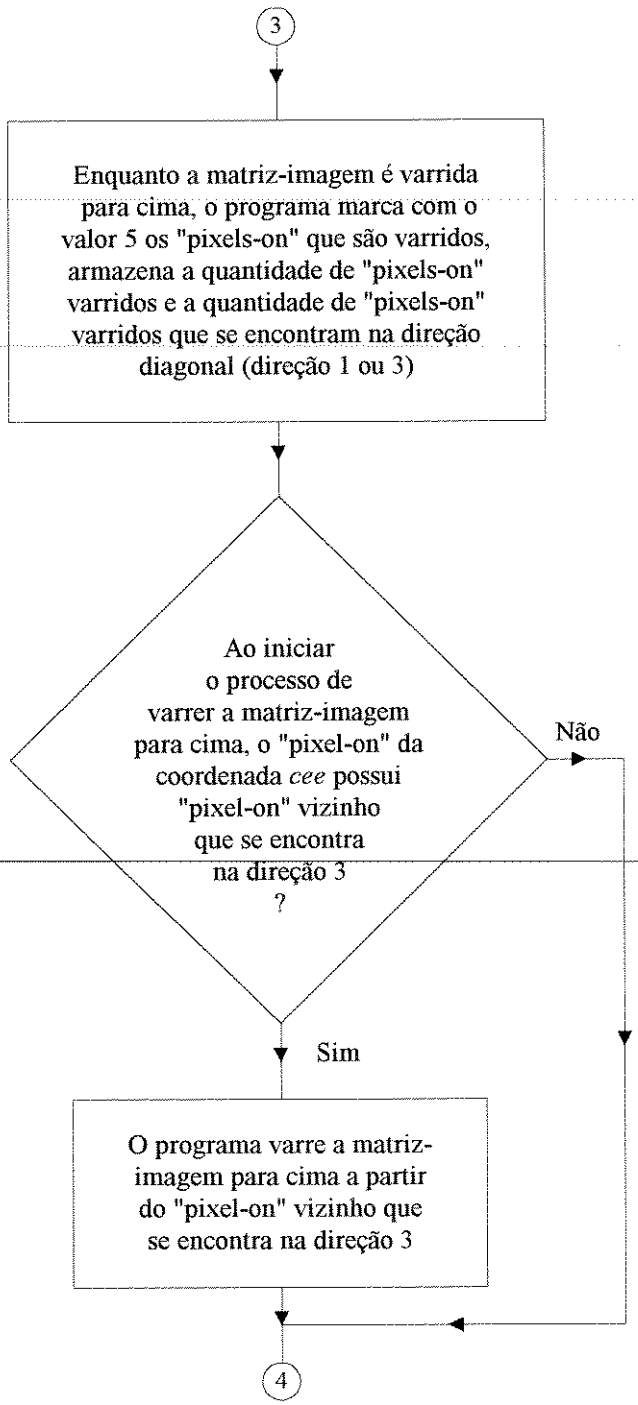


Figura 5.19(d) - Quarta parte do fluxograma da função tarja-vertical.

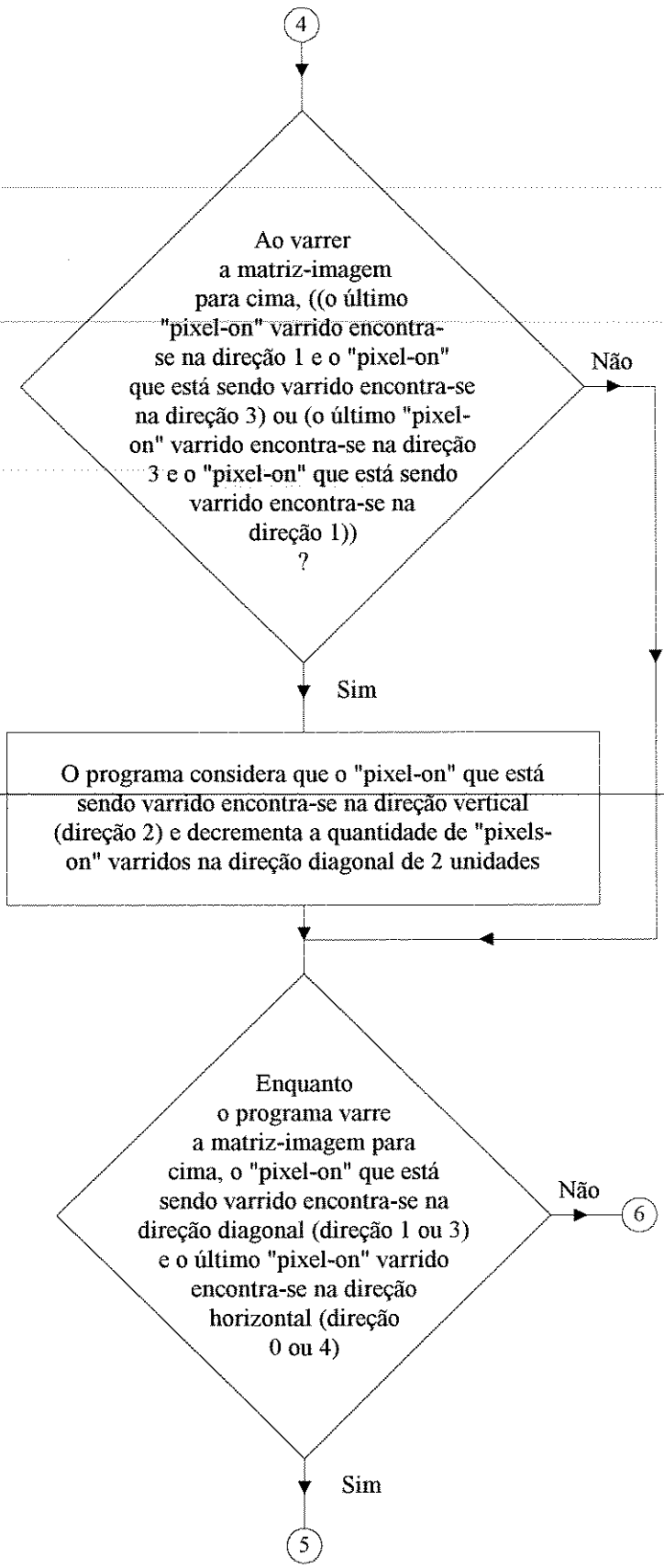


Figura 5.19(e) - Quinta parte do fluxograma da função tarja-vertical.

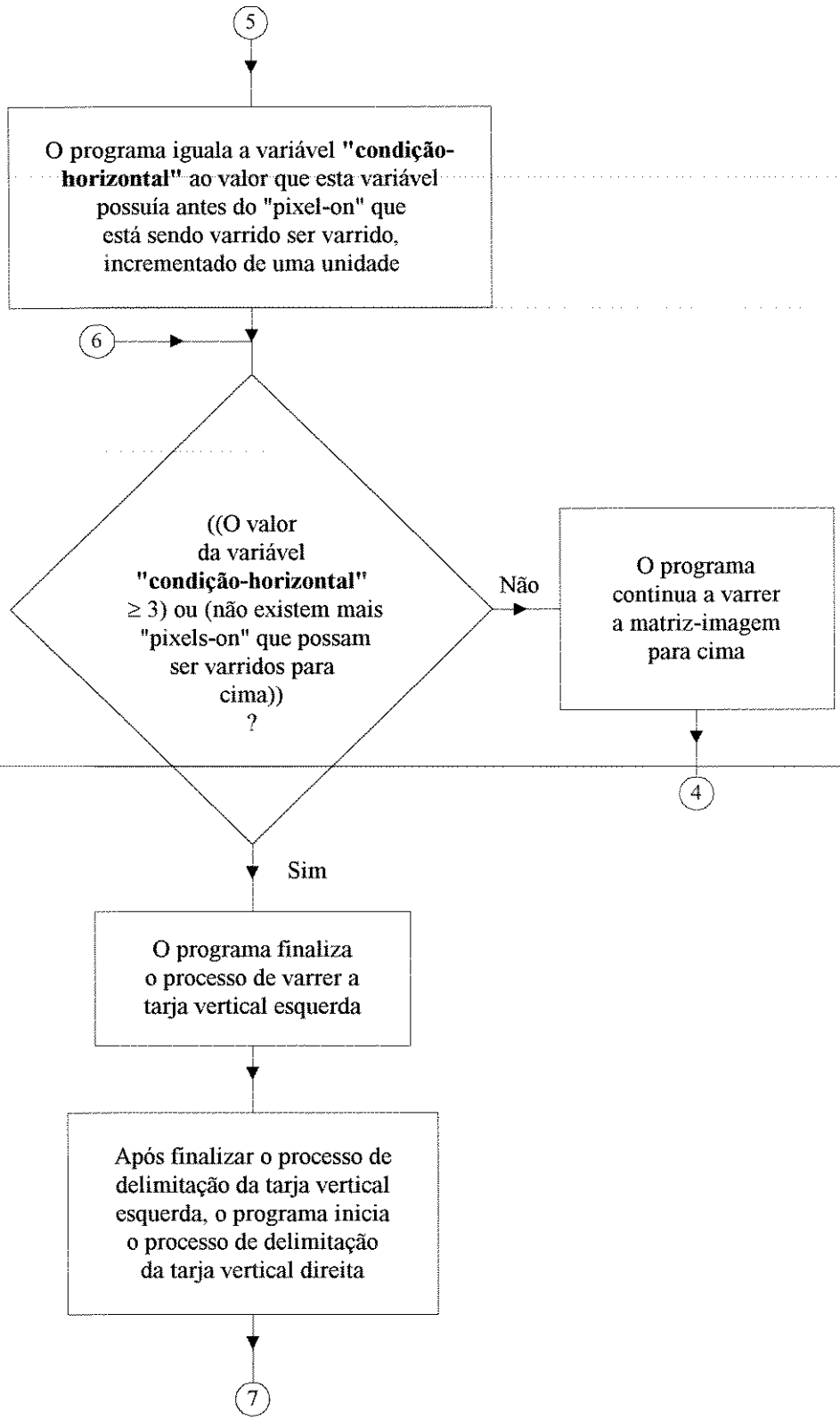


Figura 5.19(f) - Sexta parte do fluxograma da função tarja-vertical.

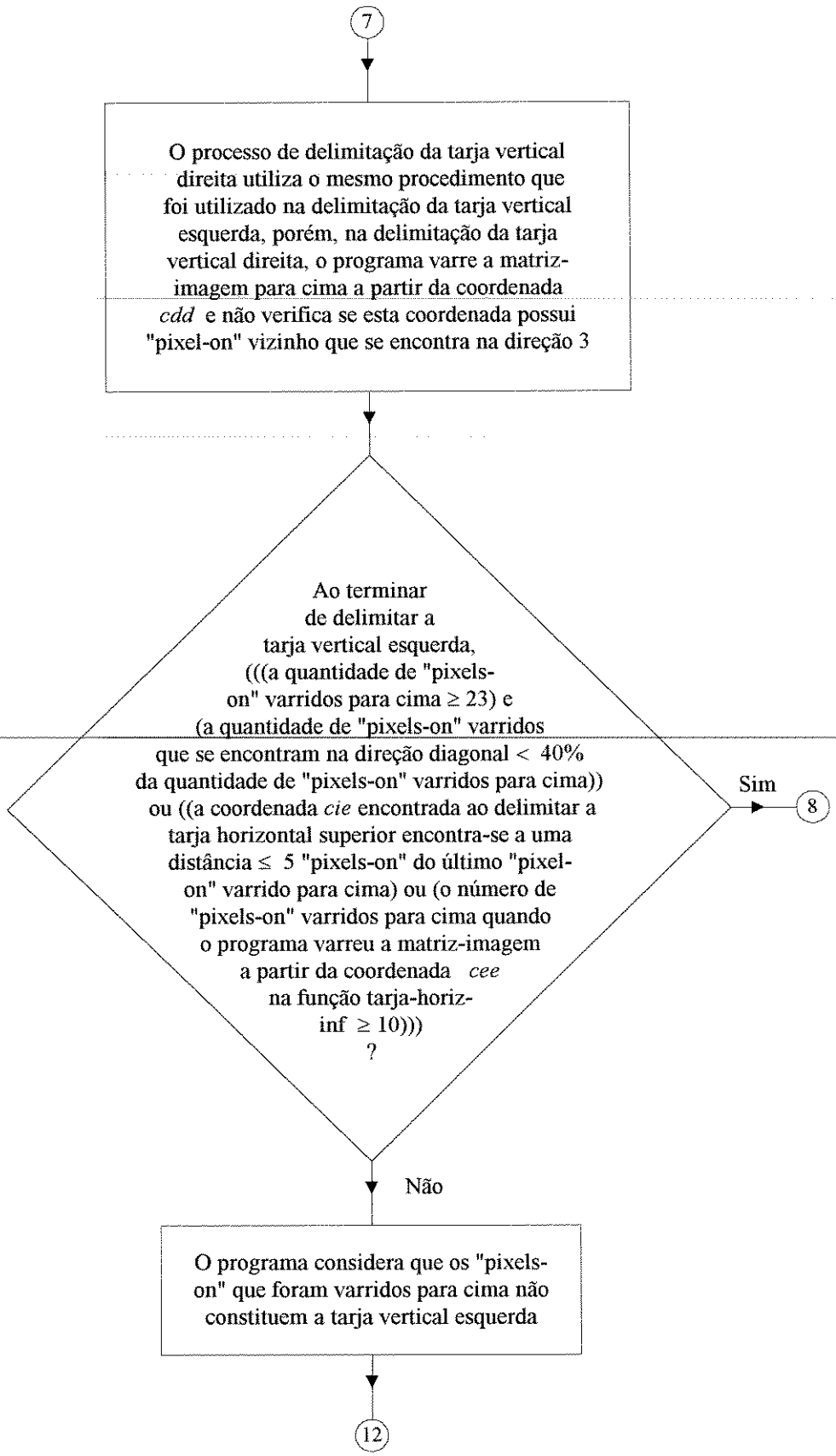


Figura 5.19(g) -Sétima parte do fluxograma da função tarja-vertical.

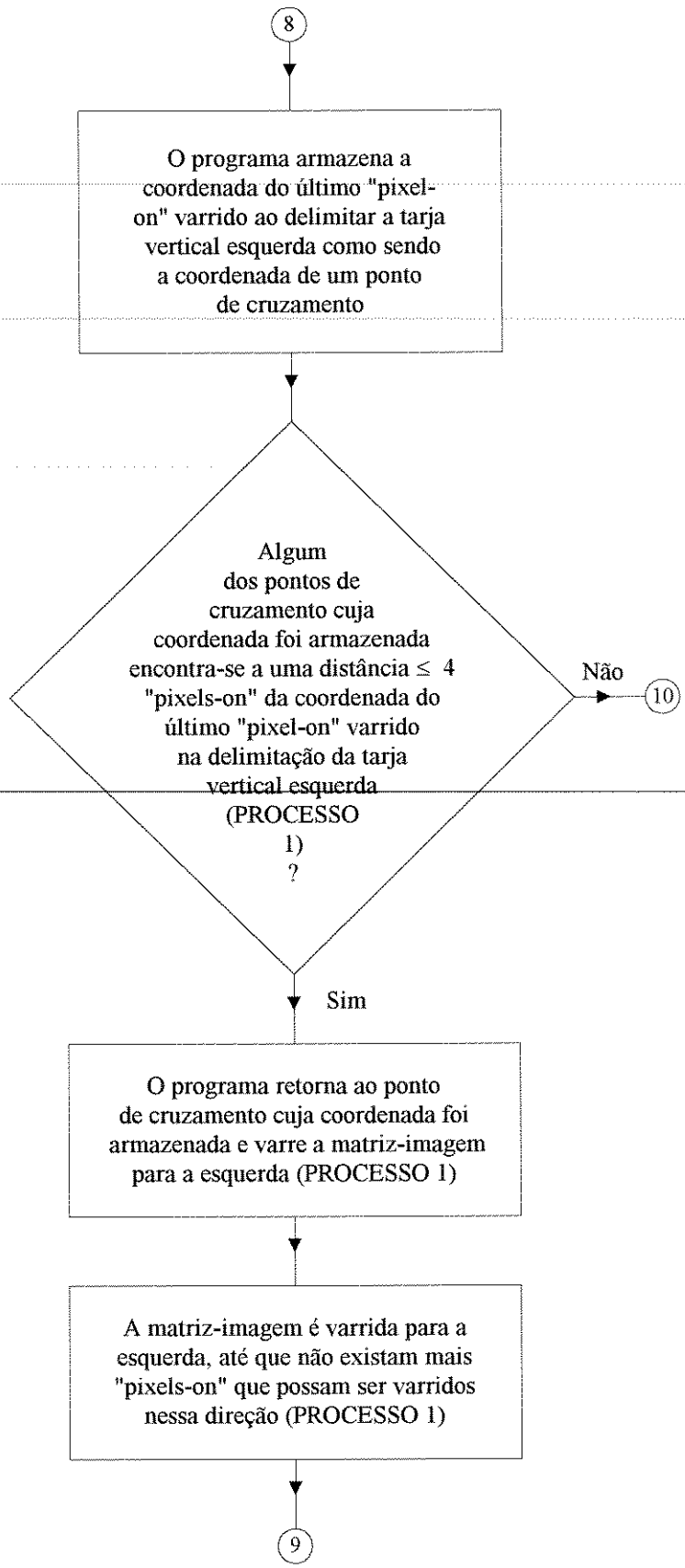


Figura 5.19(h) - Oitava parte do fluxograma da função tarja-vertical.

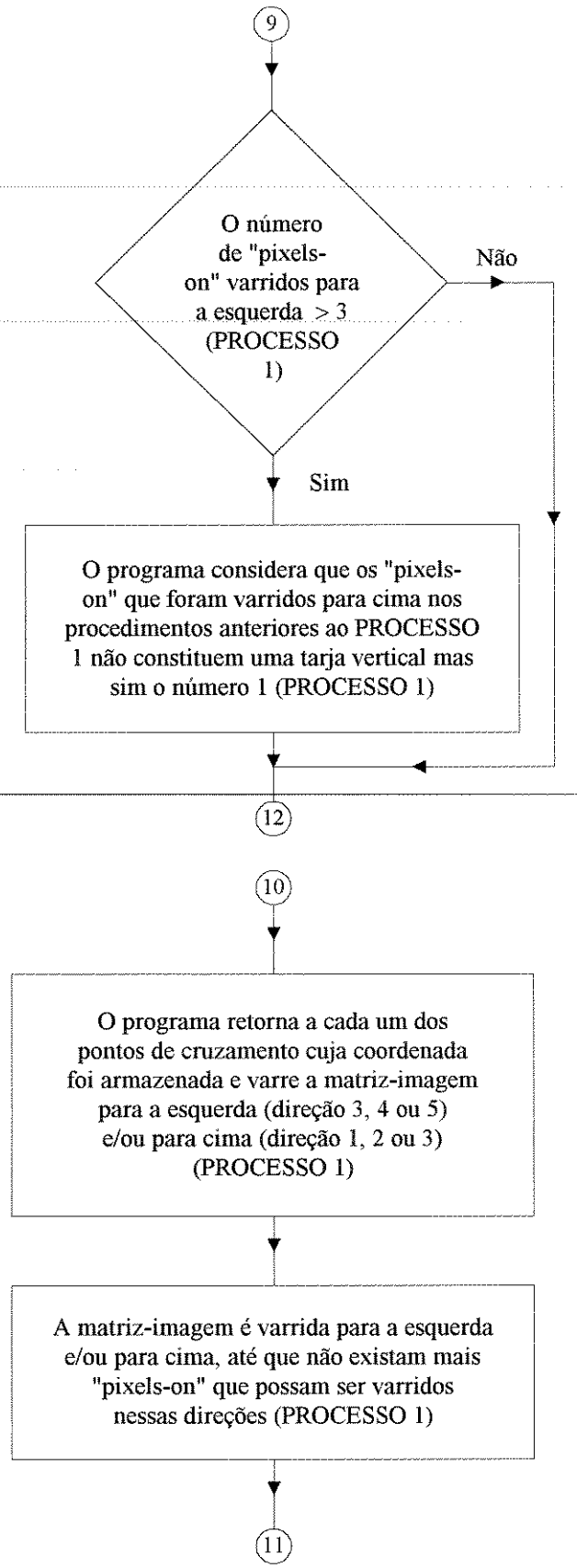


Figura 5.19(i) - Nona parte do fluxograma da função tarja-vertical.

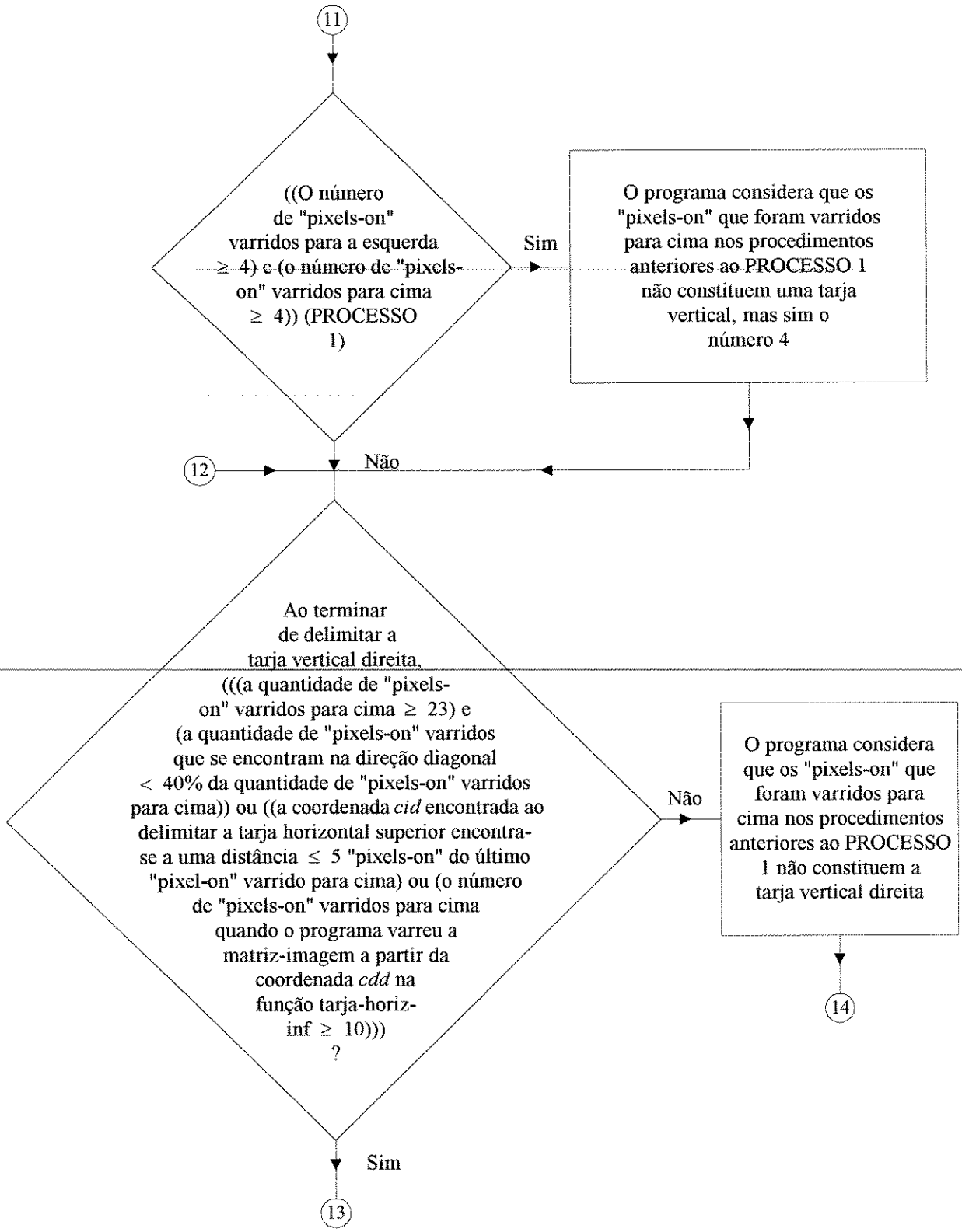


Figura 5.19(j) - Décima parte do fluxograma da função tarja-vertical.

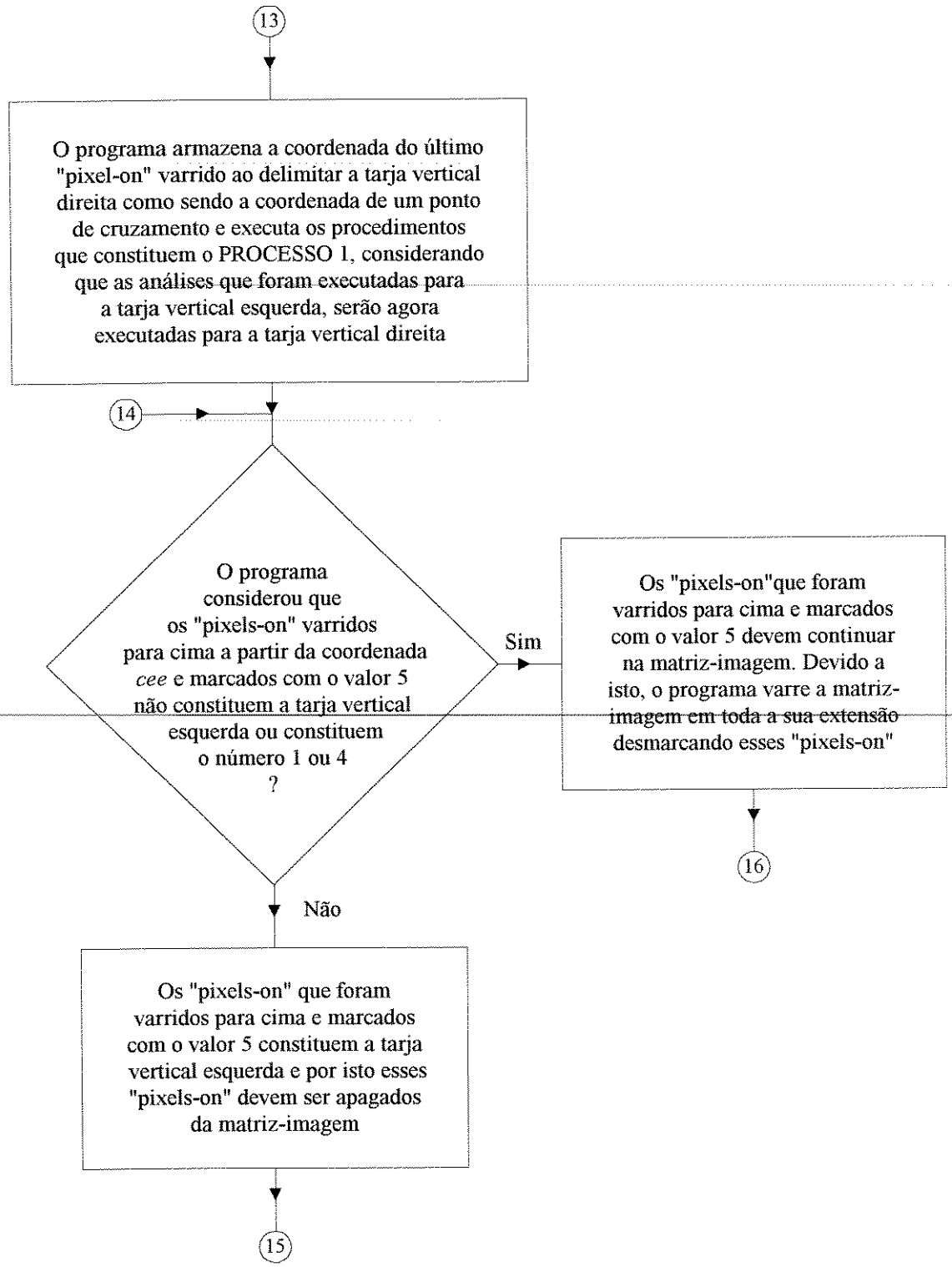


Figura 5.19(k) - Décima primeira parte do fluxograma da função tarja-vertical.

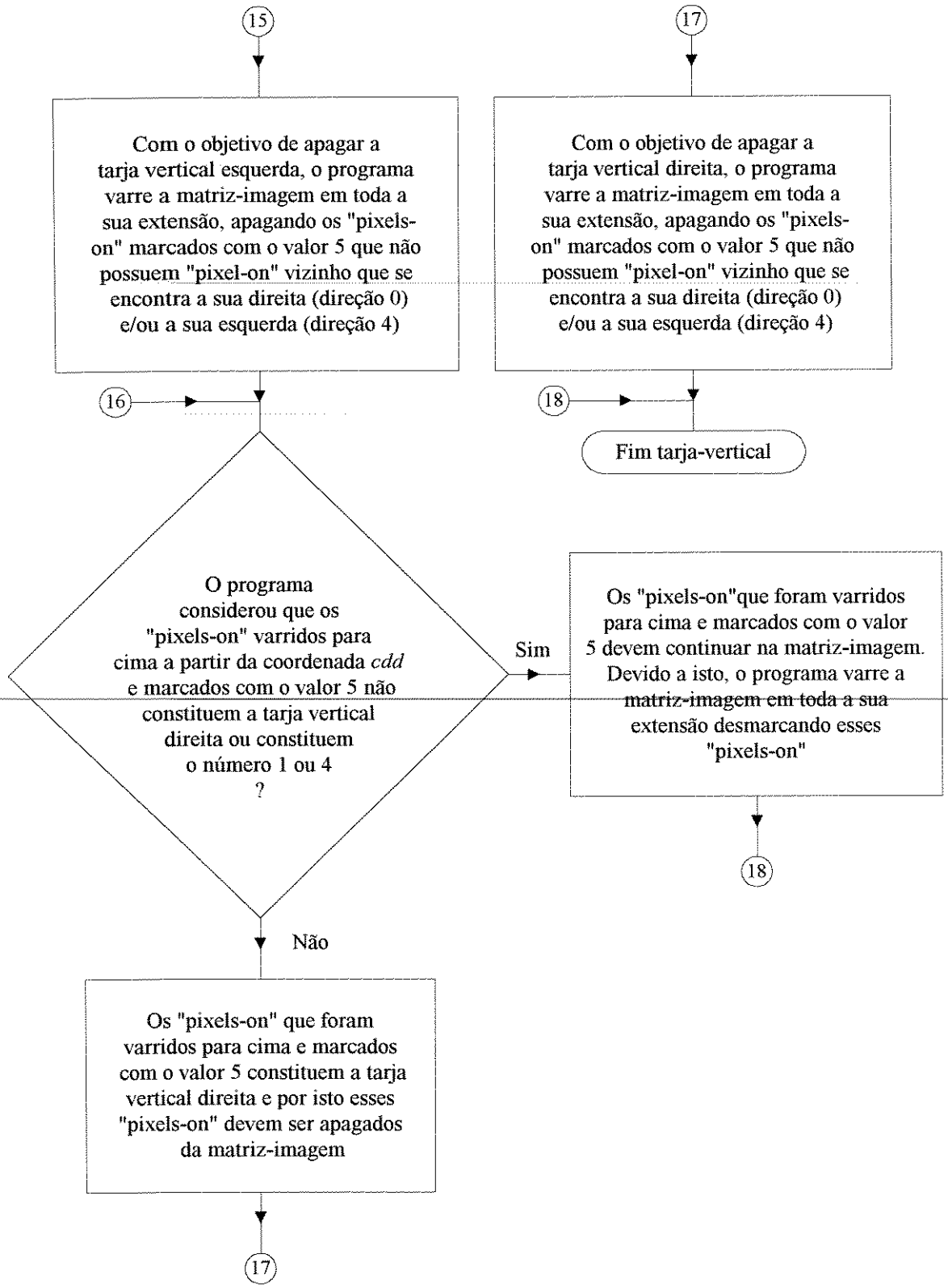


Figura 5.19(l) - Décima segunda parte do fluxograma da função tarja-vertical.

Fluxograma da função verifica-cerquilha

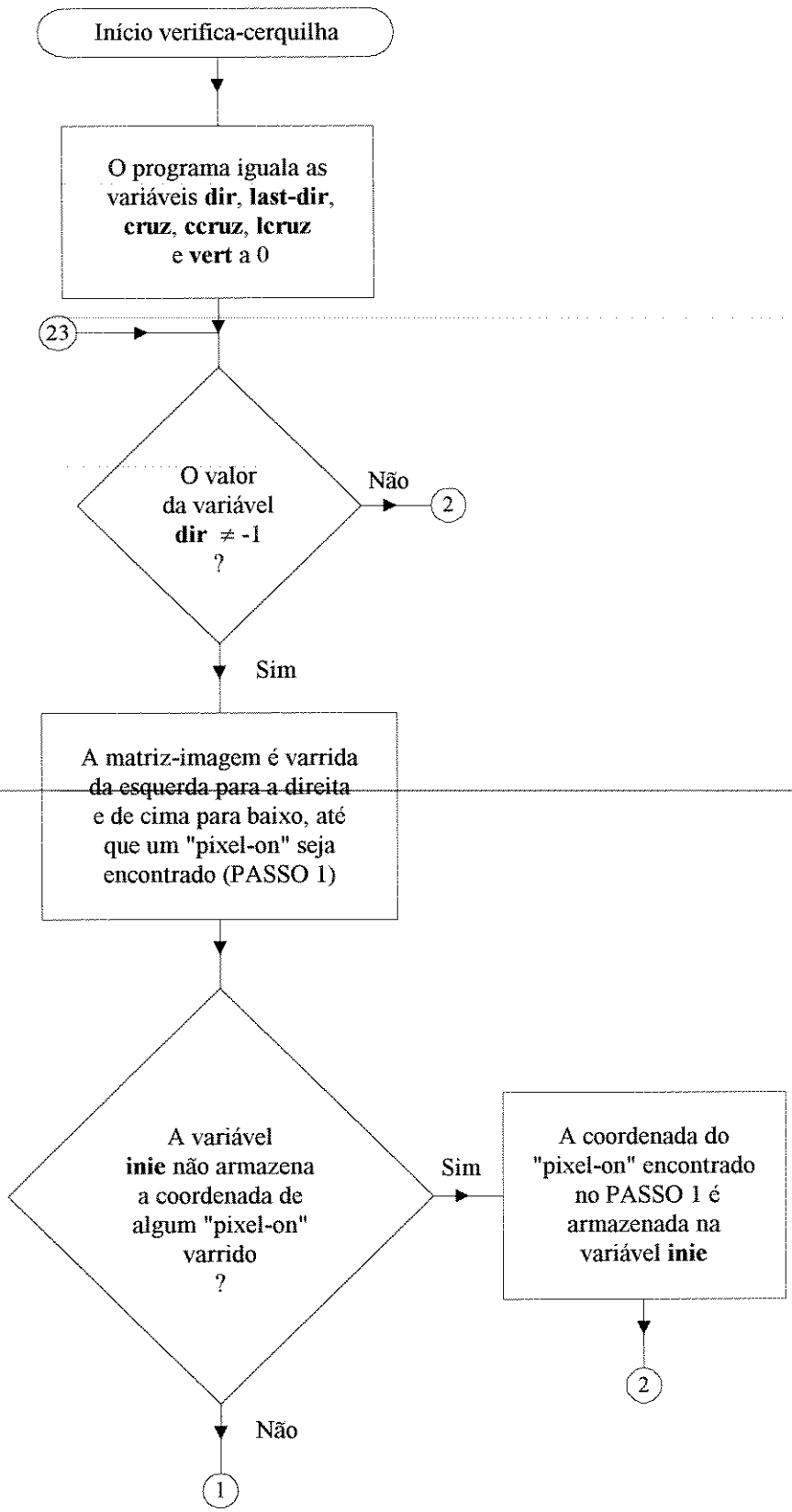


Figura 5.21(a) - Primeira parte do fluxograma da função verifica-cerquilha.

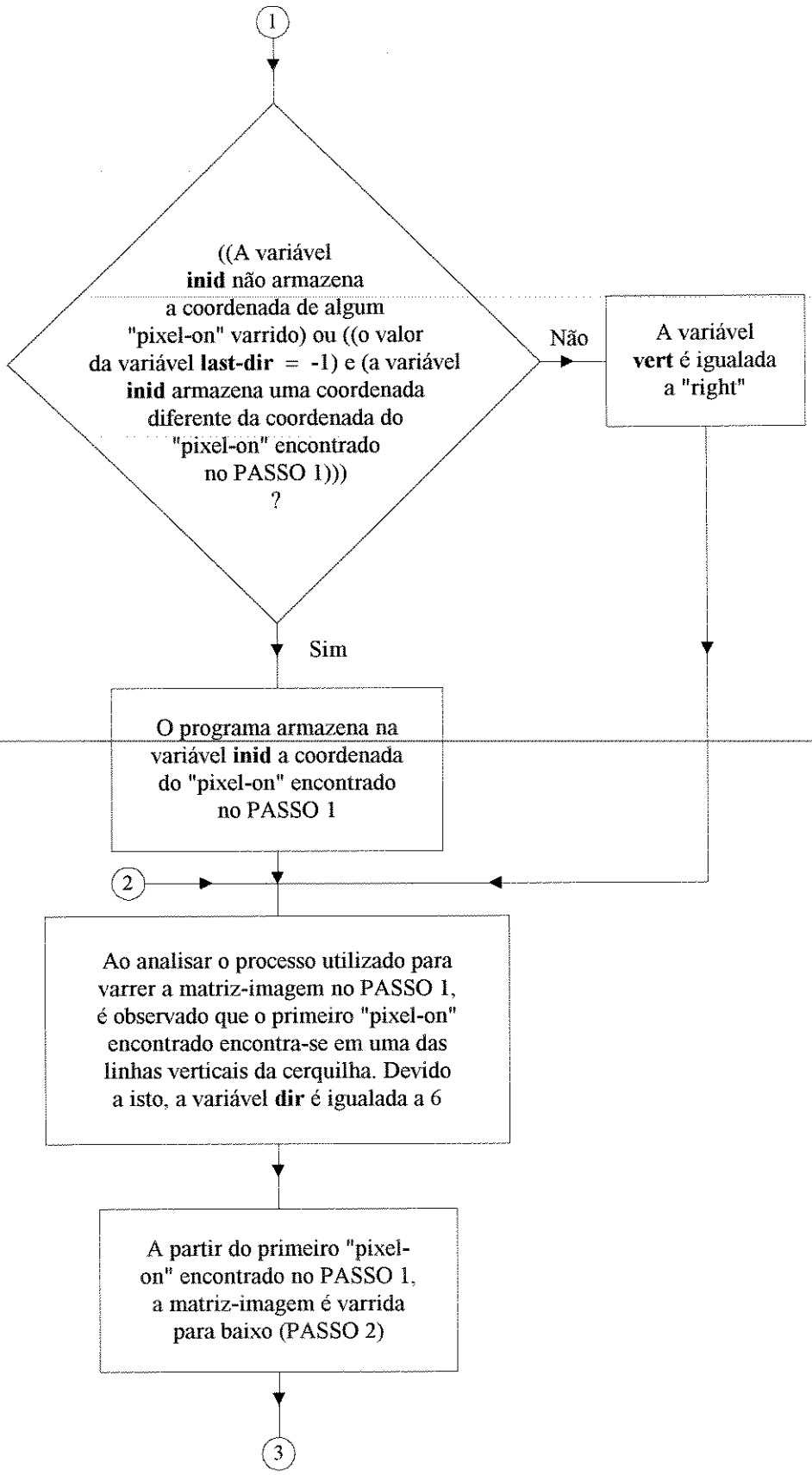


Figura 5.21(b) - Segunda parte do fluxograma da função verifica-cerquilha.

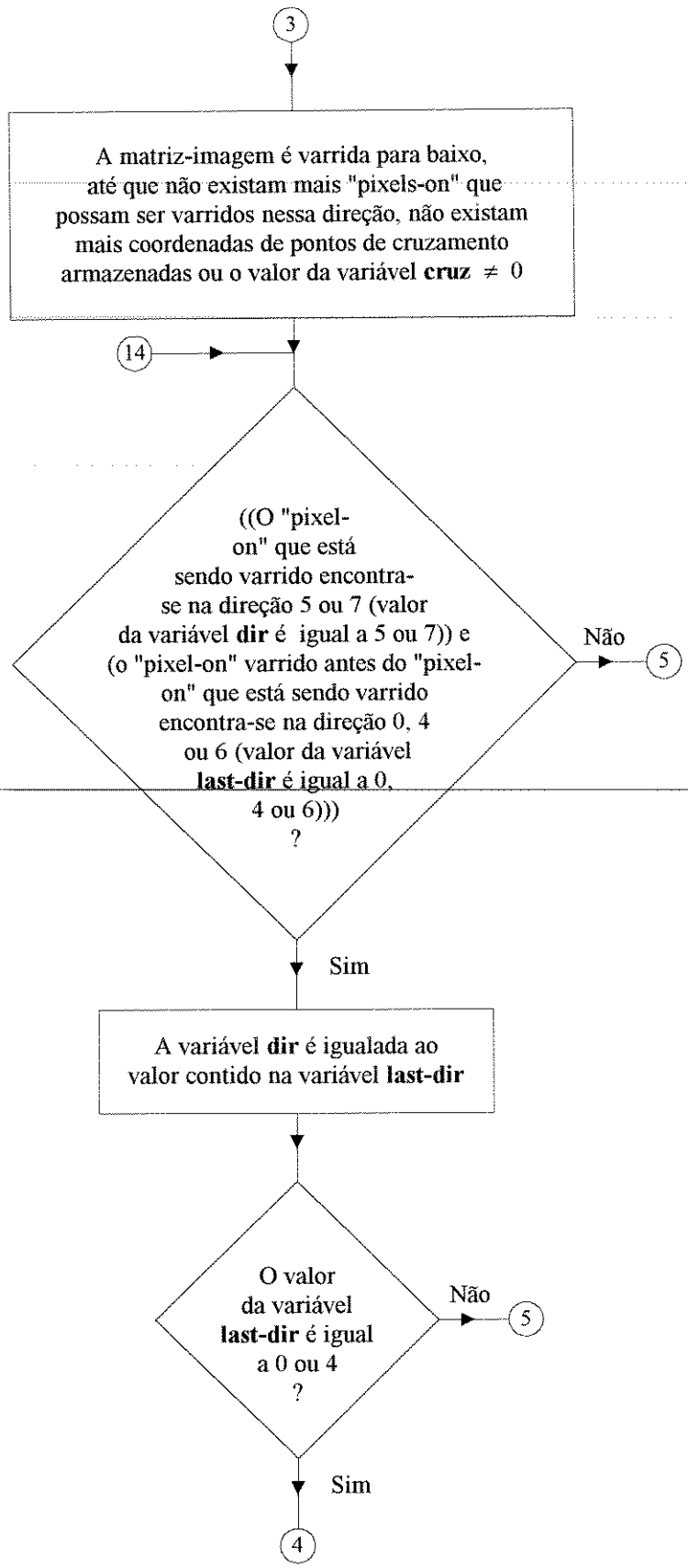


Figura 5.21(c) - Terceira parte do fluxograma da função verifica-cerquilha.

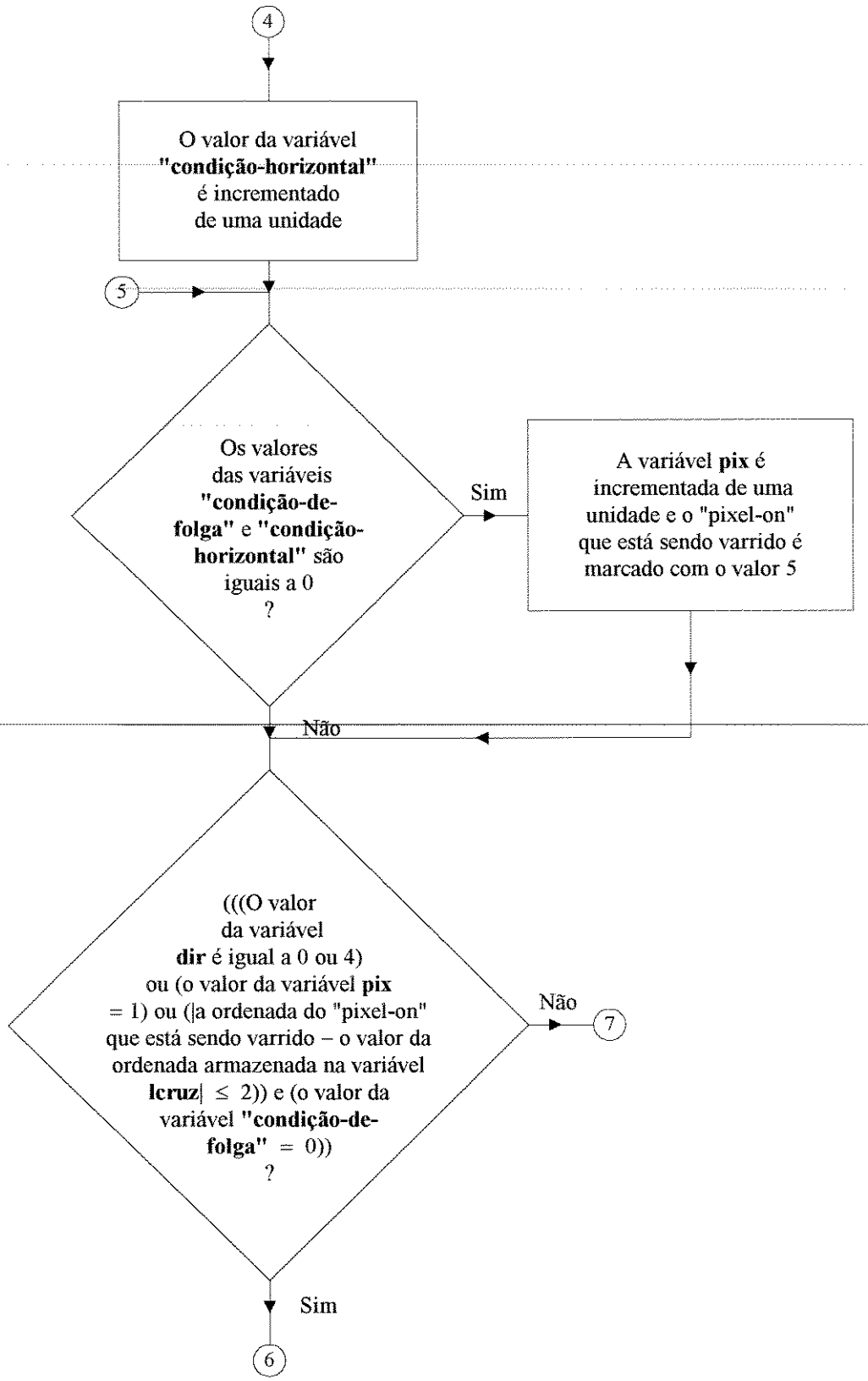


Figura 5.21(d) - Quarta parte do fluxograma da função verifica-cerquilha.

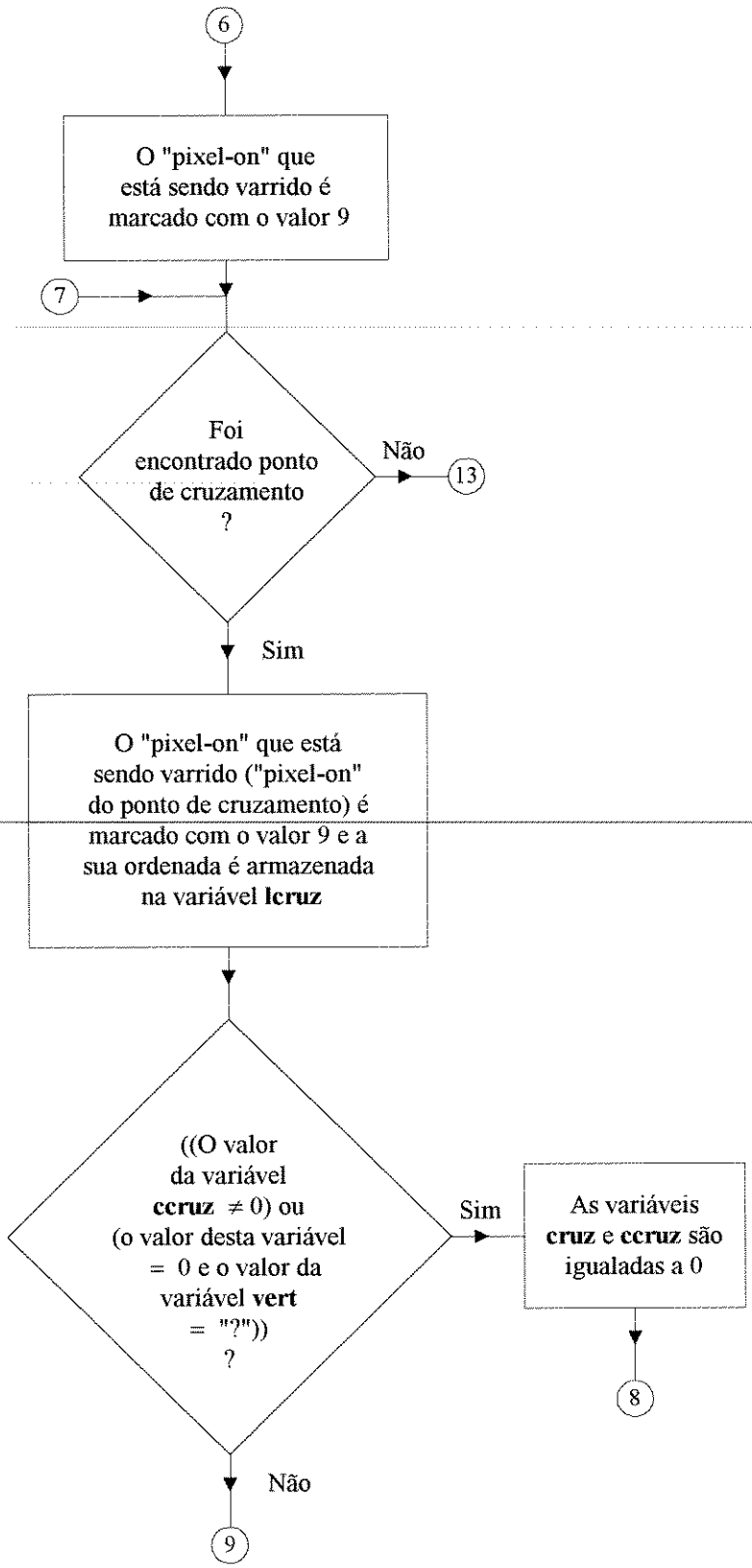


Figura 5.21(e) - Quinta parte do fluxograma da função verifica-cerquilha.

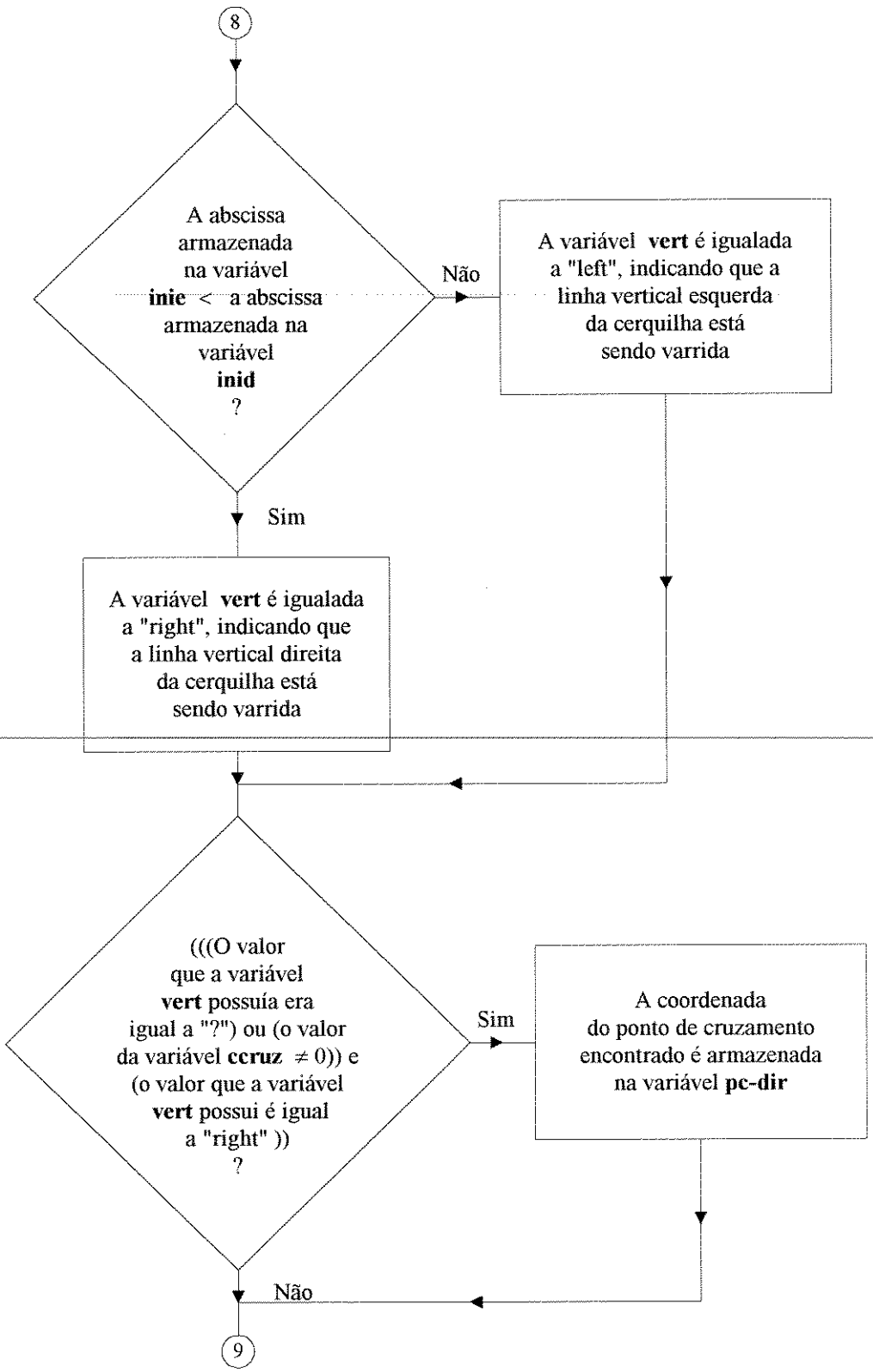


Figura 5.21(f) - Sexta parte do fluxograma da função verifica-cerquilha.

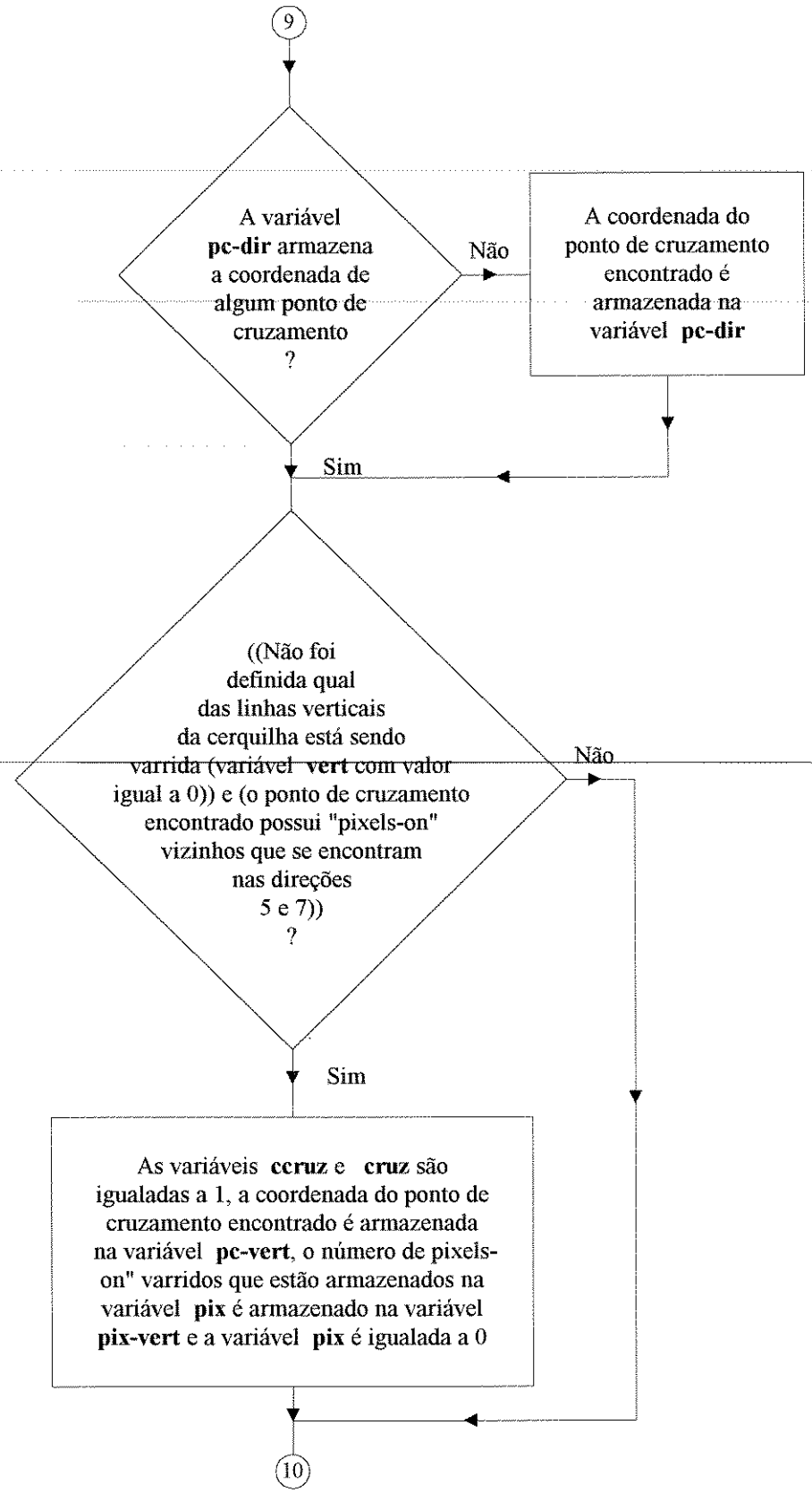


Figura 5.21(g) - Sétima parte do fluxograma da função verifica-cerquilha.

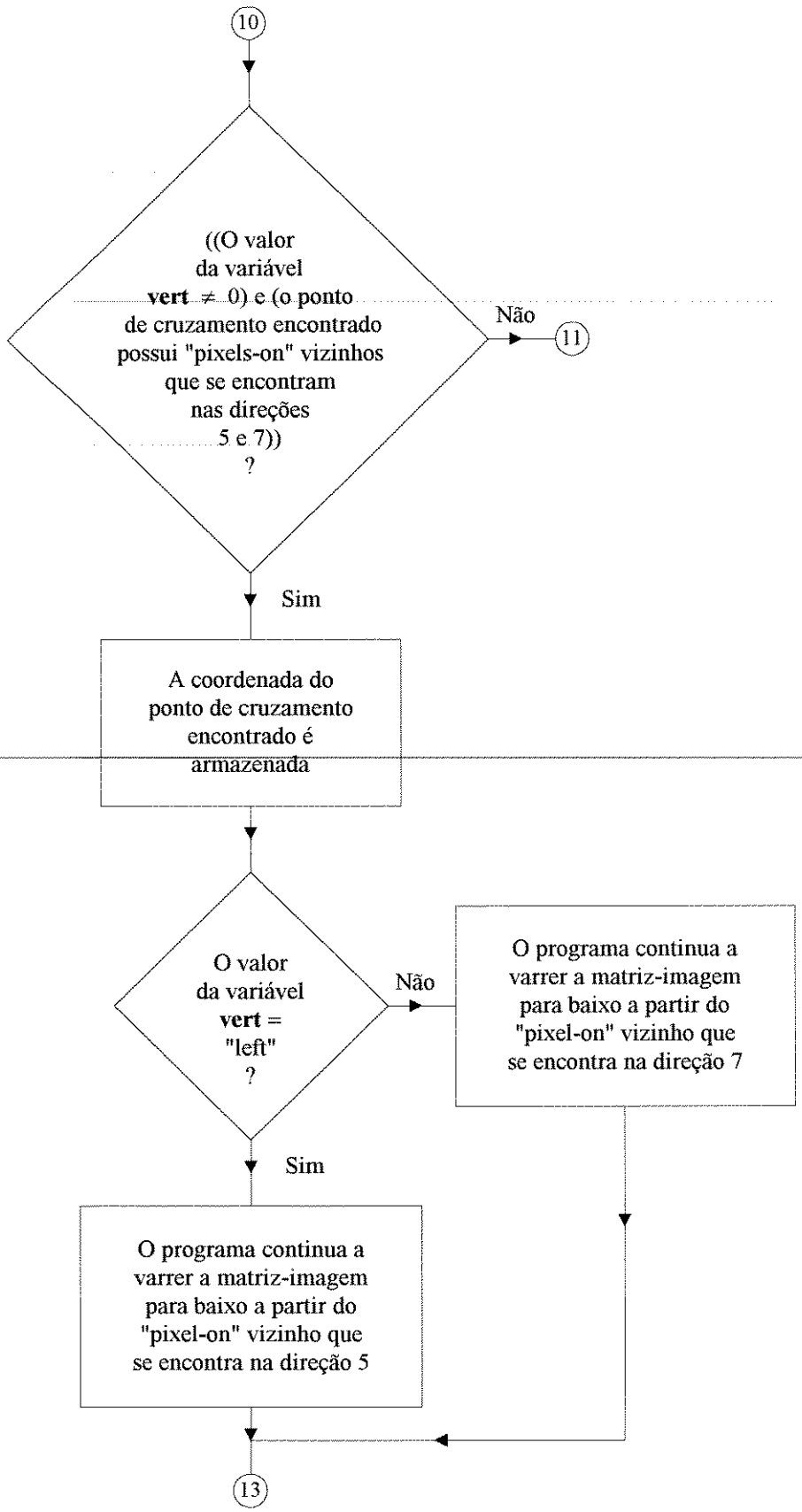


Figura 5.21(h) - Oitava parte do fluxograma da função verifica-cerquilha.

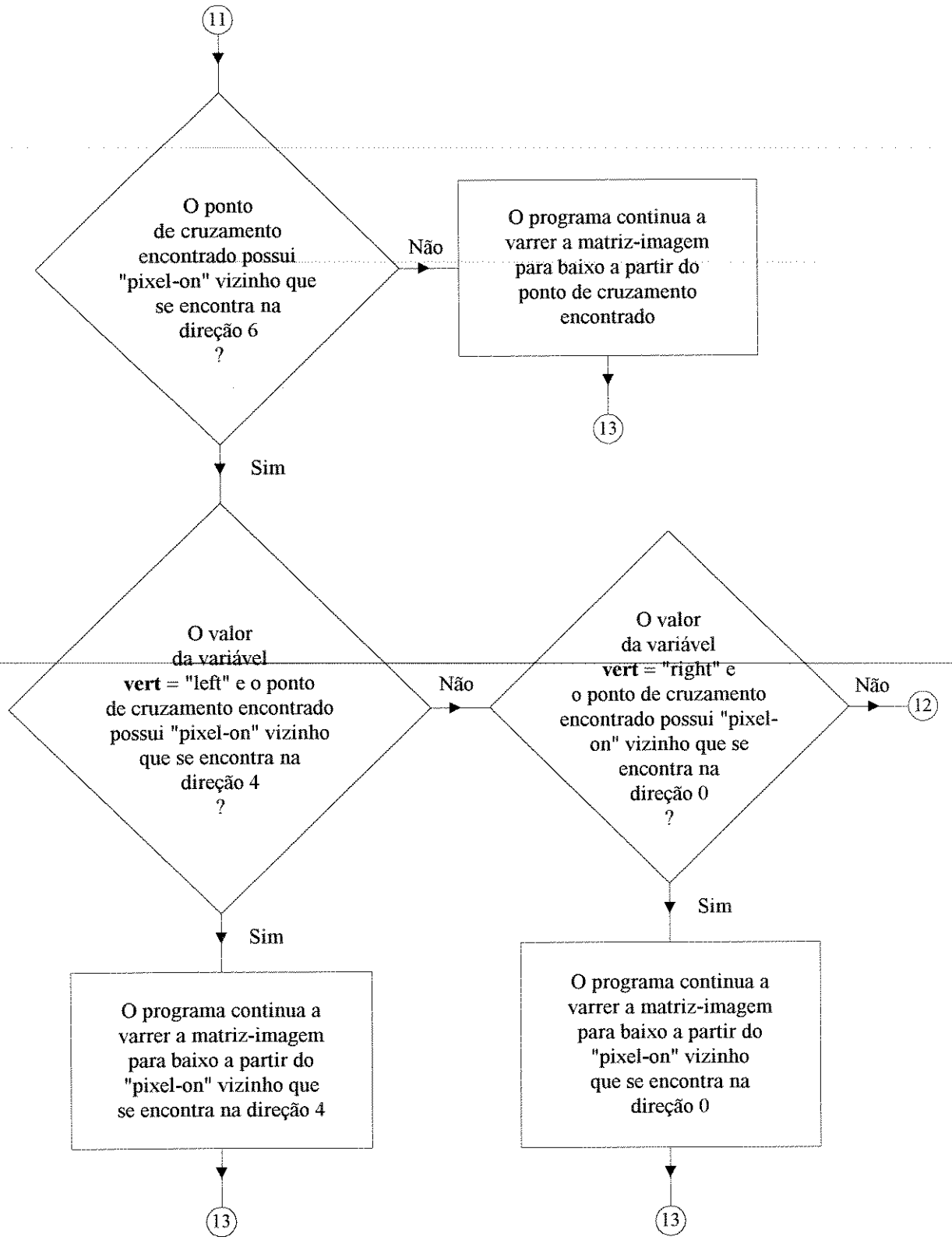


Figura 5.21(i) - Nona parte do fluxograma da função verifica-cerquilha.

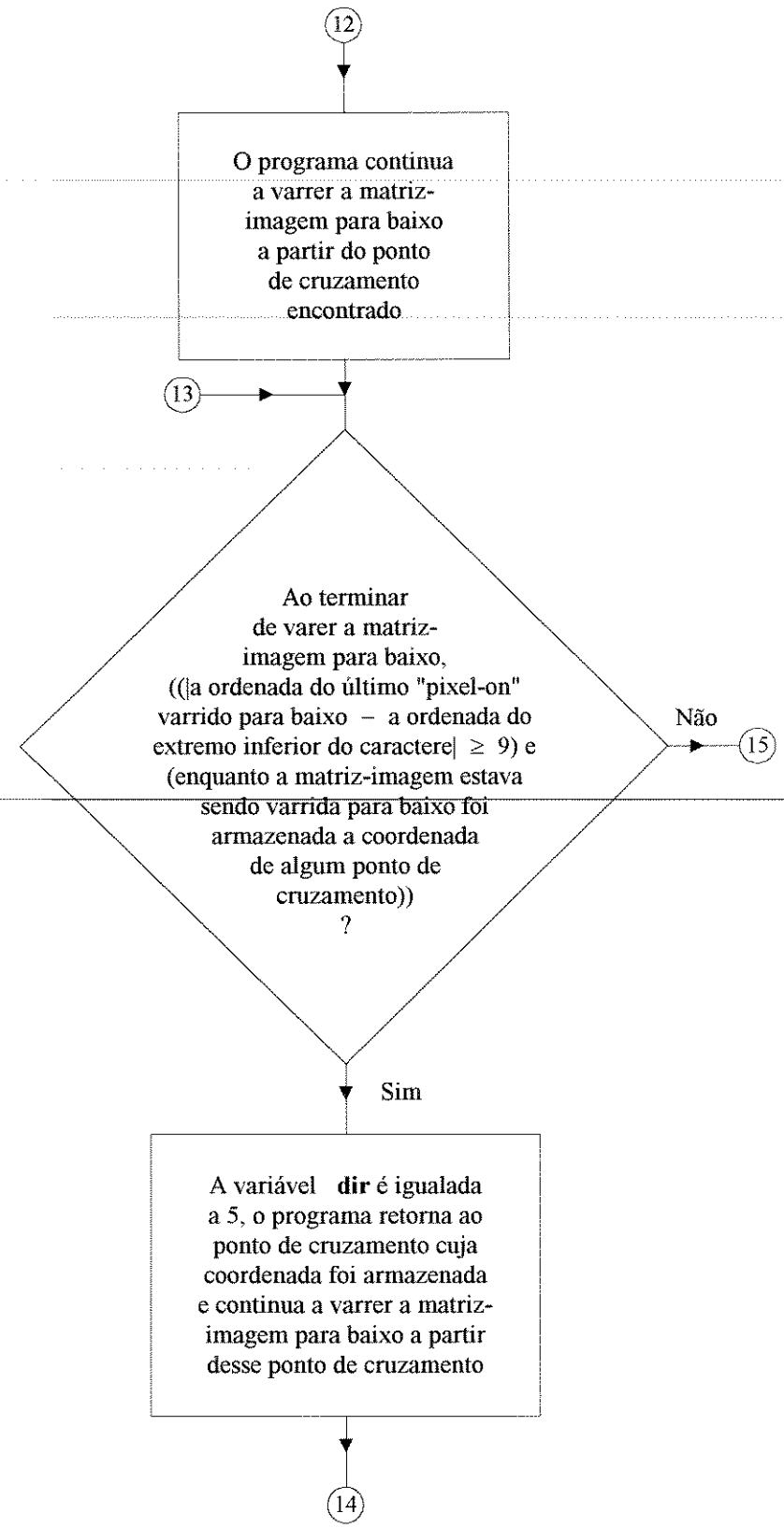


Figura 5.21(j) - Décima parte do fluxograma da função verifica-cerquilha.

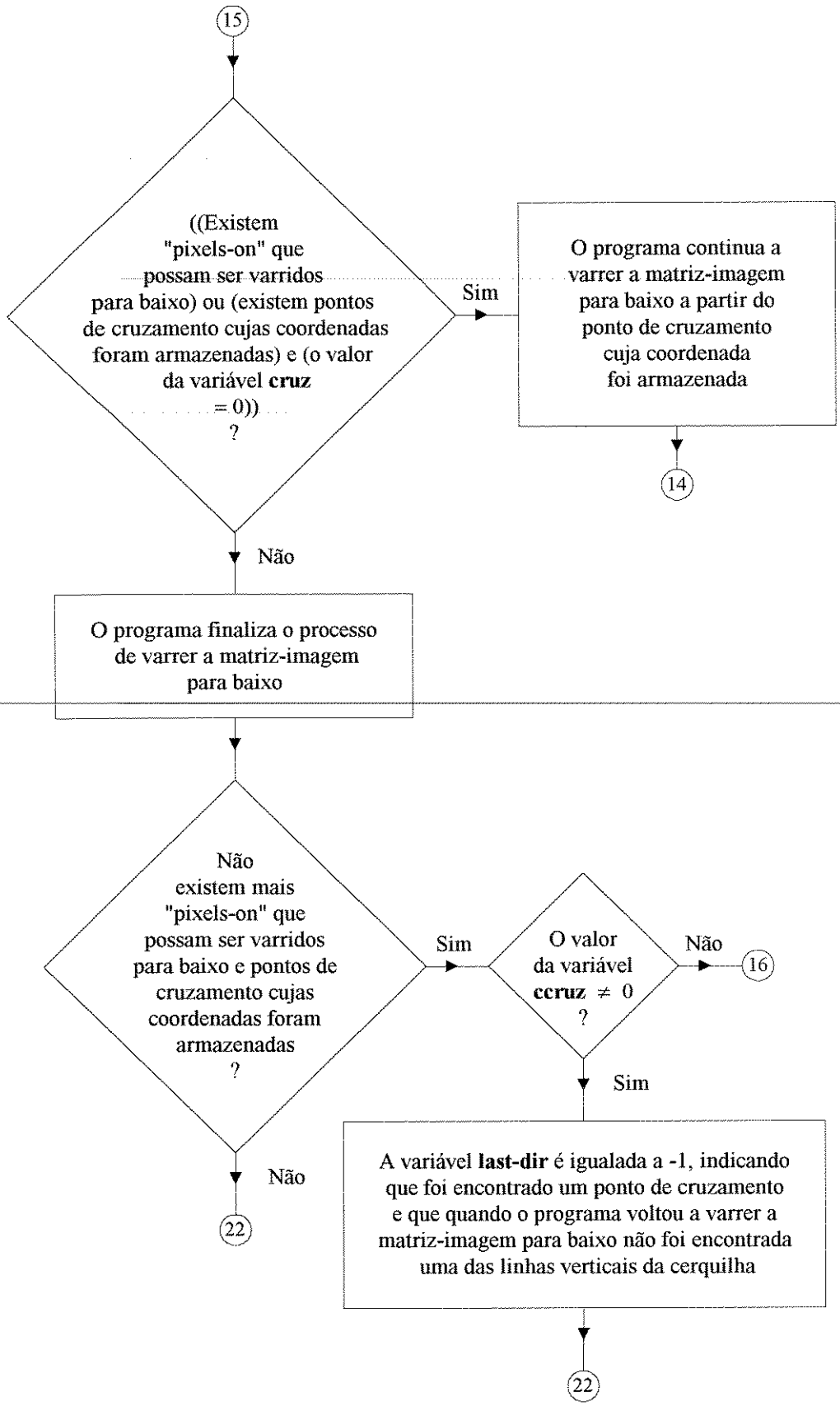


Figura 5.21(k) - Décima primeira parte do fluxograma da função verifica-cerquilha.

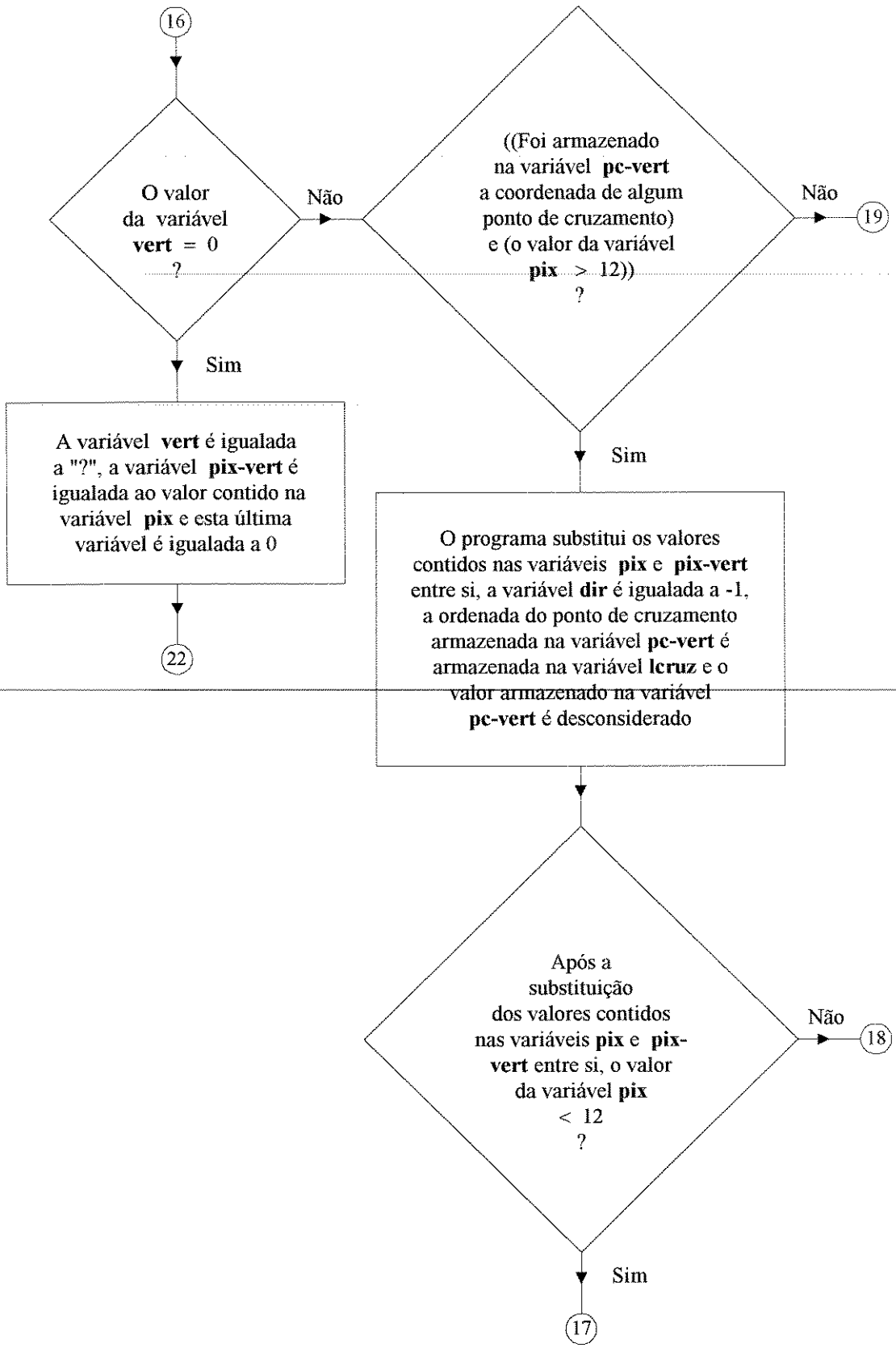


Figura 5.21(l) - Décima segunda parte do fluxograma da função verifica-cerquilha.

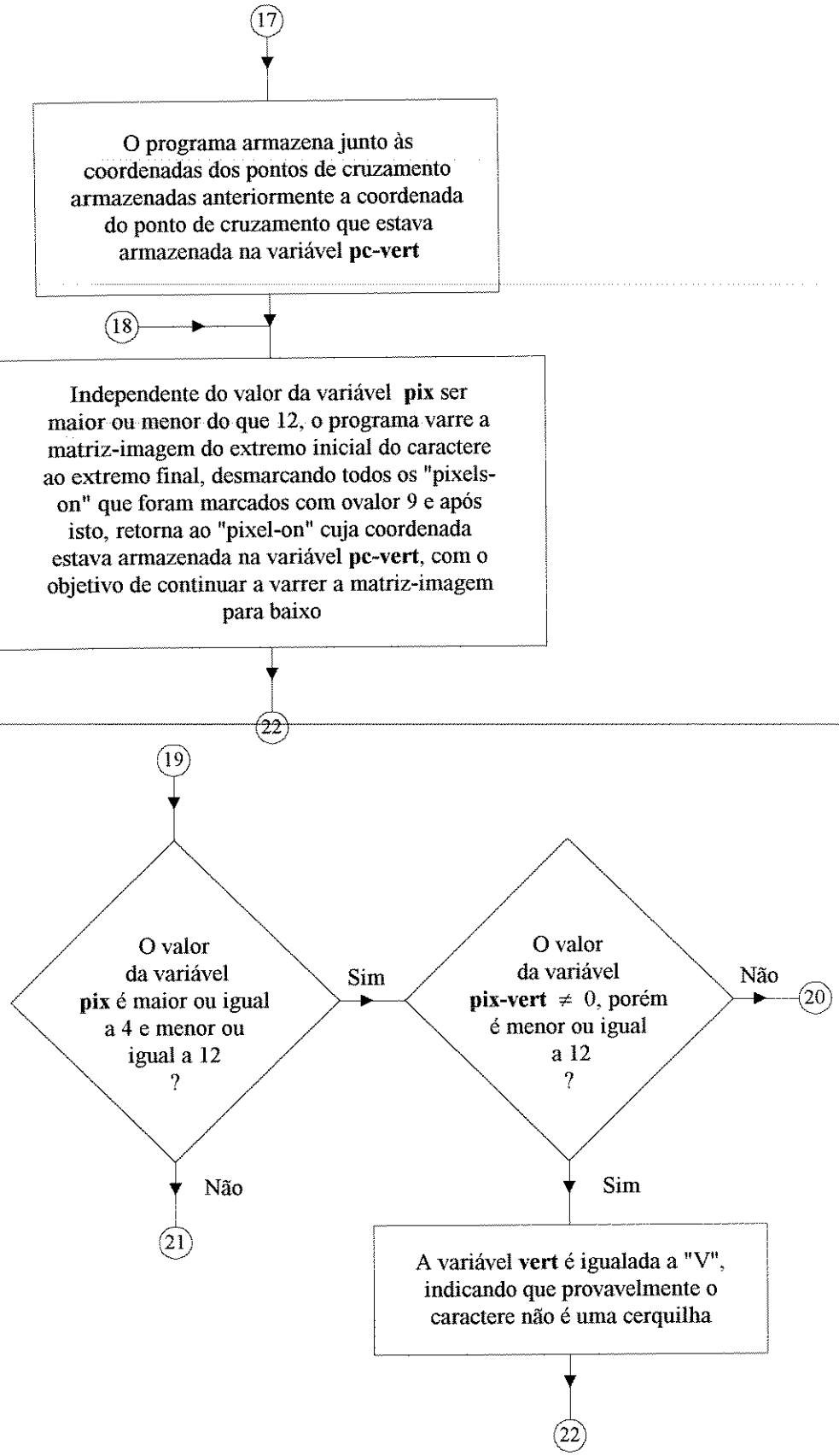


Figura 5.21(m) - Décima terceira parte do fluxograma da função verifica-cerquilha.

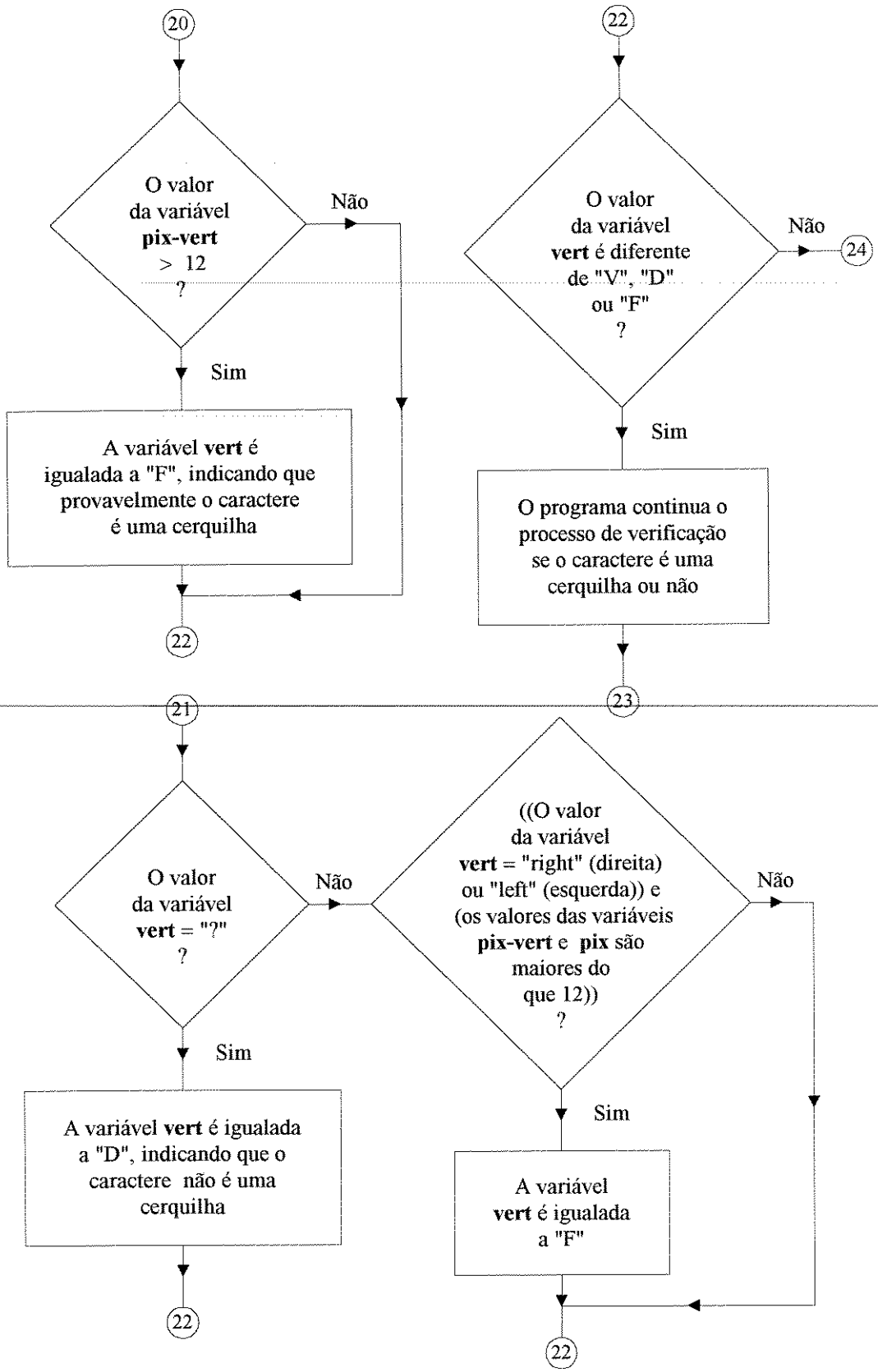


Figura 5.21(n) - Décima quarta parte do fluxograma da função verifica-cerquilha.

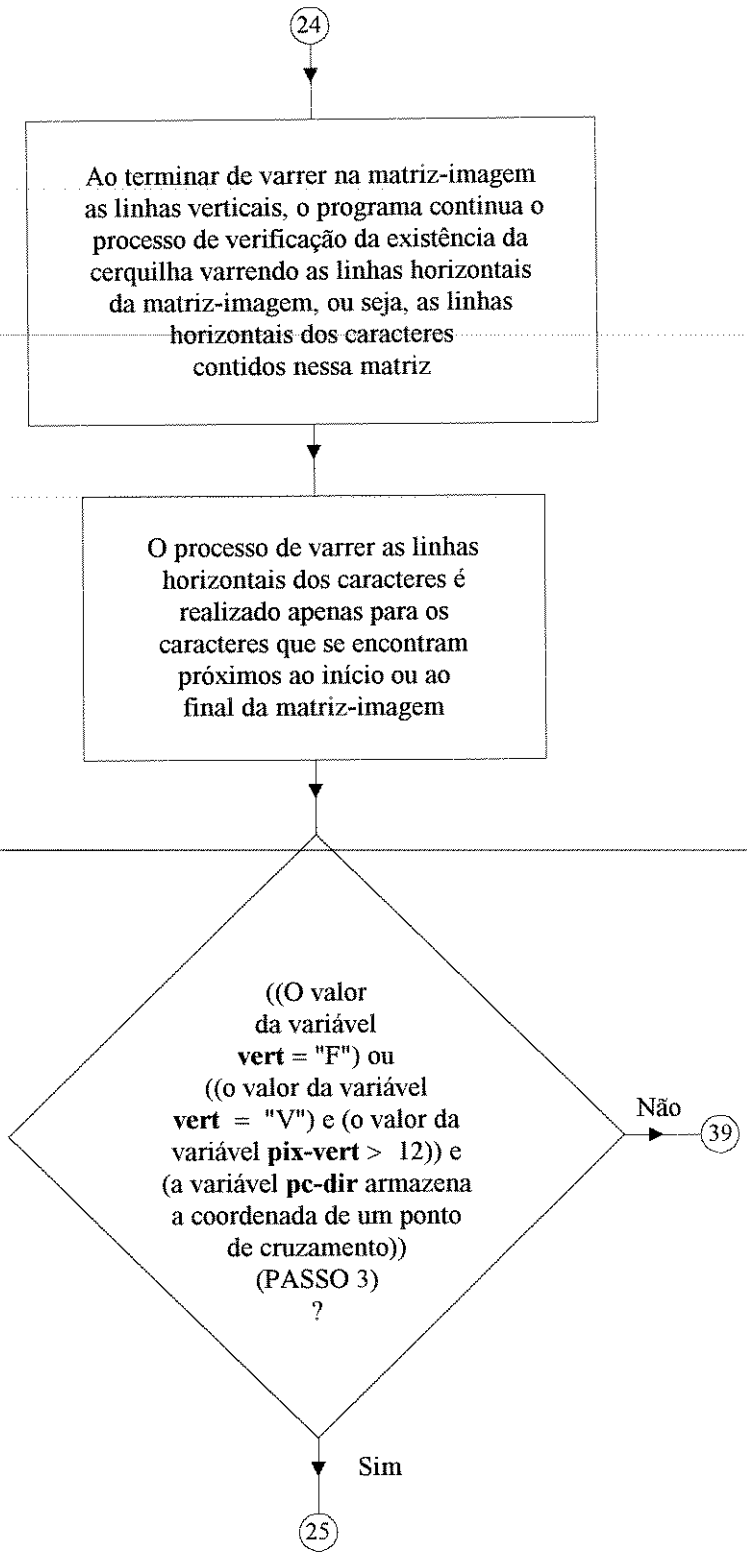


Figura 5.21(o) - Décima quinta parte do fluxograma da função verifica-cerquilha.

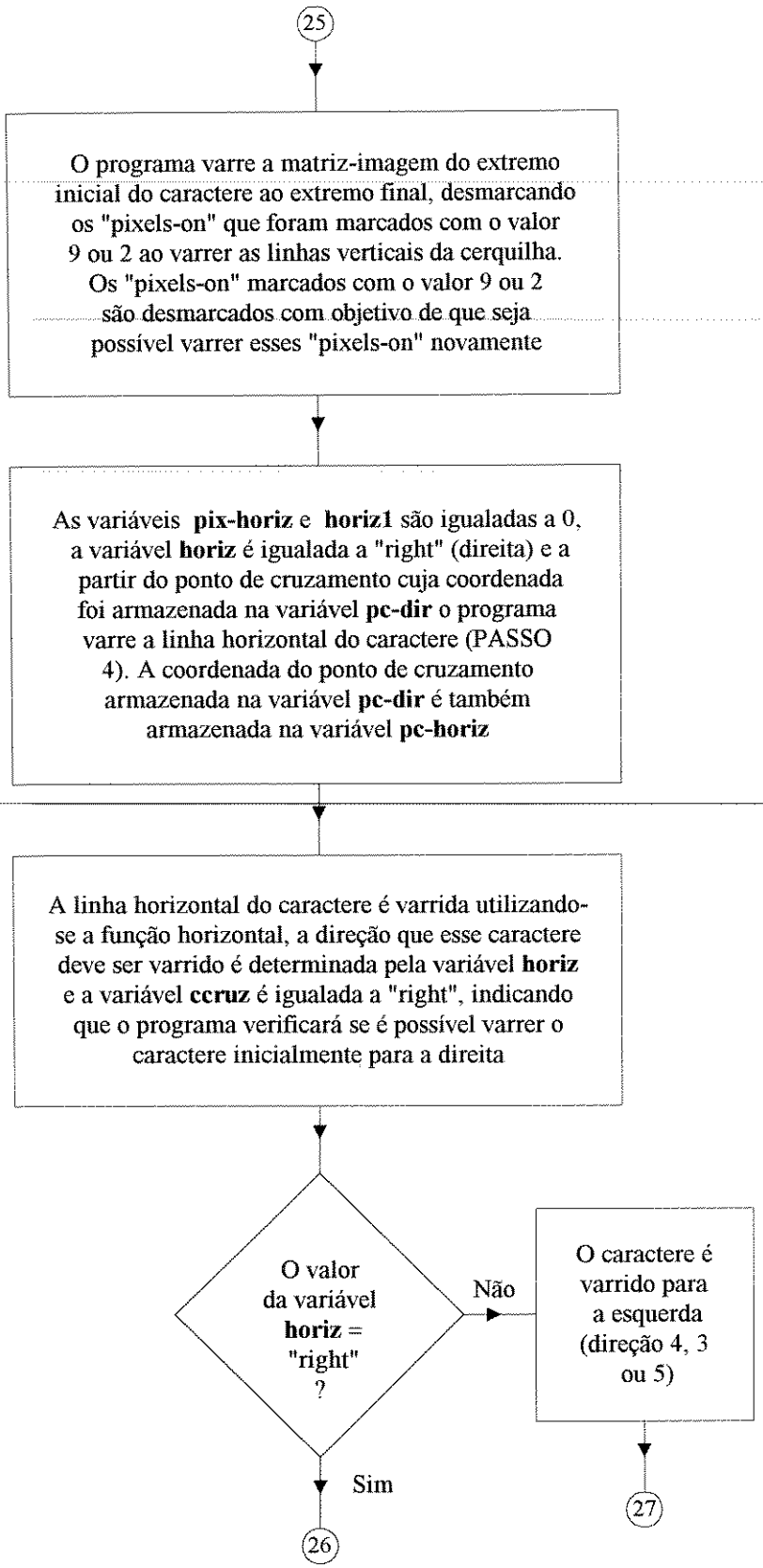


Figura 5.21(p) - Décima sexta parte do fluxograma da função verifica-cerquilha.

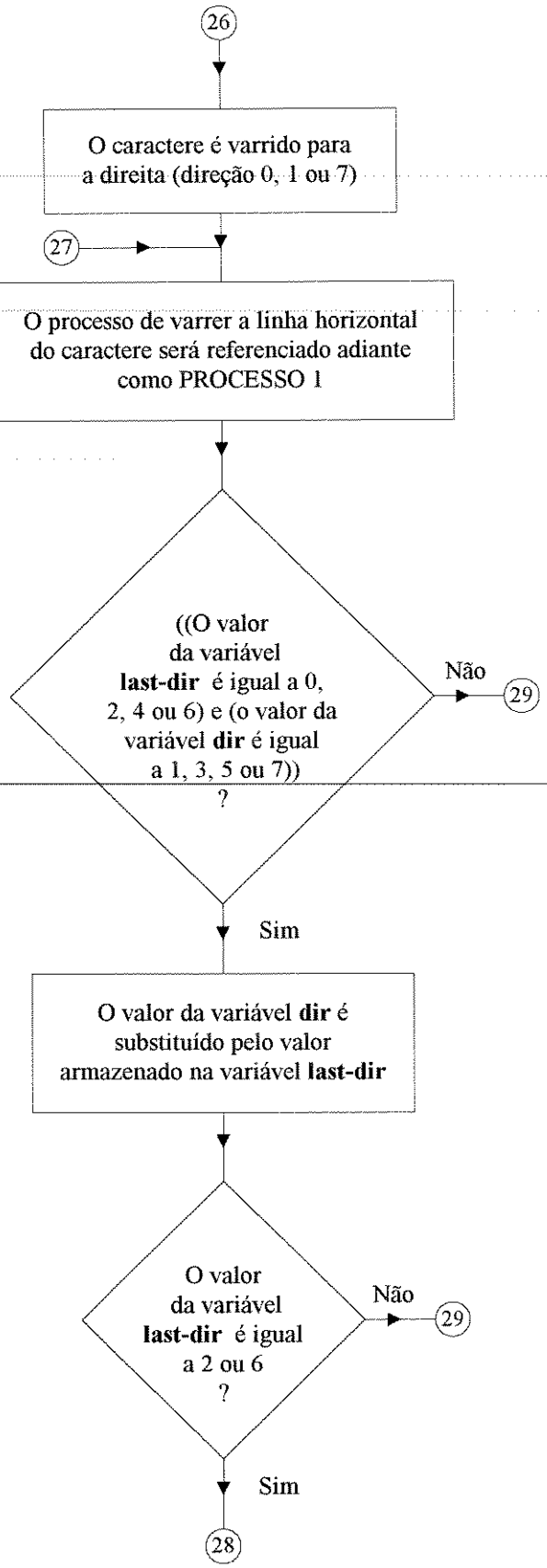


Figura 5.21(q) - Décima sétima parte do fluxograma da função verifica-cerquilha.

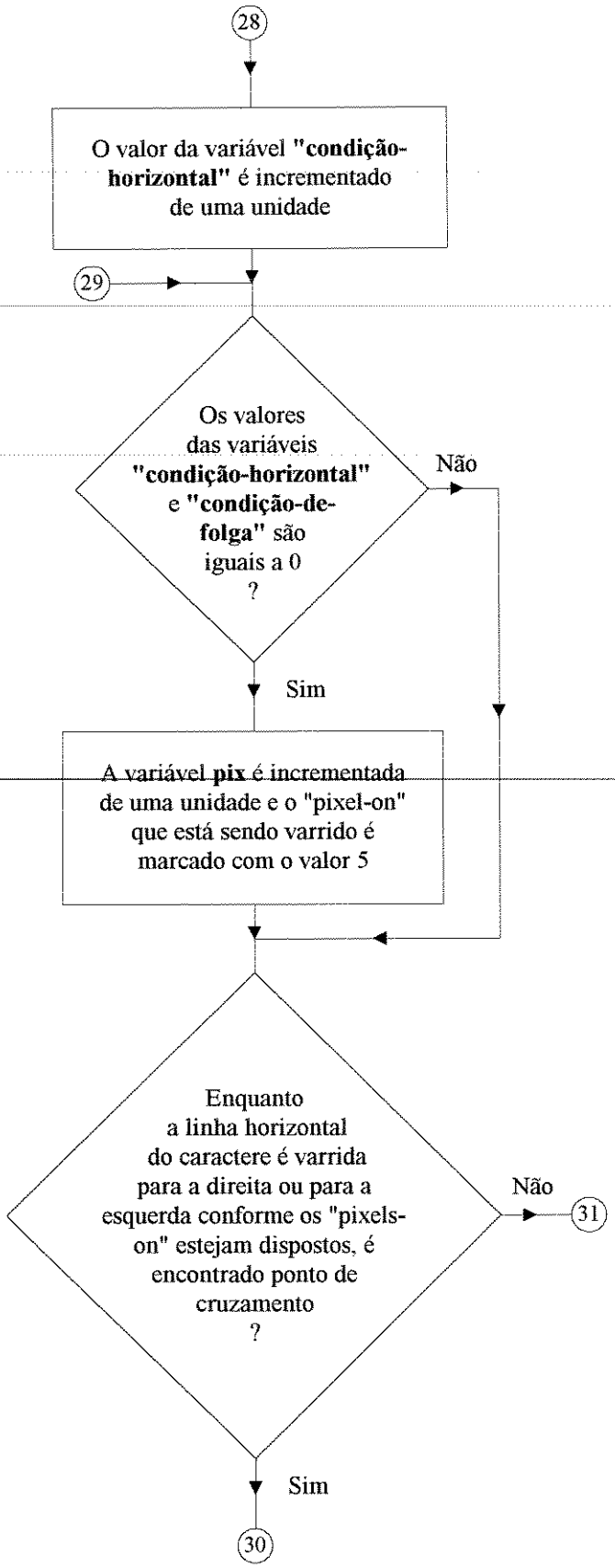


Figura 5.21(r) - Décima oitava parte do fluxograma da função verifica-cerquilha.

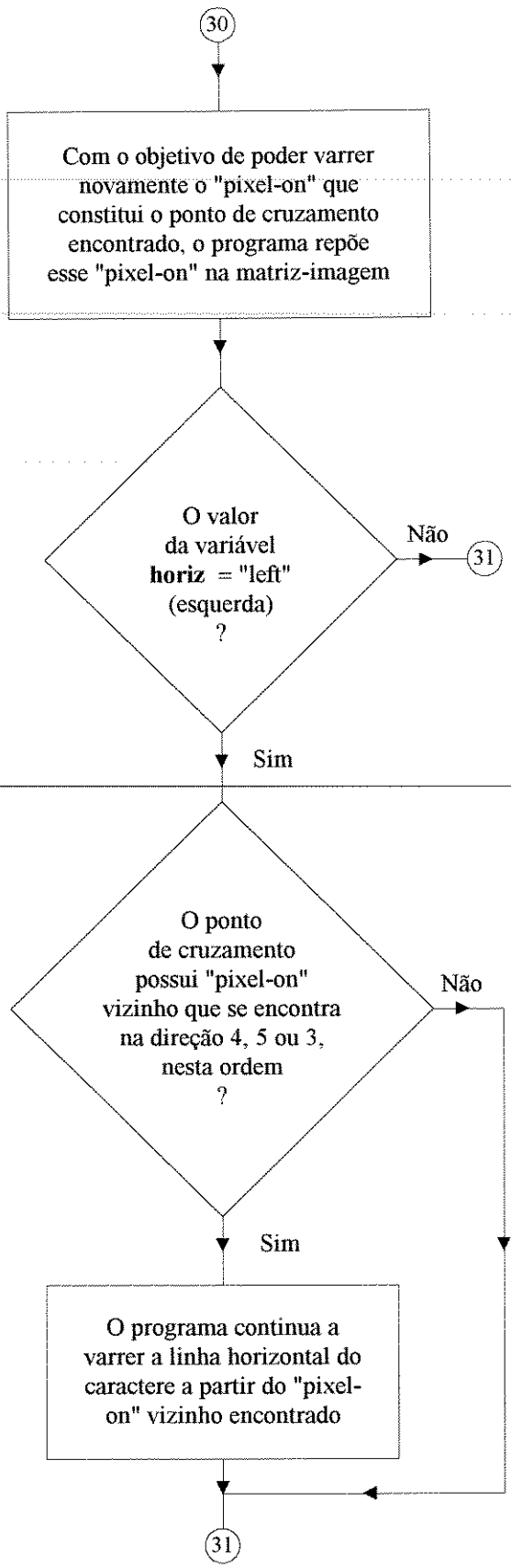


Figura 5.21(s) - Décima nona parte do fluxograma da função verifica-cerquilha.

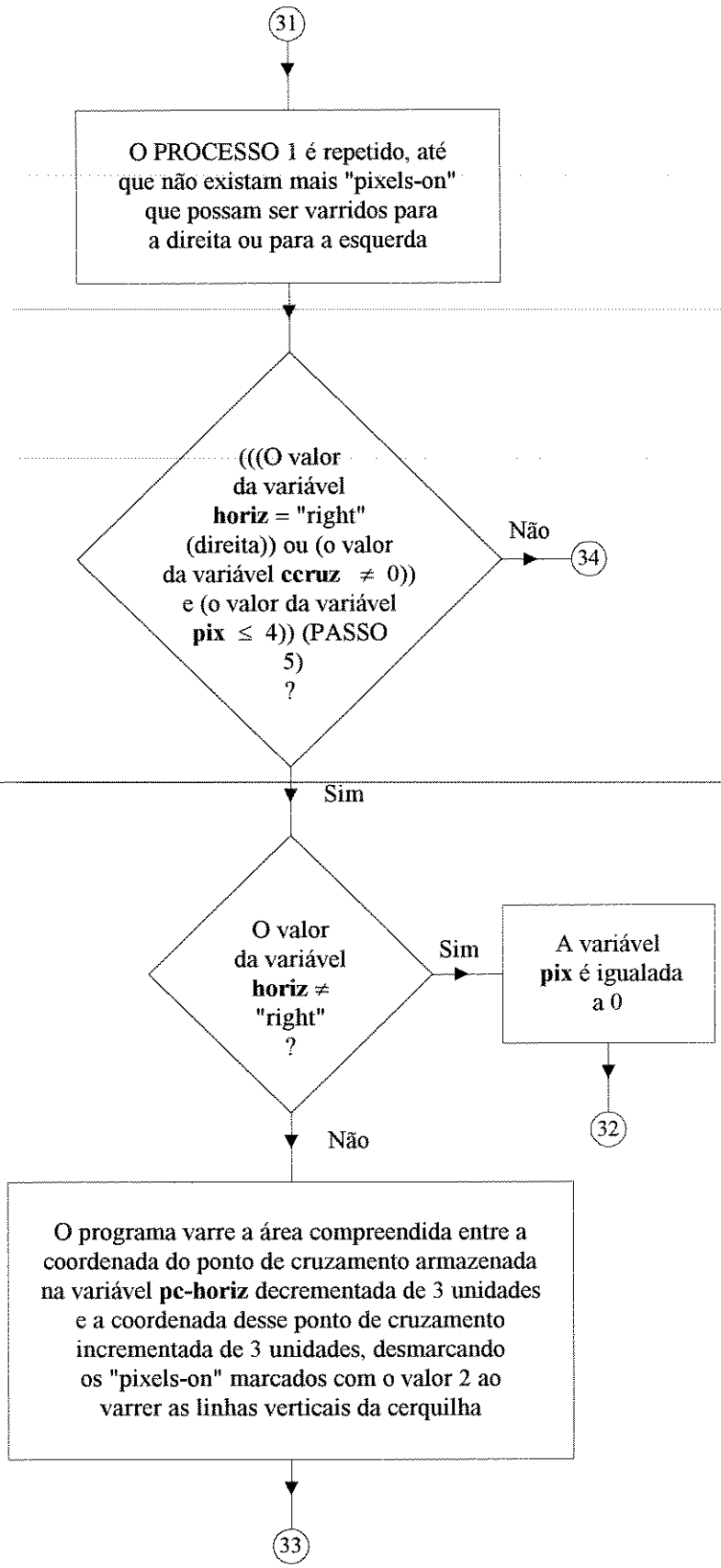


Figura 5.21(t) - Vigésima parte do fluxograma da função verifica-cerquilha.

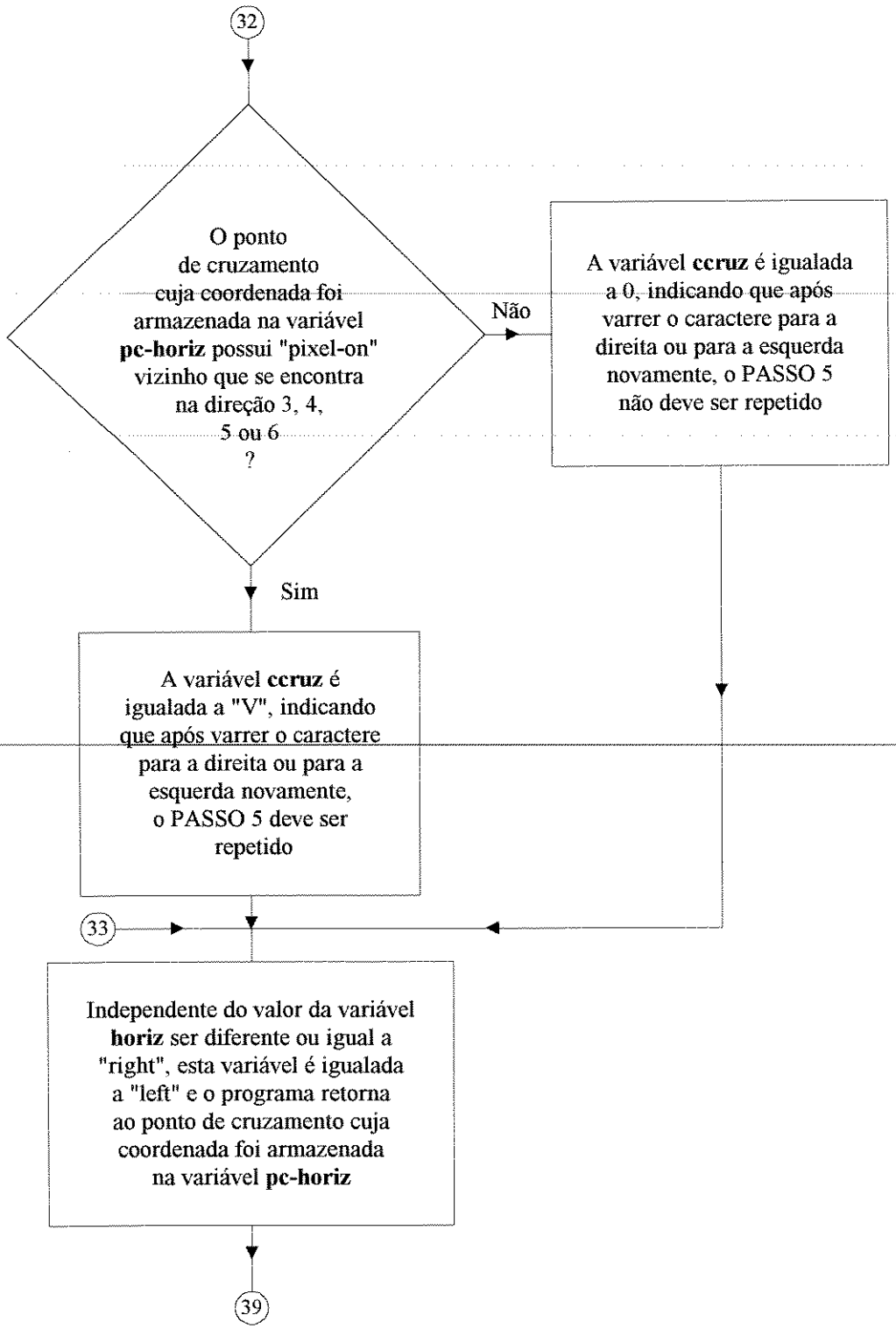


Figura 5.21(u) - Vigésima primeira parte do fluxograma da função verifica-cerquilha.

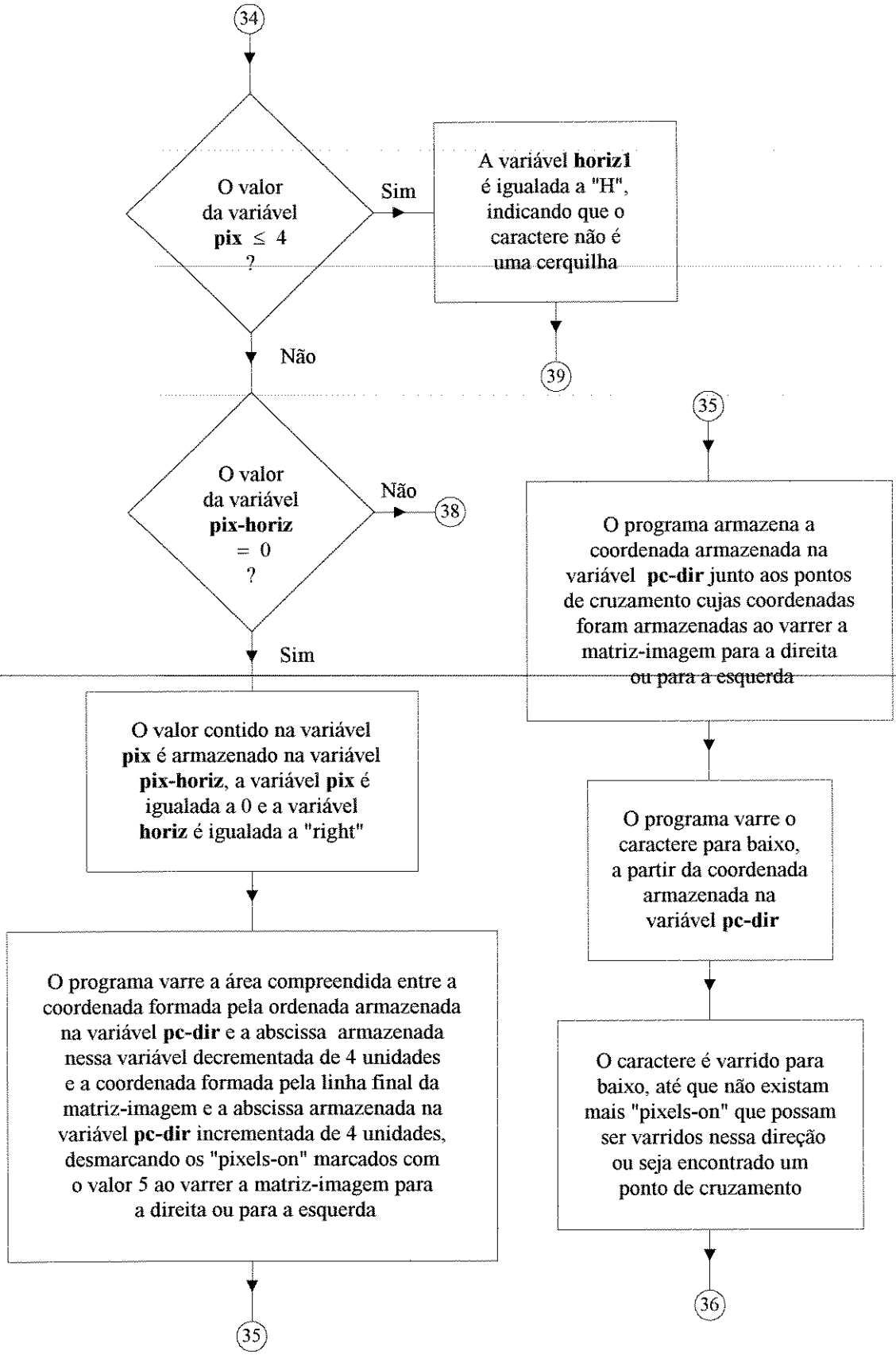


Figura 5.21(v) - Vigésima segunda parte do fluxograma da função verifica-cerquilha.

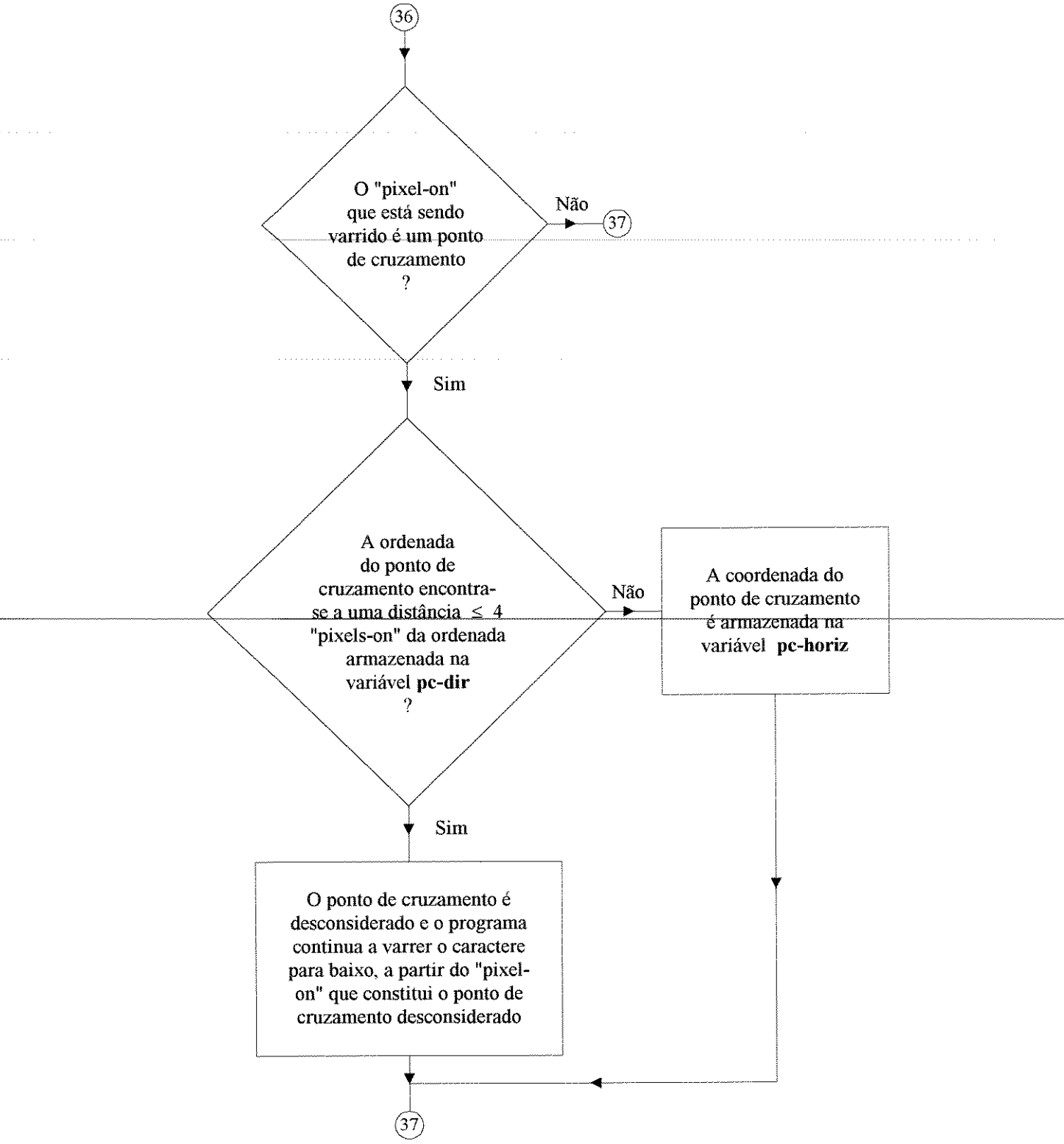


Figura 5.21(w) - Vigésima terceira parte do fluxograma da função verifica-cerquilha.

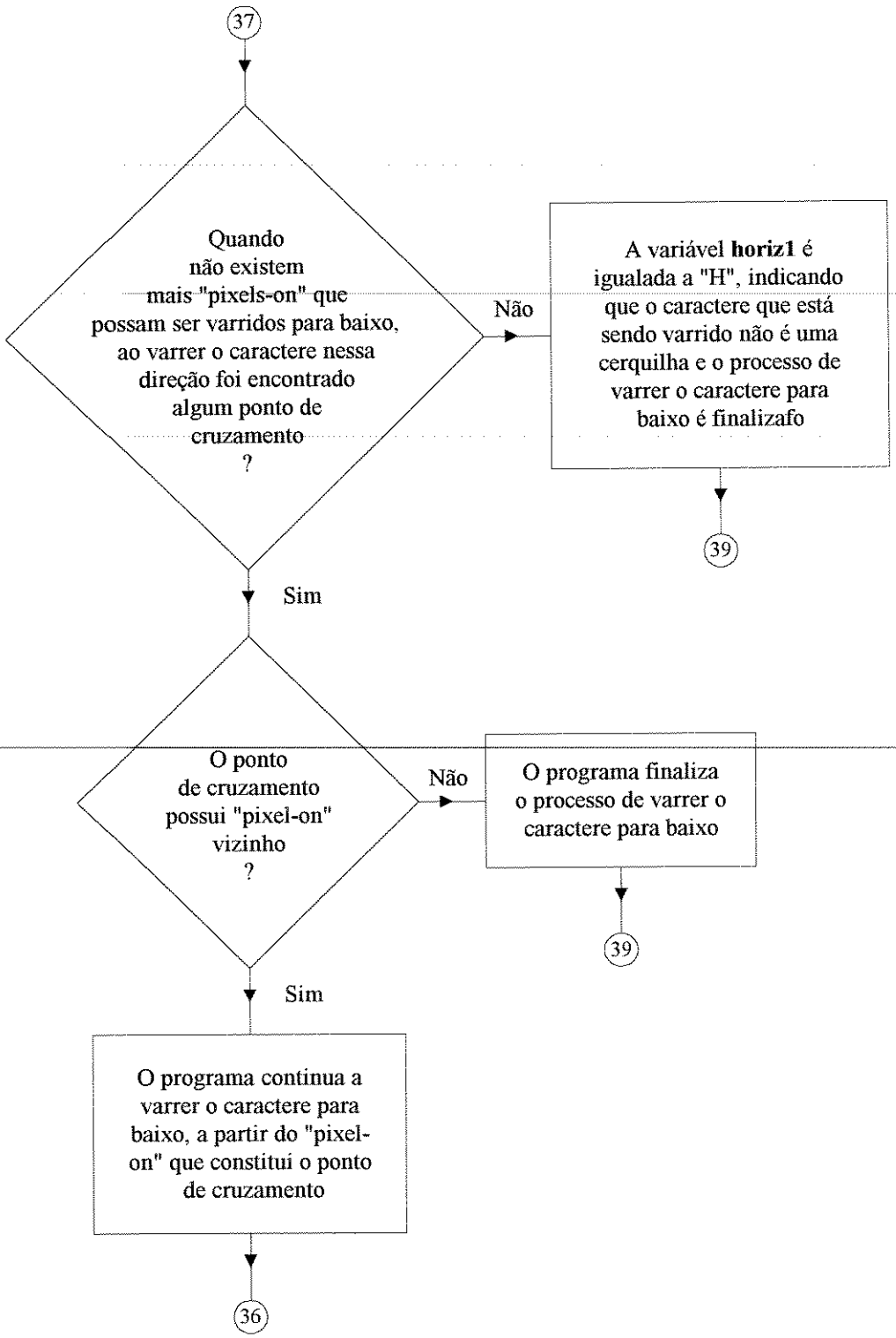


Figura 5.21(x) - Vigésima quarta parte do fluxograma da função verifica-cerquilha.

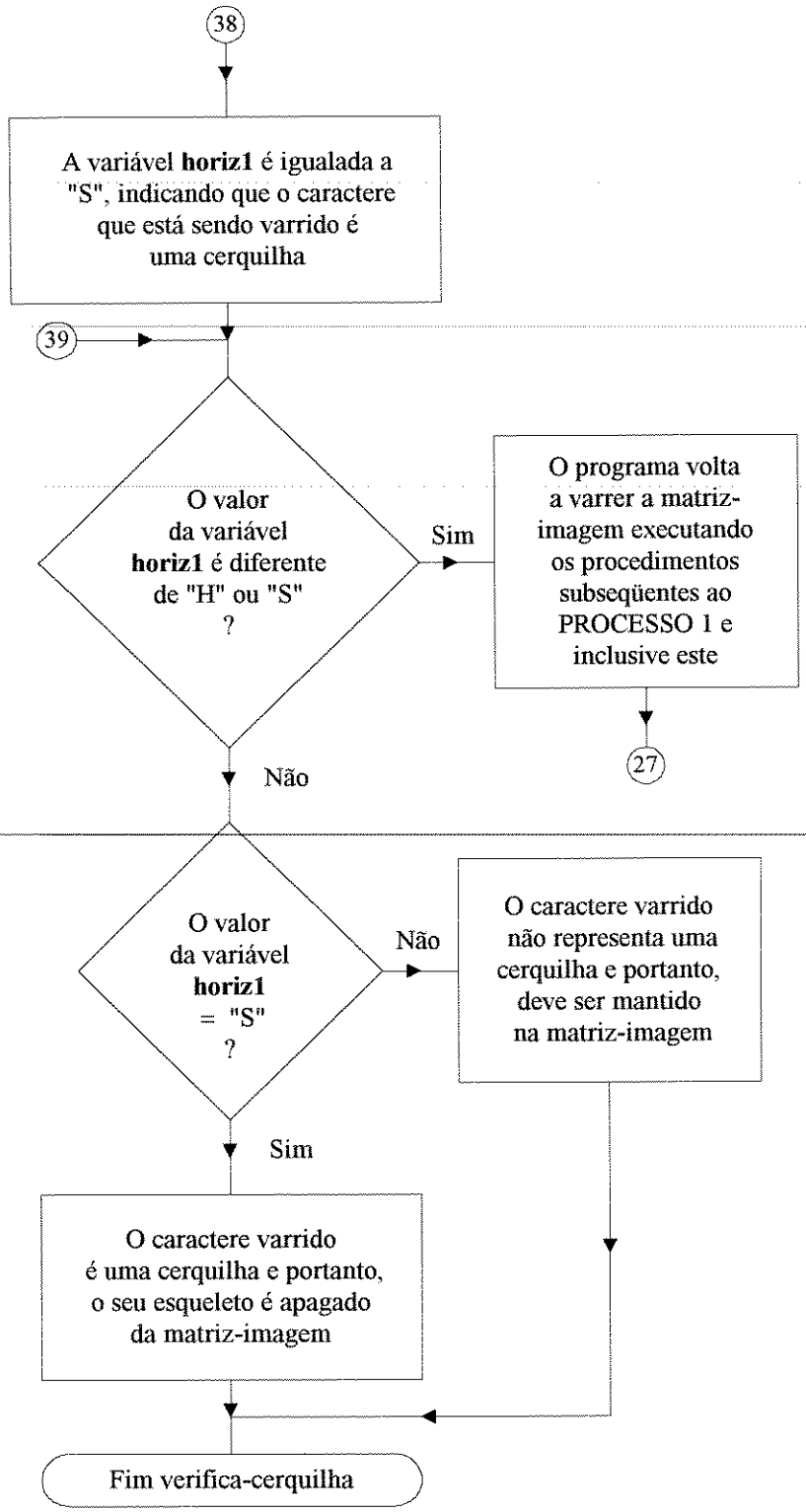


Figura 5.21(y) - Vigésima quinta parte do fluxograma da função verifica-cerquilha.

Fluxograma da função verifica-real

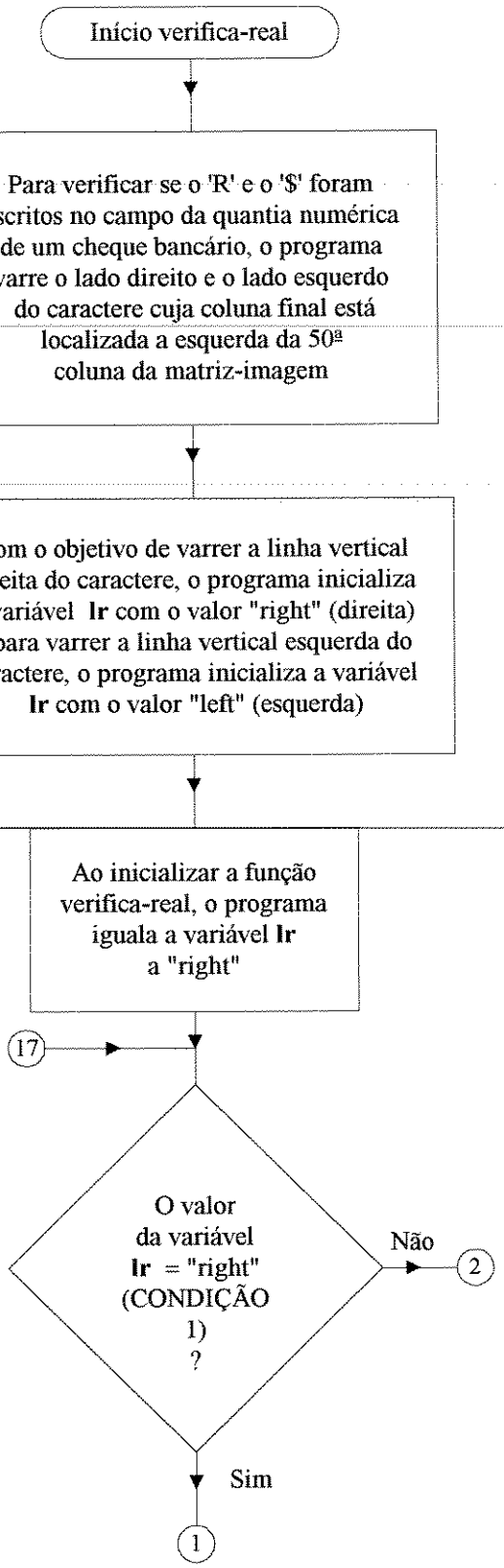


Figura 5.23(a) - Primeira parte do fluxograma da função verifica-real.

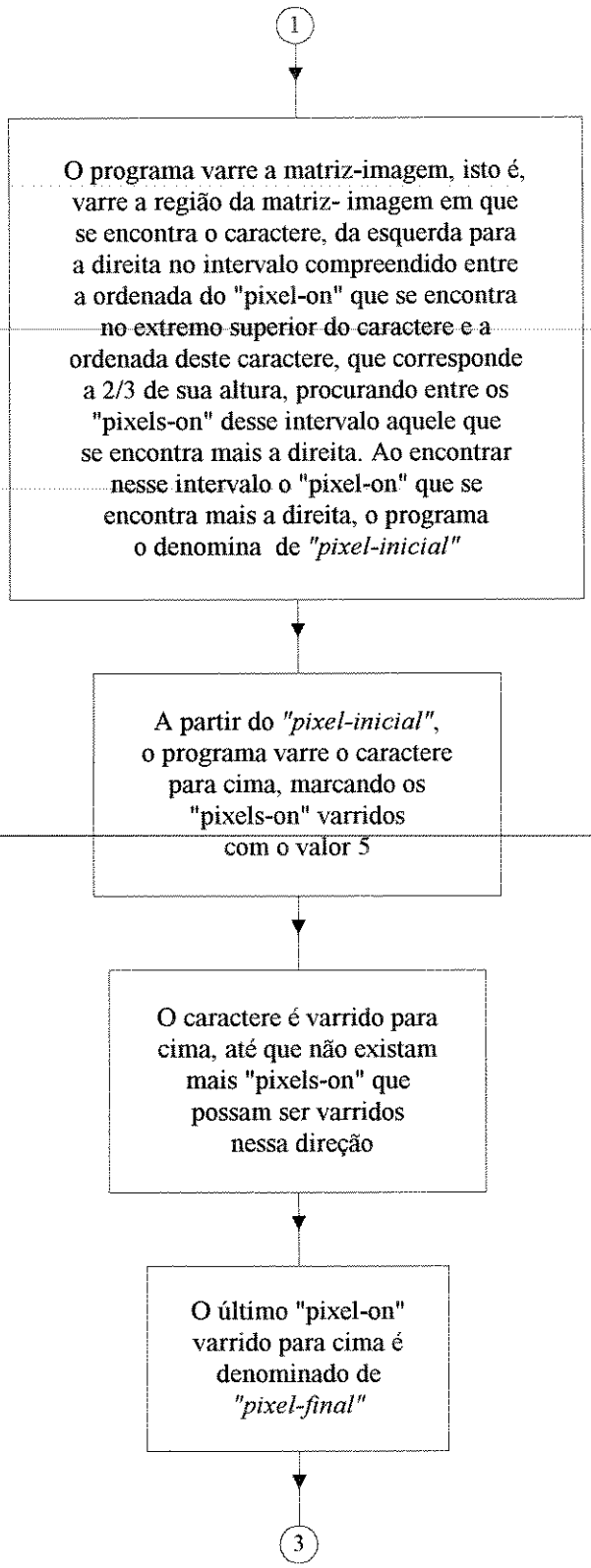


Figura 5.23(b) - Segunda parte do fluxograma da função verifica-real.

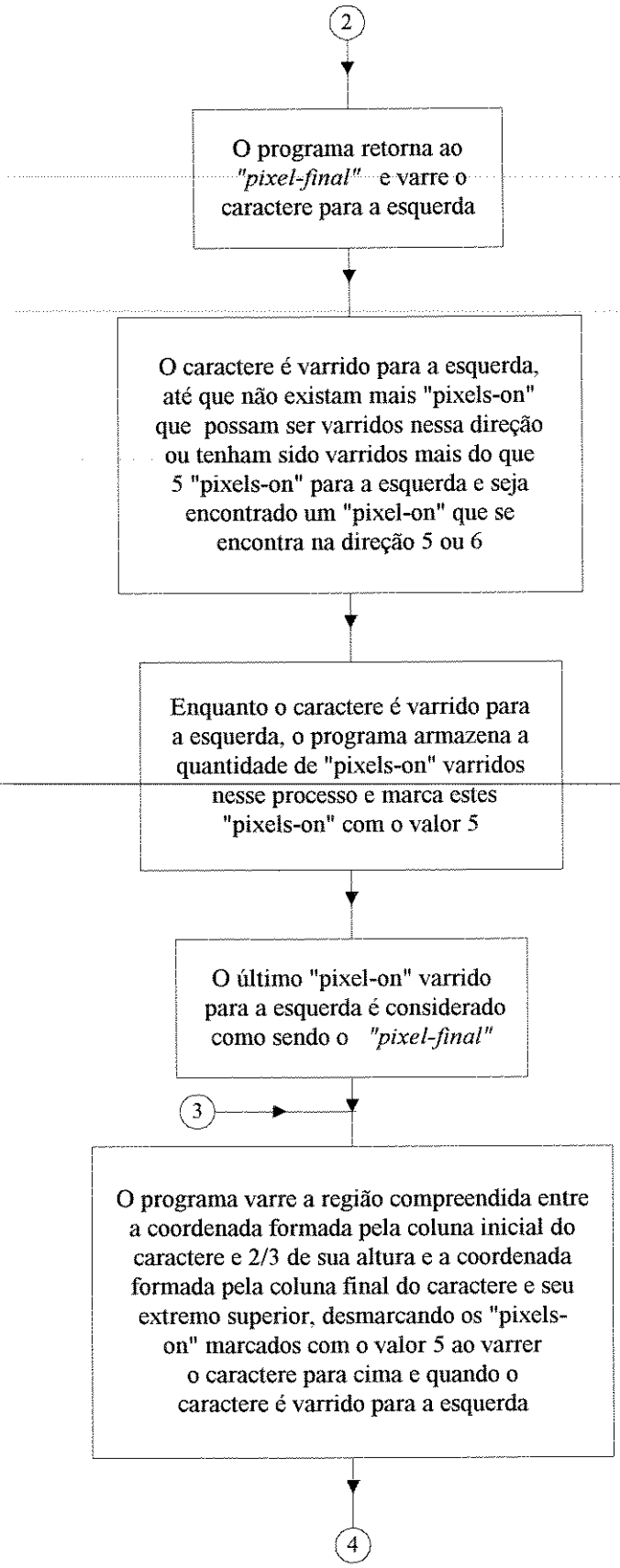


Figura 5.23(c) - Terceira parte do fluxograma da função verifica-real.

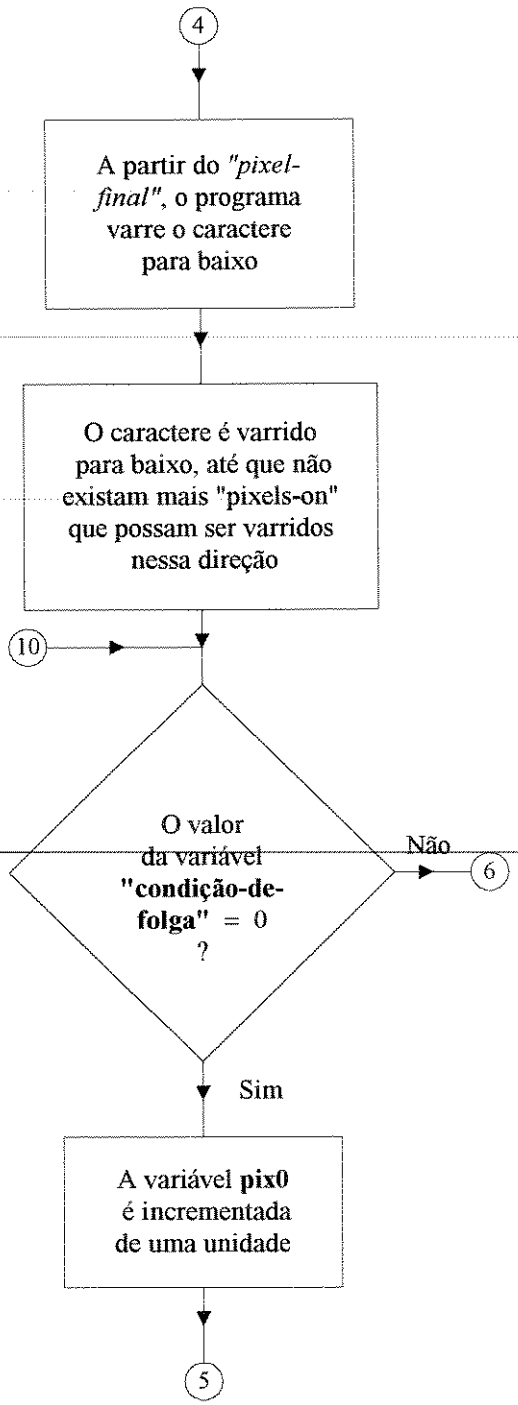


Figura 5.23(d) - Quarta parte do fluxograma da função verifica-real.

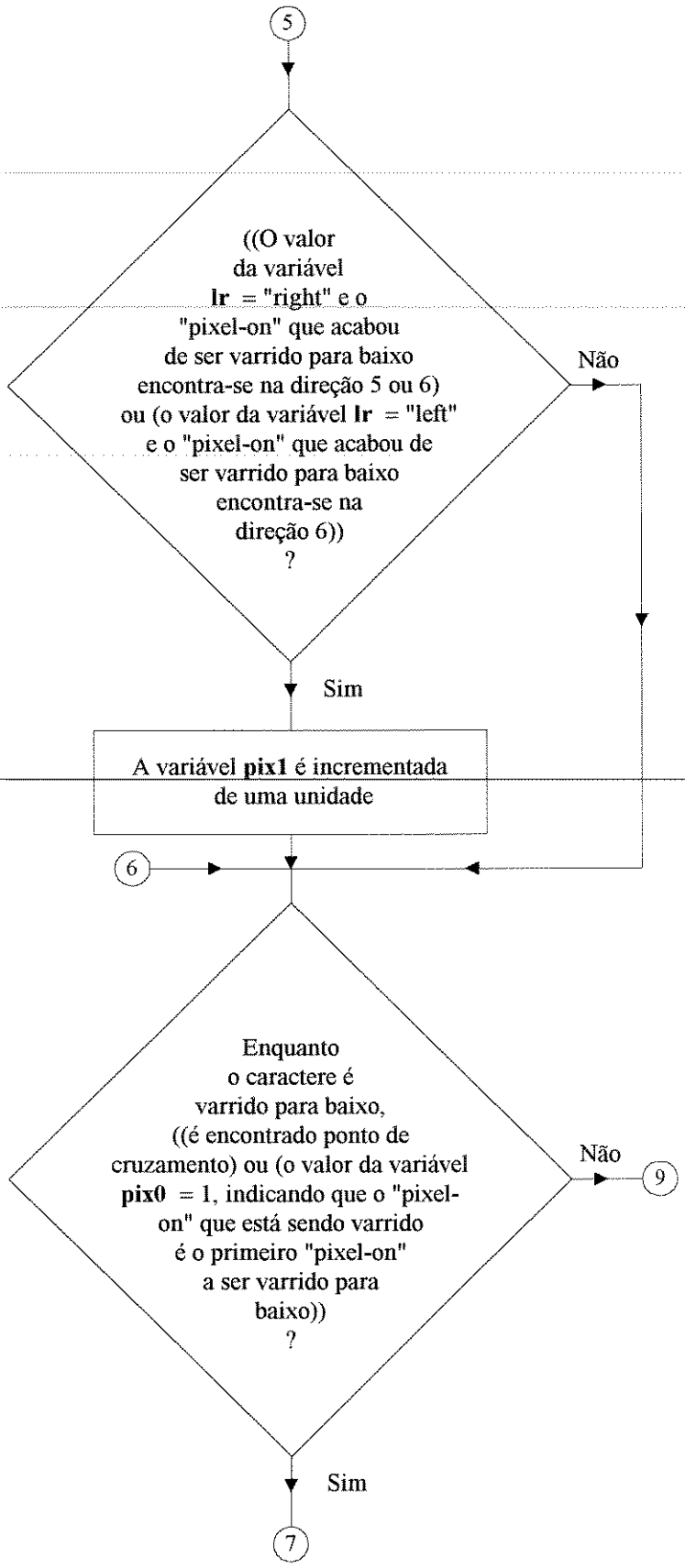


Figura 5.23(e) - Quinta parte do fluxograma da função verifica-real.

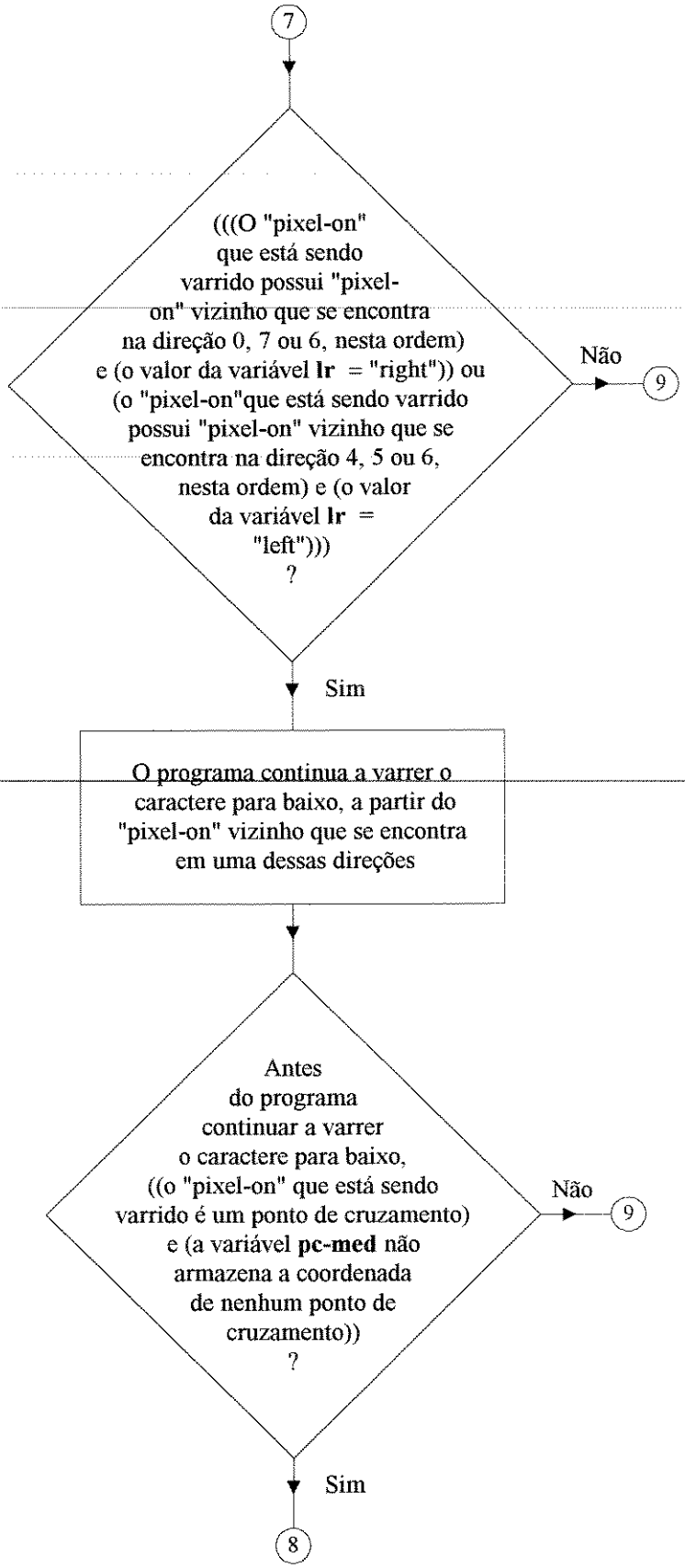


Figura 5.23(f) - Sexta parte do fluxograma da função verifica-real.

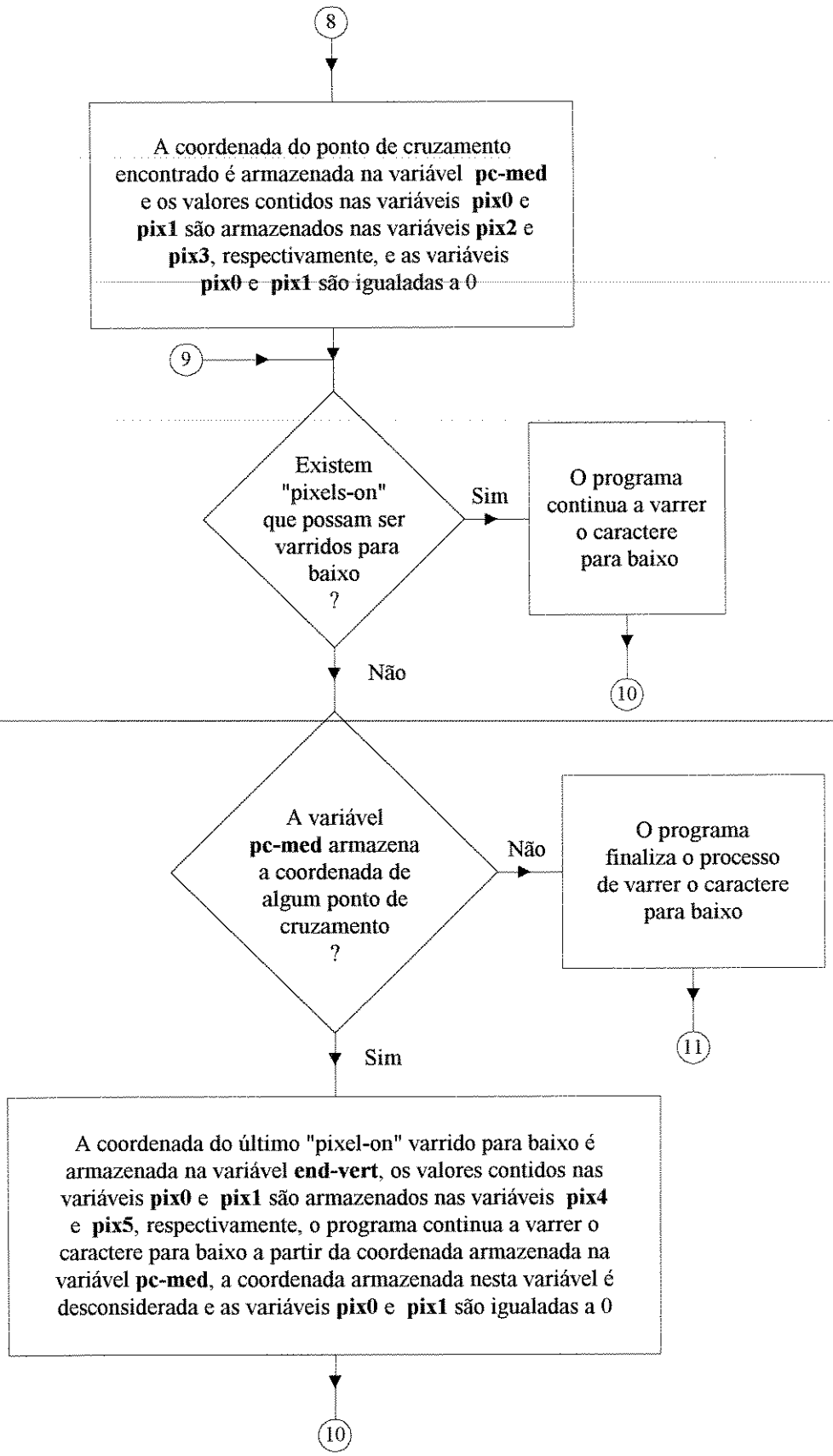


Figura 5.23(g) - Sétima parte do fluxograma da função verifica-real.

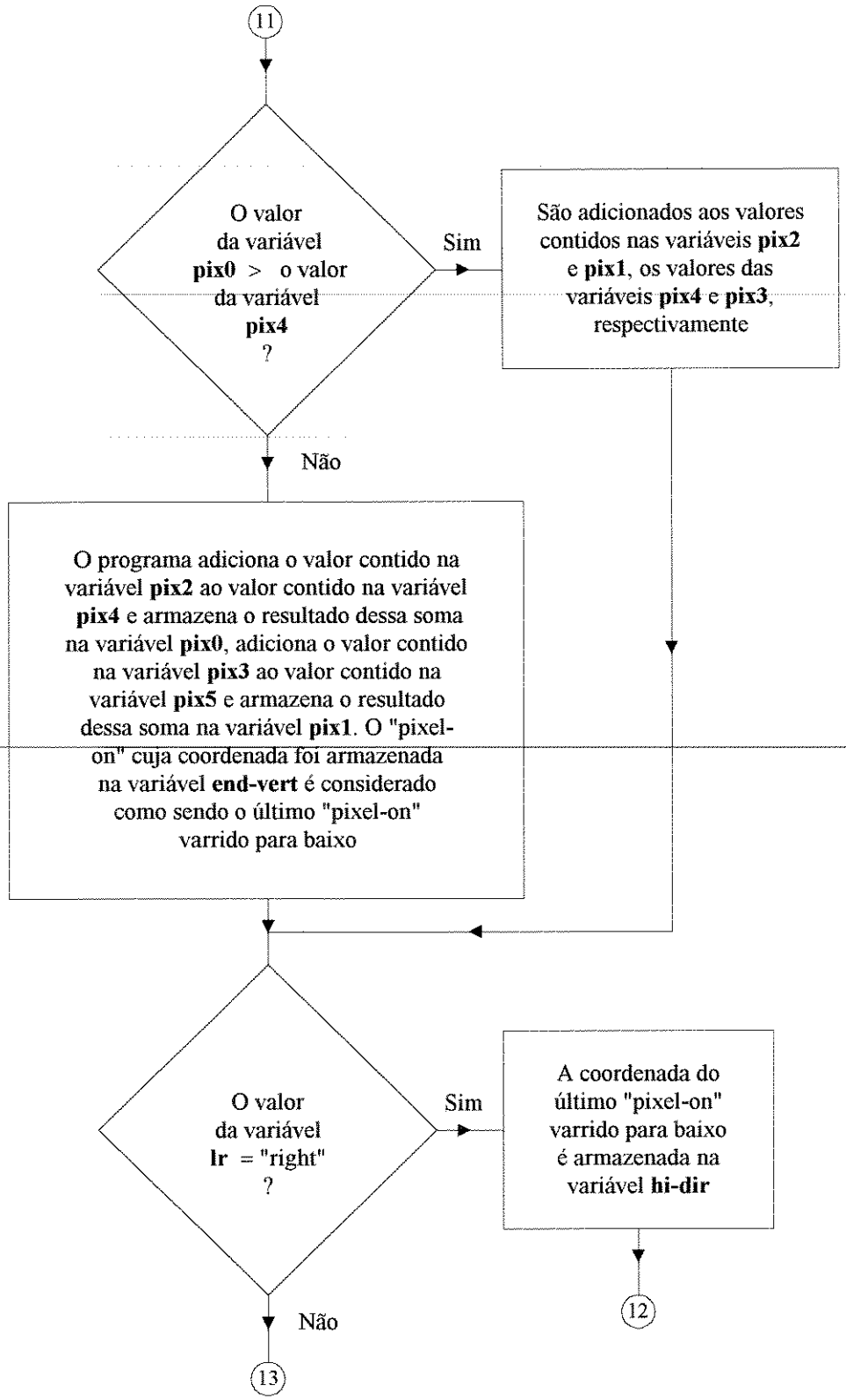


Figura 5.23(h) - Oitava parte do fluxograma da função verifica-real.

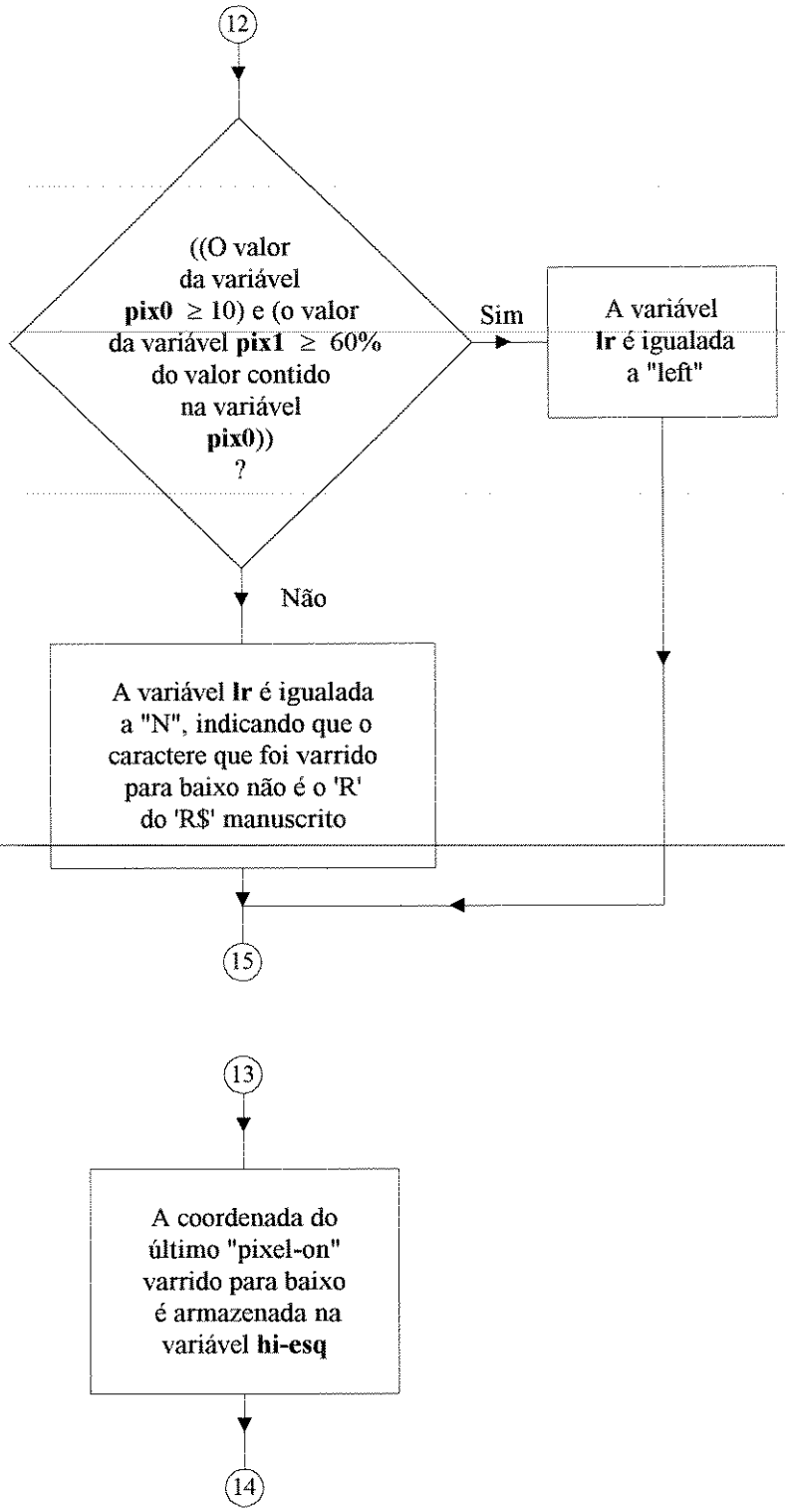


Figura 5.23(i) - Nona parte do fluxograma da função verifica-real.

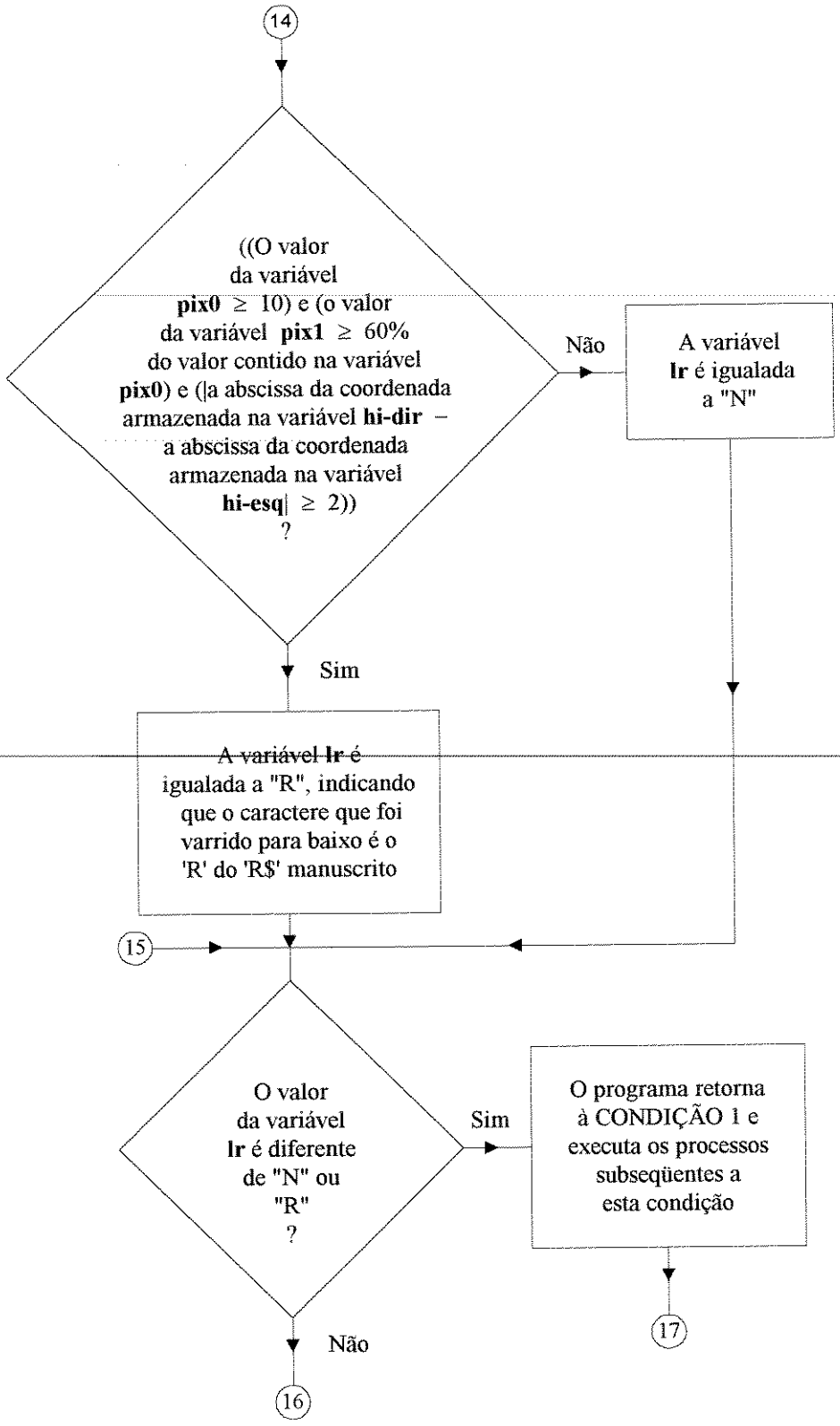


Figura 5.23(j) - Décima parte do fluxograma da função verifica-real.

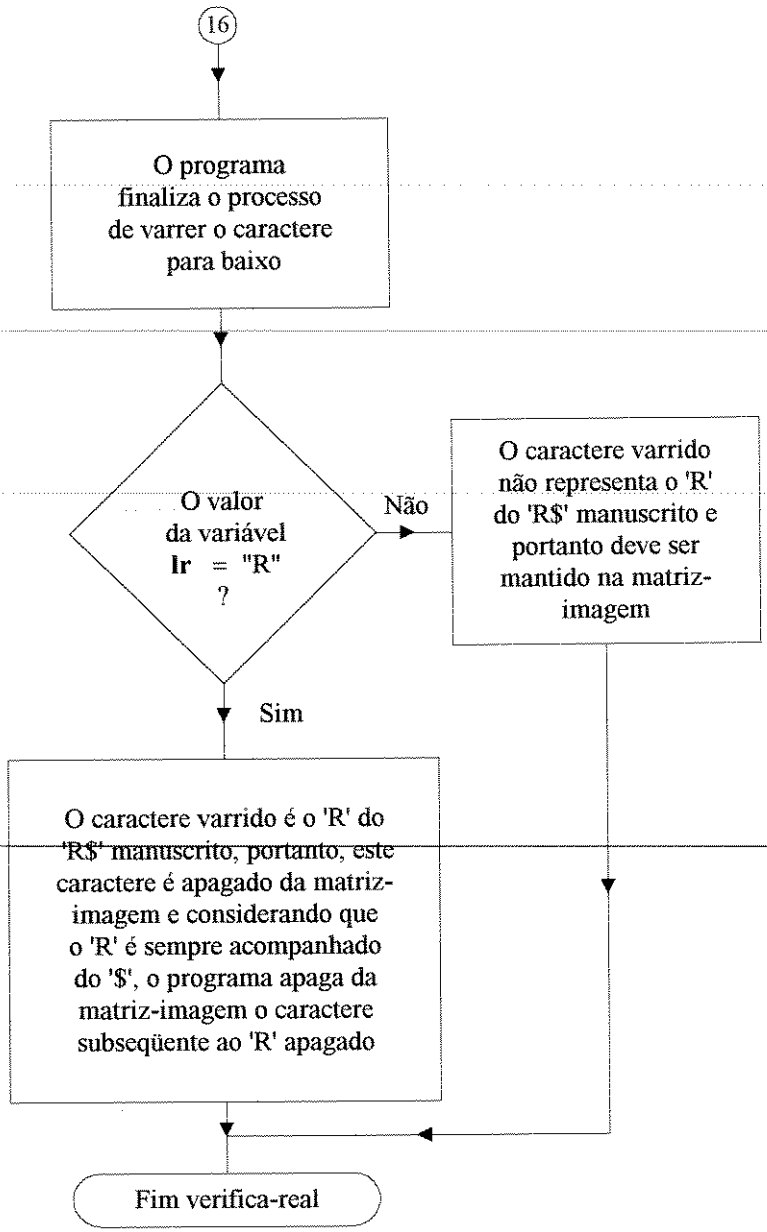


Figura 5.23(k) - Décima primeira parte do fluxograma da função verifica-real.