UNIVERSIDADE ESTADUAL DE CAMPINAS

FACULDADE DE ENGENHARIA ELÉTRICA E DE COMPUTAÇÃO

DEPARTAMENTO DE ENGENHARIA DE COMPUTAÇÃO E AUTOMAÇÃO INDUSTRIAL

Dissertação de Mestrado

Estudo de Código de Barras por Análise de Imagens

Ricardo Correia Soares

Orientador: **Prof. Dr. Roberto de Alencar Lotufo**

DISSERTAÇÃO DE MESTRADO

UNIVERSIDADE ESTADUAL DE CAMPINAS

FACULDADE DE ENGENHARIA ELÉTRICA E DE COMPUTAÇÃO

DEPARTAMENTO DE ENGENHARIA DE COMPUTAÇÃO E AUTOMAÇÃO INDUSTRIAL

Estudo de Código de Barras por Análise de Imagens

Autor: **Ricardo Correia Soares**

Orientador: **Prof. Dr. Roberto de Alencar Lotufo**

Banca examinadora:

Prof. Dr. Roberto de Alencar Lotufo – FEEC / UNICAMP

Prof. Dr. Clésio Luis Tozzi – FEEC / UNICAMP

Prof. Dr. Jacques Facon – PPGIA PUC / PUC - Paraná

Dissertação submetida à Faculdade de Engenharia Elétrica e de Computação da Universidade Estadual de Campinas, para preenchimento dos pré-requisitos parciais para obtenção do Título de Mestre em Engenharia Elétrica. Área de concentração: Engenharia de Computação.

12 de novembro de 2001

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

So11e

Soares, Ricardo Correia
Estudo de Código de barras por análise de imagens / Ricardo Correia Soares.--Campinas, SP: [s.n.], 2001.

Orientador: Roberto de Alencar Lotufo.
Dissertação (mestrado) - Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação.

1. Código de barras. 2. Processamento de imagens. I. Lotufo, Roberto de Alencar. II. Universidade Estadual de Campinas. Faculdade de Engenharia Elétrica e de Computação. III. Título.

Resumo

Neste trabalho, descrevemos técnicas para decodificação de Códigos de Barras unidimensionais (EAN, UPC e Code 39). A decodificação é realizada através de Análise de Imagens e Reconhecimento de Padrões. Inicialmente, imagens contendo algum dos códigos acima citados são adquiridas através de digitalização por um leitor ótico de mesa, que gera imagens em 256 níveis de cinza. As técnicas de Análise de Imagens consistem em: estudo da inclinação do eixo longitudinal de cada símbolo presente na imagem e binarização da imagem (transformação da imagem com 256 níveis de cinza em uma nova imagem contendo apenas 2 níveis de cinza). O Processo de Reconhecimento de Padrões envolve as etapas de aquisição de atributos da imagem binarizada, classificação de padrões de larguras e decodificação da informação contida na distribuição de larguras das barras e espaços. Além da eficiência da metodologia que abordamos em casos comuns , onde as barras não sofreram danos graves, observamos que ela é também eficaz em alguns casos críticos, quando as barras sofreram danos.

Abstract

In this work, we described techniques for decoding of Linear Barcodes (EAN, UPC and Code 39). The decoding is accomplished through Images Analysis and Patterns Recognition. Initially, images holding some of the codes mentioned above are acquired through scanner reading, that generates images in 256 gray levels. The techniques of Images Analysis consist in: study of the inclination of the longitudinal axis of each symbol that appears in the image and image thresholding (transformation of the image with 256 gray levels into a new image with only 2 gray levels). The Process of Pattern Recognition involves the stages of acquisition of attributes of the image with 2 gray levels, classification of patterns of widths and decoding of the information held in the widths distribution of bars and spaces. Besides the methodology efficiency that we broached in common cases, in which the bars didn't suffer serious damages, we observed that this is also effective in some critical cases, when the bars suffered damages.

Agradecimentos

Ao Prof. Dr. Roberto de Alencar Lotufo, pela oportunidade de enaltecer meus conhecimentos através de sua orientação sempre perspicaz ao longo deste trabalho.

Aos meus amados pais Renato e Bernadete e aos meus irmãos José Renato e Renan, pelo incessante apoio e incentivo em todos os caminhos que percorri.

À minha querida tia Avani e ao meu tio Hugo, pelo essencial apoio no início da minha chegada à cidade de Campinas.

Ao CNPq, pelo indispensável apoio financeiro durante a realização deste trabalho.

Ao Prof. Dr. Ariovaldo V. Garcia , Coordenador de Pós-Graduação da FEEC, e ao Prof. Dr. José Cláudio Geromel , Pró-Reitor de Pós-Graduação, pelo apoio ao agilizar o processo burocrático no final do curso.

Ao FAEP, pela importante concessão da bolsa auxílio-ponte.

Aos meus amigos do LCA, que sempre acrescentaram algo em minha vida acadêmica.

Aos profissionais da Biblioteca de Engenharia, especialmente à Rose, pelo importantíssimo auxílio na busca por artigos científicos.

Aos profissionais do CECOM, especialmente à Dra. Fernanda, que administraram de forma incomparável a minha saúde.

E a esta magnífica Universidade que é a UNICAMP, que foi minha inesquecível e acolhedora casa ao longo do mestrado, disponibilizando tanta infra-estrutura, indispensável na nobre arte da Educação.

Aos meus pais Renato e Bernadete
e aos meus irmãos José Renato e Renan.

Sumário

Capítulo 1

Introdução

Com a crescente evolução da Automação Comercial, a necessidade de métodos de aquisição de dados de forma rápida e confiável favoreceu o desenvolvimento de diversos meios de automatização, entre eles os Códigos de Barras.

O Código de Barras consiste em um símbolo composto de barras escuras paralelas com espessuras variadas, impressas em uma superfície mediante espaçamentos determinados por especificações técnicas padronizadas de cada tipo de código. Sua principal função é a aquisição de dados para gerenciamento de entrada automatizada de informações em um sistema computadorizado, onde um dispositivo de leitura (***scanner***[1.1]) realiza, através de um processo ótico, uma varredura sobre o Código de Barras, capturando sinais mediante o lançamento de um feixe luminoso e reflexão desta radiação por espaços em branco.

O leitor ótico converte as informações luminosas em dados digitais compatíveis a uma linguagem computacional, capaz de verificar a validação da ***simbologia*** (forma na qual se codifica a informação nas barras e espaços do símbolo).

É importante distinguir os conceitos de ***símbolo*** e ***simbologia*** que serão utilizados ao longo deste trabalho. ***Simbologia*** é a forma como são padronizadas as regras que estabelecem quantidade e larguras das barras e espaçamentos a fim de expressar uma palavra-código (informação codificada que será interpretada por um banco de dados inerente a um sistema computadorizado). Existem inúmeras *simbologias* atualmente utilizadas comercialmente; entre as mais populares destacam-se UPC (Universal Product Code), EAN (European Article Numbering) e Code 39 (Código com 3 elementos largos entre os 9 que o

[1.1] Dispositivo que, ligado a um computador, transforma informações visuais em dados digitais.

compõem), que serão analisados nos próximos capítulos. ***Símbolo*** é a representação gráfica impressa do tipo de *simbologia* adotada, ou seja, é o Código de Barras propriamente dito.

Inicialmente, os Códigos de Barras foram destinados à codificação de licenciamento de veículos, onde o número de série do veículo era codificado através de barras pretas e brancas. Ao longo do tempo, surgiu a necessidade de um aumento na capacidade de armazenamento de informações.

Em 1984, a Automotive Industry Action Group (AIAG) padronizou um código que consistia em um empilhamento de quatro estruturas do Código 39, já existente naquele momento. Este novo código foi destinado à identificação de estoques de mercadorias a serem transportadas.

Atualmente, existem inúmeras simbologias de Códigos de Barras, destinadas a diversas áreas. Cada uma possui características próprias de regras para codificação de cada caractere (letras, números ou caracteres especiais), assim como requisitos de impressão, decodificação, verificação de erros, entre outros. Tais características e requisitos são definidos através da padronização de especificações. Existem diversas organizações responsáveis pelo controle da padronização de especificações dos Códigos de Barras geralmente utilizados hoje, entre elas destacam-se: ANSI (American National Standards Institute), HIBC (Health Industry Business Communications Council), AIAG (Automotive Industry Action Group), AIM[1,2,] (Automatic Identification Manufacturers), UCC (Uniform Code Council).

Basicamente, as simbologias dos códigos diferem-se umas das outras segundo aspectos de representação visual e dos tipos de dados capazes de serem armazenados, além da diferença de capacidade máxima de armazenamento. Esses aspectos indicam a viabilidade ou não do uso de cada uma das simbologias a uma determinada aplicação.

[1,2] www.aimglobal.org

1.1 Benefícios oferecidos pelos Códigos de Barras

- *Melhoria na eficiência operacional* - Uma vez que os Códigos de Barras oferecem um armazenamento de informação mais rápido e seguro, a velocidade de operação é favorecida .

- *Redução de Tempo* - Dependendo da aplicação, a redução de tempo gasto pode ser um fator importante. O registro automático de informações de lotes de mercadorias transportadas e dos preços de produtos em caixas de supermercados são exemplos deste benefício.

- *Redução de Erros* - A incidência de erros em uma operação pode ser uma fonte potencial a um aumento de custos e problemas. Existem diversas situações que expressam bem a importância da exatidão dos dados a serem analisados, umas delas estão presentes na área farmacêutica e de análise em bancos de sangue.

- *Redução de Custos* - Os equipamentos exigidos em um sistema de codificação por meio dos Códigos de Barras são de baixo custo, além disso, a confiabilidade e rapidez de leitura favorece a uma redução de gastos.

- *Benefícios ao usuário* - os equipamentos de leitura e impressão são flexíveis e fáceis de se instalar. Além disso, o usuário pode padronizar suas próprias etiquetas, interagindo o usuário às suas necessidades.

1.2 Estrutura

A estrutura do Código de Barras é mostrada na Figura 1.1.

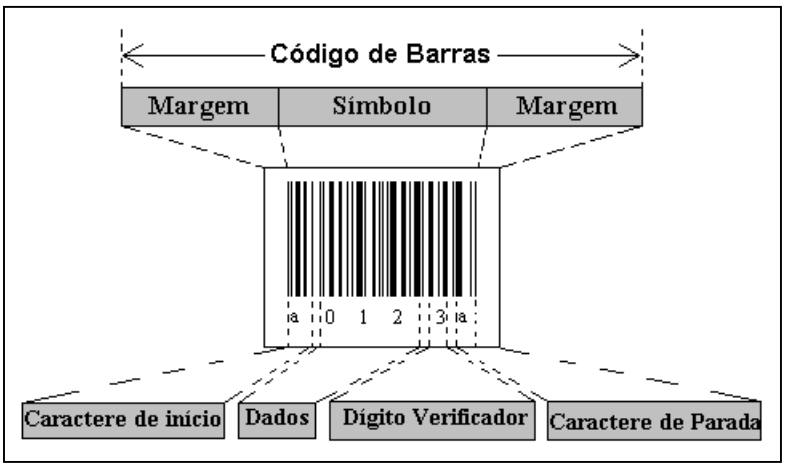


Figura 1.1

Margens Inicial e Final de Silêncio (ou Zonas de Silêncio) - são espaços geralmente brancos, onde nada é impresso. Estas margens devem possuir dez vezes a largura da barra mais estreita do código.

Símbolo - consiste na área composta de barras verticais paralelas espaçadas de forma variada.

Caractere de Início - indica o início do dado. Este caractere varia conforme o sistema de codificação.

Dados - área destinada ao conteúdo de informações a serem codificadas.

Dígito Verificador – certifica a ocorrência erro na leitura através de um dígito de dado interpretado.

Caractere de Parada - indica o término do dado.

1.3 Processo de Leitura

Cada caractere codificado é representado por um padrão de barras largas e estreitas. O leitor ótico usa um ***foto-sensor*** (dispositivo que converte energia luminosa em energia elétrica) para converter o Código de Barras em um sinal elétrico que será transformado em bits à medida em que percorre o código, calculando a relação entre as larguras de barras e espaços. Dessa forma, ocorre uma tradução dos diferentes padrões impressos em caracteres inteligíveis a um sistema computadorizado.

A codificação pode ser feita por ***Largura de Módulo***, onde o valor lógico "zero" é representado por barras estreitas e o valor lógico "um" é representado por barras largas ou por ***Análise de Reflexão***, onde as barras brancas (refletivas) indicam o valor lógico "zero" e as pretas (não-refletivas) representam o valor lógico "um".

Os caracteres de início e de parada do código, juntamente com o dígito verificador, informam ao leitor ótico o sentido de varredura do feixe luminoso.

O dígito verificador é calculado no instante da impressão do código. Ao longo da varredura do leitor ótico sobre o código, uma checagem de soma é calculada e deve coincidir com aquela impressa no símbolo. Caso isto não ocorra, o leitor ótico interpreta esta discrepância como um erro de leitura e realiza uma nova varredura.

1.4 Proposta do trabalho

Neste trabalho pretendemos, através de técnicas de análise de imagens e reconhecimento de padrões, decodificar uma simbologia apresentada em uma imagem, analisando aspectos de rotação, binarização, decodificação e direcionamento de varredura.

1.5 Estrutura do trabalho

No capítulo seguinte, apresentamos conceitos essenciais para uma visão geral de estrutura das simbologias que serão estudadas, incluindo algumas teorias aplicadas para decodificação.

Técnicas de Processamento de Imagens e análises de procedimentos para decodificação estão presentes nos capítulos 3, 4 e 5.

O capítulo 6 apresenta Análises de Verificação de Qualidade das simbologias, e finalizando o trabalho, o capítulo 6 apresenta as conclusões obtidas.

Nossas conclusões estão presentes no capítulo 7.

Finalizamos o trabalho com os apêndices A, B e C onde são apresentados a Tabela de Bandeiras do Código EAN, os algoritmos implementados em linguagem MATLAB® e resultados alcançados através de imagens de entrada e saída.

Capítulo 2

Conceitos Iniciais

Apresentamos neste capítulo, diversos conceitos que envolvem a descrição dos Códigos de Barras estudados e métodos para identificação precisa de suas medidas. Serão apresentados também breves históricos das simbologias estudadas, assim como as tabelas de especificações que regem cada uma delas.

2.1 Princípios de Codificação

Conforme indicado no capítulo anterior, existem basicamente dois princípios de codificação de informação em uma dimensão. Uma delas consiste na subdivisão de intervalos em ***módulos*** (menor unidade da medida de largura de uma barra ou espaçamento) que assumem valores "1 " ou "0", onde os módulos com valores "1" são expressos em forma impressa de barra escura (doravante denominada barra), e aqueles com valores "0" são interpretados como espaços entre as barras escuras; caracterizando a ***Codificação por Análise de Reflexão***, também denominada "Código Delta" [3]. Cada barra ou espaço pode conter um ou mais módulos. A outra metodologia, denominada ***Codificação por Largura de Módulo*** assume cada barra ou espaço como um bit, impondo valor lógico "0" para o elemento (seja barra ou espaço) estreito, e valor lógico "1" para o elemento largo. Este tipo de codificação é também denominado Código Binário [4], uma vez que são usadas apenas duas larguras pré-definidas, geralmente na proporção 3:1.

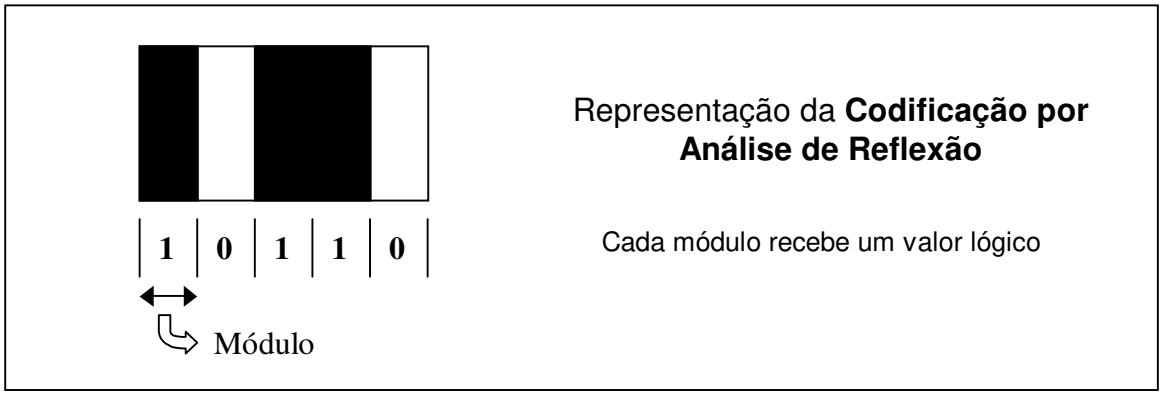


Figura 2.1.a

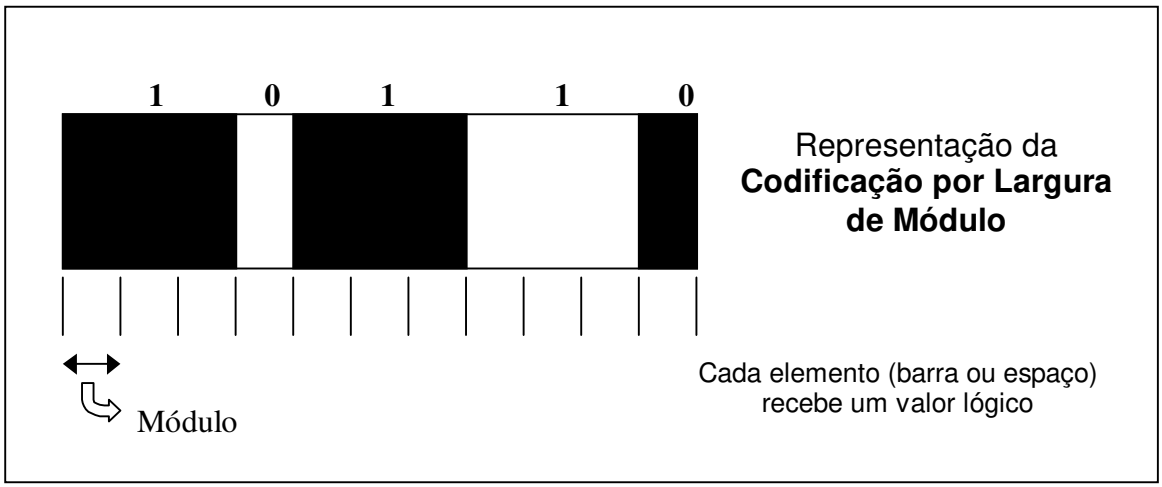


Figura 2.1.b

Analisando a figura 2.1.a, observamos que para codificar a ***string*** [2.1] 10110, pelo método de codificação por reflexão, são necessários cinco módulos, enquanto que para o método de análise de largura de módulo, são necessários onze módulos se a proporção de um elemento estreito para um largo for 1:3.

[2.1] Seqüência de caracteres alfanuméricos ou palavras consecutivas que são manipuladas e tratadas como uma unidade pelo computador

2.2 Pontos Críticos Considerados

Na análise dos Códigos de Barras, um dos pontos mais críticos consiste no problema de interação entre barras impressas, geralmente ocasionado pelo ***Efeito de Espalhamento de Tinta*** (seção 6.1.2). Este problema impede a delimitação precisa entre duas barras, gerando erros de decodificação ou impedindo a decodificação.

Em alguns casos, as larguras das barras são calculadas através da detecção de bordas [5], determinando-se assim a fronteira entre barras e espaços. Entretanto, devido ao problema de *espalhamento de tinta*, esta detecção de bordas muitas vezes gera delimitações incorretas.

Existem métodos[6] que resistem aos processos de borramento; necessitando primeiramente da estimação de desvio padrão de um núcleo (máscara de filtragem) de borramento mínimo na "imagem" da forma de onda capturada. Em seguida, reconhecimentos estatísticos de padrão são usados para classificação dos picos encontrados na forma de onda. Esses métodos fogem do escopo deste trabalho.

2.2.1 Sentido de Varredura

Durante o processo de leitura, é necessário definir o sentido de varredura do leitor ótico, ou seja, o ponto de partida e parada da análise de larguras dos elementos no símbolo impresso. Isso é necessário, pois a princípio, o real posicionamento do símbolo não é conhecido. Eventualmente o símbolo pode estar sendo lido do seu ponto final para o inicial.

Na nossa análise, antes da varredura, o eixo longitudinal do símbolo deverá estar na posição horizontal, mediante algum processo de alinhamento, que no nosso trabalho será por meio da análise dos ângulos do vetor gradiente, que será melhor detalhada no capítulo seguinte.

Para determinar o sentido de varredura, existem alguns caminhos que podem ser seguidos. Um deles baseia-se no fato da existência de palavras-código especiais de começo e fim, presentes no início e no final de alguns símbolos.

Um outro caminho é dado pela análise do dígito verificador, que, ao ser determinado, pode certificar se o símbolo foi lido do fim ao início.

Para um código binário (mencionado anteriormente) por exemplo, com três módulos compostos por uma barra e um espaço, teremos apenas as opções mostradas na figura 2.2 .

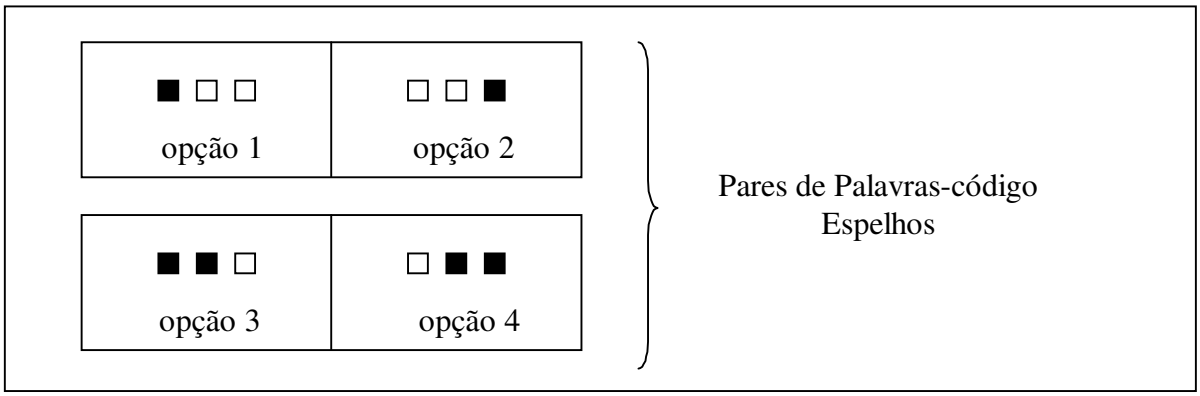


Figura 2.2

Dessa forma, se as palavras de começo e fim em um símbolo forem um dos pares acima, não será possível determinar qual o sentido de varredura.

O Código UPC usa padrões de começo e fim que não estão presentes na sua tabela de codificação, ao contrário do Código 39 que usa uma única palavra-código de começo e fim que está presente na sua tabela de codificação.

2.3 Universal Product Code – UPC

Considerado a primeira simbologia robusta adotada comercialmente, o UPC foi desenvolvido por George J. Laurer. Em abril de 1973, a indústria americana de mercadorias e produtos estabeleceu formalmente o UPC como o Código de Barras padrão de

identificação para diversos produtos. Mais tarde, em dezembro de 1976, uma outra simbologia semelhante ao UPC foi adotada devido a interesses de outros países, sendo denominada EAN (European Article Numbering). O "Uniform Code Council[2.2] " (UCC) regulamenta a adoção do UPC em produtos comercializados ou não.

Atualmente existem cinco versões do UPC:

UPC-A versão mais adotada comercialmente e que será descrita e analisada neste trabalho;

UPC-E segunda versão mais utilizada, com supressão de dígitos zeros, e destinada a produtos com espaço útil de impressão reduzido;

UPC-B versão desenvolvida para a indústria farmacêutica, composta de 11 dígitos e sem dígito de checagem de erro;

UPC-C composta de 12 dígitos, esta versão é destinada a viabilizar compatibilidades de leituras entre indústrias;

UPC-D versão de comprimento variável, composto de no mínimo 12 dígitos.

2.3.1 Definição

Como dito anteriormente, neste trabalho abordaremos a versão A do código UPC, doravante denominada apenas UPC, a fim de simplificação conceitual.

Cada palavra-código desta simbologia é formada por duas barras e dois espaçamentos, ocupando um total de sete módulos.

O UPC é uma simbologia numérica (codifica apenas dígitos decimais) que utiliza ***Codificação por Análise de Reflexão***.

A largura mais estreita é normalizada[2.3], tornando-se valor padrão de um módulo. Vale ressaltar que, para cada código impresso, um valor específico de medida de um módulo

[2.2] Uniform Code Council, Inc., Princeton Pike Corporate Center, 1009 Lenox Dr.,Suite 202, Lawrenceville, New Jersey 08648

[2.3] *Normalizar:* adoção de um valor de medida como referência unitária para mensurar outros valores.

é adotado, uma vez que os Códigos de Barras são considerados "**sistemas abertos**" de codificação, ou seja, suas dimensões são variáveis.

2.3.2 Estrutura

Composto de 12 dígitos decimais, o Código UPC possui uma estrutura caracterizada pela presença de três padrões que subdividem o símbolo em dois campos (esquerdo e direito). Os padrões laterais são definidos pela seqüência 101 e o padrão de guia central pela seqüência 01010. A figura 2.3 ilustra com detalhes a estrutura desta simbologia.

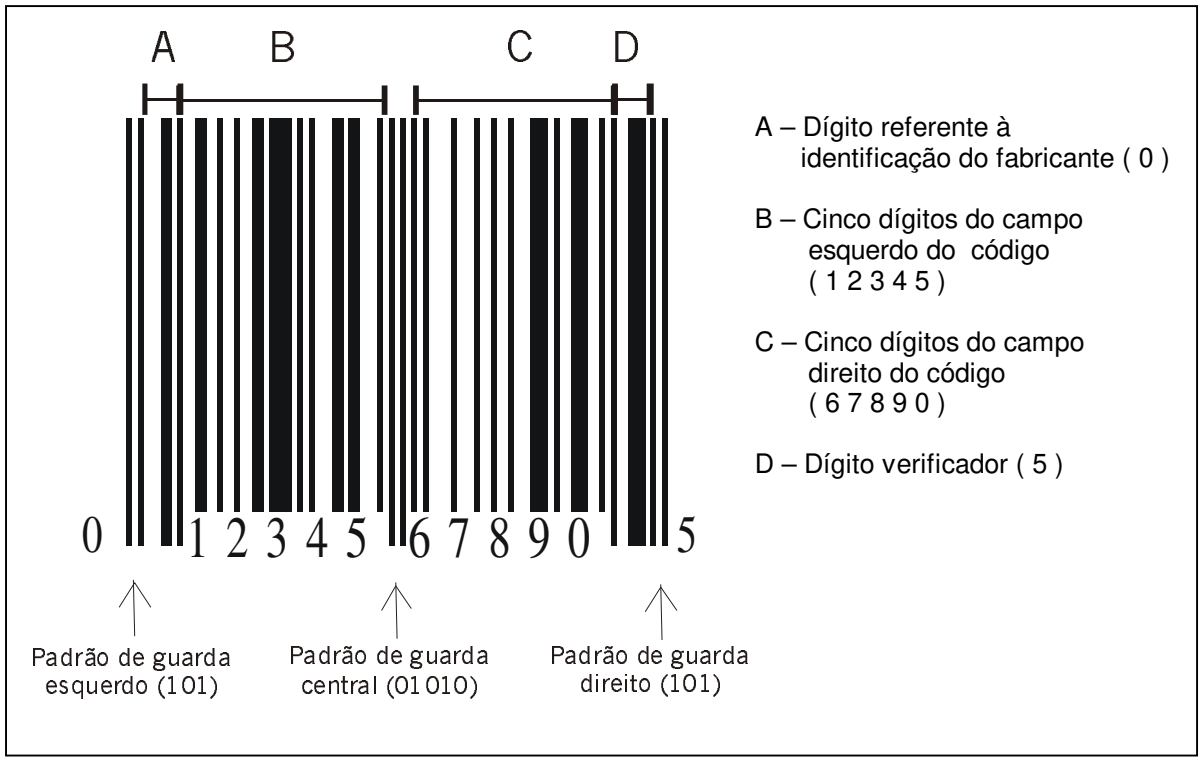


Figura 2.3

Cada dígito decimal pode ser expresso por meio de duas palavras-código, dependendo do posicionamento do dígito no símbolo (representação gráfica). Se um dígito estiver no lado esquerdo do padrão de guia central, a palavra-código que o representa terá

paridade ímpar, ou seja, a soma dos módulos ocupados pelas barras resulta em um número ímpar; caso o dígito esteja localizado no lado direito, a palavra-código terá paridade par (soma dos módulos das barras resulta em número par). A Tabela I mostra o conjunto de caracteres do código UPC e suas representações.

Uma vez que é necessário o reconhecimento do sentido de varredura do leitor ótico, não é possível incluir palavras-código que sejam espelho uma da outra, ou seja, opostamente idênticas, pois isso geraria inconsistências na decodificação.

Pela Tabela I, verificamos que as palavras-código presentes no campo esquerdo do símbolo iniciam com um espaçamento (bit 0), enquanto que aquelas presentes no campo direito iniciam com uma barra (bit 1). Esta característica, juntamente com a informação de paridade e o padrão de guia central são utilizados para a determinação do sentido de varredura, a fim de resultar em uma decodificação correta e válida. As larguras dos elementos são definidas pelos valores c_1,c_2,c_3 e c_4 na tabela. Entretanto, o *efeito de espalhamento de tinta* dificulta a determinação correta destas medidas de larguras. Assim, a forma mais utilizada consiste na medida de distância entre duas barras (t_1) e entre dois espaçamentos (t_2), que não são vulneráveis ao efeito de espalhamento. Por definição: t_1(ímpar)= $c_3 + c_4$; t_1(par)= $c_1 + c_2$ e t_2=$c_2 + c_3$.

Especificação UPC [8]					
Dígito	Esquerda (ÍMPAR)	Direita (PAR)	Padrão de Largura c_1,c_2,c_3,c_4	*Distâncias* (ímpar) t_1,t_2	(par) t_1,t_2
0	0001101	1110010	3,2,1,1	2,3	5,3
1	0011001	1100110	2,2,2,1	3,4	4,4
2	0010011	1101100	2,1,2,2	4,3	3,3
3	0111101	1000010	1,4,1,1	2,5	5,5
4	0100011	1011100	1,1,3,2	5,4	2,4
5	0110001	1001110	1,2,3,1	4,5	3,5
6	0101111	1010000	1,1,1,4	5,2	2,2
7	0111011	1000100	1,3,1,2	3,4	4,4
8	0110111	1001000	1,2,1,3	4,3	3,3
9	0001011	1110100	3,1,1,2	3,2	4,2

Tabela I

2.3.3 Interpretação

No uso comercial, o primeiro dígito presente no código UPC é um dígito de sistema, destinado a compatibilização dos códigos numéricos existentes antes do UPC, podendo tomar os seguintes valores:

0 – códigos UPC em geral;

2 – para produtos de peso variável;

3 – código americano de indústrias farmacêuticas;

4 – codificação personalizada de uso interno em estabelecimentos comerciais;

5 – codificação de cupons.

Do segundo ao sexto dígito temos a identidade da empresa detentora do produto. Cada empresa possui uma identificação própria. A seqüência composta do sétimo ao décimo primeiro dígito identifica o produto de cada fabricante. Cada item de uma determinada empresa possui um código específico. Por fim, o último dígito ("***dígito verificador***"), verifica a validade do código,**"** e será melhor explicado no item seguinte.

2.3.4 Decodificação

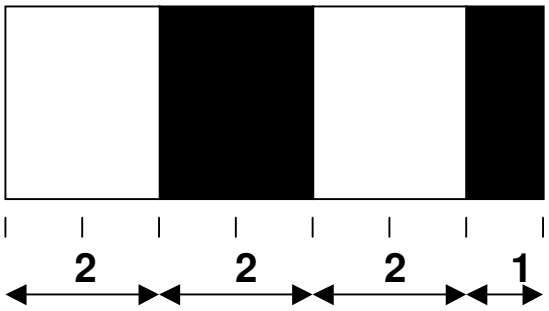
No momento da decodificação de um Código de Barras, um efeito comum de *espalhamento de tinta* provocado durante o processo de impressão do Código de Barras resulta numa problemática que pode ser um fator crítico dependendo da intensidade deste efeito. Se o espalhamento verificado não for de grande intensidade, Códigos de Barras binários (Código 39, por exemplo) poderão ser decodificados de forma satisfatória, uma vez que neste tipo e simbologia existem apenas barras largas e estreitas. As larguras impressas diferem entre si na proporção de 1:3 módulos, existindo dessa forma, uma tolerância com relação a um possível espalhamento de tinta.

Ao contrário dos Códigos Binários, os códigos UPC/EAN são extremamente vulneráveis ao *efeito de espalhamento de tinta*, uma vez que suas palavras-código possuem

elementos com larguras diferindo-se muitas vezes por apenas um módulo. Este fato faz com que alguns dígitos sejam decodificados de forma incorreta. Existem pares especiais de dígitos que são confundidos entre si, pois possuem características semelhantes que favorecem a uma decodificação incorreta, embora válida. Esses pares são (1 e 7) e (2 e 8). A Figura 2.4 mostra uma exemplificação do efeito de espalhamento entre esses pares especiais de dígitos.

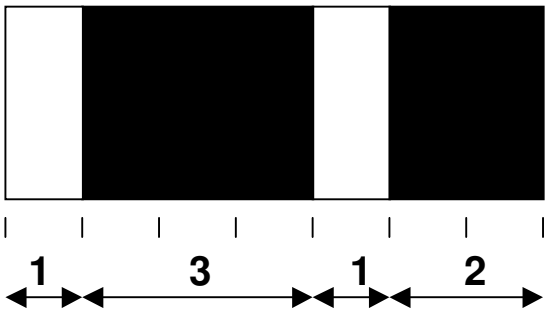
Conforme ilustra a Figura 2.4, assim como os dígitos 1 e 7 podem ser confundidos devido ao problema de *espalhamento de tinta*, os dígitos 2 e 8 também são suscetíveis a esse tipo de equívoco. Dessa forma, uma técnica destinada a solucionar este tipo de problema consiste na adoção de outros parâmetros além dos coeficientes c1,c2, c3 e c4. Esta técnica baseia-se nas distâncias entre duas barras (t1) e espaçamentos (t2) subseqüentes entre si.

O dígito verificador é destinado a impedir este tipo de decodificação incorreta porém válida.



Representação em grande escala do dígito **1** (para o campo esquerdo do símbolo), corretamente expresso.

1 ➔ (2 2 2 1)



Resultado do efeito de espalhamento de tinta no sentido da direita para a esquerda, ocorrido na representação acima.
Este efeito resultou em uma palavra-código válida (dígito **7**), embora incorretamente interpretada.

7 ➔ (1 3 1 2)

Figura 2.4

2.3.4.1 Dígito de verificação

Sendo calculado mediante os onze primeiros dígitos, o dígito de checagem encontra-se no final do código UPC. Como os onze primeiros dígitos são estabelecidos de forma variada (sob as normas estabelecidas pelo UCC), o dígito verificador é uma forma de verificar se realmente o leitor identificou de forma correta esses dígitos.

Para o cálculo do dígito verificador, o algoritmo a seguir deve ser usado:

1. Enumerar os dígitos no sentido da direita para a esquerda. O dígito verificador tomará a posição "1";
2. Somar os valores dos dígitos presentes nas posições pares;
3. Multiplicar o valor da soma encontrada no passo anterior por 3;
4. Somar os valores dos dígitos presentes nas posições ímpares, iniciando-se pelo dígito presente na posição 3;
5. Somar os valores encontrados nos passos 3 e 4;
6. Calcular o menor valor que somado ao resultado do passo 5 resulta em um número múltiplo de 10. Este valor será o valor do dígito verificador.

Como exemplo de cálculo do dígito verificador, consideremos o símbolo presente na Figura 2.3.

Passo 1:

1º = Dígito verificador que a princípio é desconhecido;
2º = Dígito 0;
3º = Dígito 9;
4º = Dígito 8;
5º = Dígito 7;
6º = Dígito 6;
7º = Dígito 5;

8º = Dígito 4;

9º = Dígito 3;

10º = Dígito 2;

11º = Dígito 1;

12º = Dígito 0;

Passo 2:

Soma dos dígitos nas posições pares: 0 + 8 + 6 + 4 + 2 + 0 = **20**

Passo 3:

Multiplicando o valor anterior por 3: **20 ×** 3 **= 60**

Passo 4:

Soma dos dígitos nas posições ímpares (iniciando-se pelo terceiro dígito):

9 + 7 + 5 + 3 + 1 = **25**

Passo 5:

Soma dos valores encontrados nos passos 3 e 4: 60 + 25 = **85**

Passo 6:

O menor número que, somado resultado do passo anterior, resulta em um número múltiplo de 10 é **5**, ou seja: 85 + **5** = 100. ➔ (Dígito verificador: **5**)

2.4 European Article Numbering - EAN

Devido ao grande sucesso alcançado pelo código UPC nos Estados Unidos, surgiu nos países europeus o interesse em desenvolver um outro sistema de Códigos de Barras que mais tarde seria denominado European Article Numbering (EAN). Em 1974 uma comissão de 12 países foi formada para estudar e desenvolver este novo sistema de codificação destinado a produtos industriais e comerciais. A adoção desta nova simbologia iniciou em

dezembro de 1976, e em 1977 foi criada a Associação Européia de Numeração de Artigos (European Article Numbering Association) responsável pelas diretrizes ligadas à implantação da simbologia. Mais tarde, em 1981, a EAN tornou-se a Associação Internacional de Numeração de Artigos, sediada em Bruxelas, entretanto a sigla EAN foi conservada.

2.4.1 Estrutura

Os códigos UPC e EAN possuem a mesma estrutura de codificação, sendo compatíveis entre si, embora sejam utilizados em regiões distintas no mundo. Basicamente, estes códigos diferem entre si pelo número de dígitos. Enquanto o UPC possui 12 dígitos de codificação, o EAN possui um dígito a mais, entretanto o número de barras presentes em ambos é o mesmo. Na realidade, o décimo terceiro dígito adicional do código EAN não é apresentado na forma de padrão de barras e espaços no símbolo; durante o processo de decodificação, ele é indiretamente calculado com o auxílio de uma Tabela Auxiliar de Paridade (Figura 2.5) e dos seis primeiros conjuntos de padrões apresentados, correspondentes aos seis primeiros dígitos codificados do campo esquerdo, que seguem paridades variadas, regidas pela Tabela Auxiliar de Paridade. Esta tabela determina a utilização de outras duas tabelas de padrões (**A** ou **B**) para cada dígito do campo esquerdo. Os dígitos do campo direito seguem a mesma tabela do código UPC, definida como tabela **C**.

Dessa forma, podemos dizer que o código UPC é uma versão do código EAN, pois através da Tabela Auxiliar de Paridade, vemos que, quando o décimo terceiro dígito é zero, a paridade de todos os dígitos seguem a tabela do código UPC (veja Tabela I), e assim o código EAN apresenta-se como UPC.

Conforme sejam as larguras das barras e espaços referentes aos seis primeiros dígitos do símbolo, uma linha da tabela adicional será escolhida, determinado o valor do décimo terceiro dígito, que é utilizado no cálculo do dígito verificador. A figura 2.6 ilustra o

processo de determinação do décimo terceiro dígito, que visualmente é impresso como o primeiro dígito do código EAN.

Observando o exemplo da Figura 2.6, vemos que neste caso, o 13º dígito é nove, e portanto a seqüência de paridade dos seis primeiros dígitos (788534) é dada pela alternância das tabelas **A**,**B**,**B**,**A**,**B**,**A**, conforme a última linha da Tabela auxiliar de Paridade. Dessa forma, os dígitos 7,5 e 4 serão representados na forma de padrões conforme a Tabela **A**, enquanto que os dois dígitos 8 e o dígito 3 serão representados mediante padrões presentes na Tabela **B**.

De fato, no momento de geração de um código EAN, compõe-se o dado com 12 dígitos (seis dígitos do campo esquerdo e seis do campo direito) e calcula-se o décimo terceiro dígito (também denominado dígito adicional) baseado nos doze dígitos iniciais.

2.4.2 Interpretação

Como o código EAN é destinado à identificação de diversos produtos em diversas partes do mundo, foi estabelecida uma identificação através dos 3 ou 2 primeiros dígitos do código, sendo denominada "bandeira". Através desta identificação, cada país que utiliza a identificação de produtos pelo Código de Barras EAN possui um órgão responsável pelo cadastramento de cada produto de uma determinada empresa/fabricante. Para o Brasil, a representação de todos os produtos cadastrados inicia com os dígitos 789 e o órgão responsável pelo registro dos produtos é a ABAC (Associação Brasileira de Automação Comercial). O Apêndice A mostra a relação das bandeiras de diversos países que adotam a representação do código EAN em seus produtos, assim como os casos extras de identificação.

Os dígitos do campo esquerdo que não fazem parte da bandeira do símbolo identificam a empresa detentora do produto, e os dígitos do campo direito são atribuídos à identificação do produto pertencente à empresa.

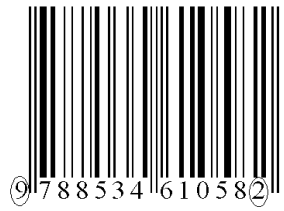
Existem dois tipos de identificações que adotam o padrão EAN e largamente utilizados atualmente: o ISSN (International Standard Serial Number) destinado a periódicos

Tabela Auxiliar de Paridade						
13º Dígito	**Seqüência de Tabelas Utilizadas para cada padrão do campo esquerdo**					
	1º padrão	**2º padrão**	**3º padrão**	**4º padrão**	**5º padrão**	**6º padrão**
0	***A***	***A***	***A***	***A***	***A***	***A***
1	**A**	**A**	**B**	**A**	**B**	**B**
2	**A**	**A**	**B**	**B**	**A**	**B**
3	**A**	**A**	**B**	**B**	**B**	**A**
4	**A**	**B**	**A**	**A**	**B**	**B**
5	**A**	**B**	**B**	**A**	**A**	**B**
6	**A**	**B**	**B**	**B**	**A**	**A**
7	**A**	**B**	**A**	**B**	**A**	**B**
8	**A**	**B**	**A**	**B**	**B**	**A**
9	**A**	**B**	**B**	**A**	**B**	**A**

Código UPC

Dígito	***Campo esquerdo***		***Campo Direito***
	TABELA A	**TABELA B**	**TABELA C**
0	0001101	0100111	1110010
1	0011001	0110011	1100110
2	0010011	0011011	1101100
3	0111101	0100001	1000010
4	0100011	0011101	1011100
5	0110001	0111001	1001110
6	0101111	0000101	1010000
7	0111011	0010001	1000100
8	0110111	0001001	1001000
9	0001011	0010111	1110100

Figura 2.5 - Tabelas do código EAN



Padrões:

0111011 0001009 0001009 0110001 0100001 0100011

7(**A**) 8(**B**) 8(**B**) 5(**A**) 3(**B**) 4(**A**)

Seqüência de Tabelas para os 6 primeiros dígitos:

A B B A B A ⇨ 13 º dígito = 9

Figura 2.6 – Cálculo do 13º dígito para um código EAN

(revistas, jornais, anuários etc.) e o ISBN (International Standard Book Number) destinado à identificação de livros. A representação do ISBN em EAN é denominada Bookland EAN.

Para converter o código ISBN para o Bookland EAN, inicialmente coloca-se o prefixo (bandeira) 978. Em seguida são colocados os nove primeiros dígitos do código ISBN, e por fim o dígito de verificação da simbologia EAN. O dígito de verificação presente no código ISBN é suprimido. Na maioria dos casos, os dígitos de verificação do ISBN e EAN são distintos, mas eventualmente podem coincidir.

Atualmente, o Bookland EAN é composto por dois Códigos de Barras dispostos lado a lado. O código maior, no lado esquerdo é o EAN derivado do ISBN. O menor, situado à direita é o conjunto de padrões que representam 5 dígitos onde diversas informações podem ser codificadas. Freqüentemente, estes dígitos representam o preço da publicação. A Figura 2.7 mostra a representação de um código ISBN sob a simbologia EAN.

Figura 2.7 – Estrutura da simbologia Bookland EAN

2.4.3 Dígito Verificador

O processo para o cálculo do dígito verificador no código EAN é idêntico àquele mostrado para o código UPC. A única diferença consiste no acréscimo de um dígito (13º dígito), que no código UPC poderia ser considerado zero. Dessa forma, o algoritmo

explicado na seção 2.4.4.1 é válido para o código EAN. A Figura 2.8 ilustra um exemplo de cálculo do dígito verificador.

POSIÇÃO	13º	12º	11º	10º	9º	8º	7º	6º	5º	4º	3º	2º	1º	
CÓDIGO	4	0	0	9	9	9	3	1	3	4	1	3	**?**	
PARES		0	+	9	+	9	+	1	+	4	+	3	=	26
ÍMPARES	4	+	0	+	9	+	3	+	3	+	1		=	20

$26 \times 3 = 78$

$78 + 20 = 98$ ➔ $98 + \mathbf{2} = 100$ ➔ Dígito Verificador=2

Figura 2.7 – Cálculo do Dígito Verificador para o código EAN

2.5 Code 39 (Código 3 de 9)

Esta simbologia surgiu em 1975, adotada pelo Departamento de Defesa dos Estados Unidos, que oficializou o Código 39 para todos os suprimentos, mediante a Norma de Padronização Militar 1189, em 1982. Atualmente esta simbologia é adotada em diversos países nas áreas de serviços hospitalares, transportes aéreos, bibliotecas, entre outras.

2.5.1 Definição

O Código 39 é um código binário alfanumérico capaz de codificar 44 caracteres (26 letras, 10 algarismos numéricos, 1 caractere de espaçamento, e 7 caracteres de símbolos). Um código binário possui elementos com apenas duas larguras possíveis para expressar os valores lógicos “0” e “1”, ou seja, para uma barra ou espaçamento, somente são possíveis espessuras largas (valor lógico “1”) ou estreitas (valor lógico “0”).

2.5.2 Estrutura

O Código 39 é constituído por palavras-código que possuem 9 elementos (barras e espaçamentos) dos quais 3 deles são largos. Esta estrutura justifica o nome da simbologia.

Cada palavra-código que expressa um caractere possui cinco barras e quatro espaçamentos. Existe um espaçamento intercaractere na concatenação das palavras –código, caracterizando esta simbologia como "*discreta*". A largura do espaçamento intercaractere deve ser aproximadamente idêntica à largura de um elemento estreito. A princípio, não há um tamanho fixo para a representação do Código 39, ou seja, é possível codificar quantos caracteres forem necessários.

Esta simbologia usa o caractere asterisco (*) como padrão de início e fim do símbolo impresso. A Tabela II mostra o conjunto de padrões para cada um dos 44 caracteres possíveis de codificação e a Figura 2.8 ilustra a estrutura de um Código 39.

Caractere	Padrão	Barras	Espaços	Caractere	Padrão	Barras	Espaços
1		10001	0100	M		11000	0001
2		01001	0100	N		00101	0001
3		11000	0100	O		10100	0001
4		00101	0100	P		01100	0001
5		10100	0100	Q		00011	0001
6		01100	0100	R		10010	0001
7		00011	0100	S		01010	0001
8		10010	0100	T		00110	0001
9		01010	0100	U		10001	1000
0		00110	0100	V		01001	1000
A		10001	0010	W		11000	1000
B		01001	0010	X		00101	1000
C		11000	0010	Y		10100	1000
D		00101	0010	Z		01100	1000
E		10100	0010	-		00011	1000
F		01100	0010	.		10010	1000
G		00011	0010	ESPAÇO		01010	1000
H		10010	0010	***		00110	1000
I		01010	0010	$		00000	1110
J		00110	0010	/		00000	1101
K		10001	0001	+		00000	1011
L		01001	0001	%		00000	0111

Tabela II

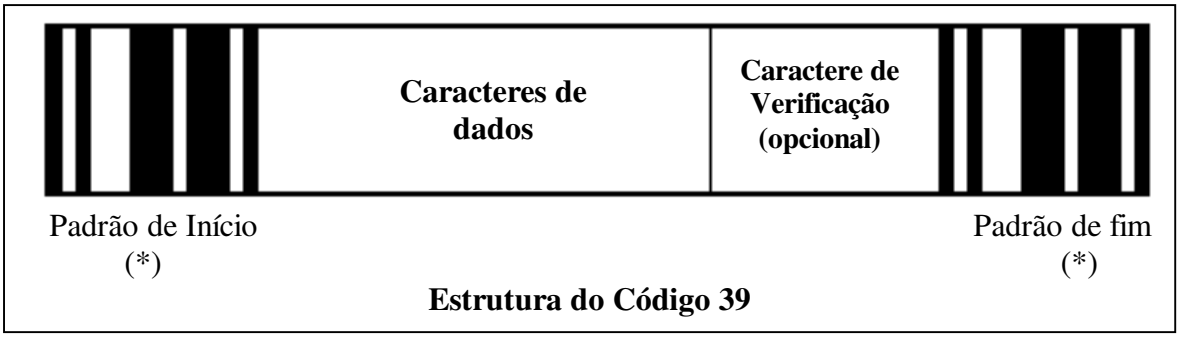


Figura 2.8 – Estrutura do Código 39

Como ilustrado na Figura 2.8, após os padrões referentes à codificação dos dados, existe a possibilidade da adoção de um padrão destinado a um caractere de verificação. Além disso, conforme mencionado no Capítulo 1, existe uma margem de silêncio presente nas laterais do símbolo, que devem ser obedecidas para evitar erros de leitura pelo dispositivo ótico.

2.5.3 Decodificação

A fim de habilitar o leitor ótico a distinguir elementos largos e estreitos, é necessário adotar uma razão de proporção. Mediante a resolução adotada no processo de impressão, a largura de um elemento largo deverá ser no mínimo duas vezes a largura de um elemento estreito. A proporção mais adequada é 3:1. Deve-se seguir o critério de uniformidade, ou seja, os elementos do mesmo tipo devem possuir a mesma medida. Dessa forma, a largura de uma barra estreita deve ser idêntica à largura de um espaçamento estreito, e vice-versa.

No momento da leitura, os padrões de início e fim do símbolo, definidos pelo caractere asterisco (*), indicam o sentido da varredura do leitor ótico, uma vez que as leituras deste padrão em sentidos opostos geram resultados distintos. As Figuras 2.9.a e 2.9.b ilustram como o sentido de varredura é determinado através dos padrões laterais.

Figura 2.9 b

Figura 2.9 a

A análise das Figuras 2.9.a e 2.9 b mostra como o leitor ótico pode iniciar a leitura

do símbolo impresso, e como o padrão asterisco é interpretado conforme seja o sentido da varredura. Se o padrão de início de leitura resultar em uma seqüência 0 1 0 0 1 0 1 0 0, o leitor ótico estará realizando uma leitura no sentido da esquerda para a direita do código, caso a seqüência seja 0 0 1 0 1 0 0 1 0, o sentido da varredura será da direita para a esquerda.

2.5.4 Dígito de verificação

O Código 39 é uma simbologia que apresenta uma estrutura favorável a uma auto-checagem, pois durante o processo de leitura, é possível avaliar a quantidade de elementos, assim como a quantidade necessária de elementos largos e estreitos para que o símbolo seja consistente. Dessa forma, para a validação do símbolo é necessário que:

1. A quantidade total de elementos (barras e espaços) encontrados pelo leitor seja um múltiplo de 9 ;
2. A quantidade total de elementos largos seja um múltiplo de 3 ;
3. A quantidade total de elementos estreitos seja um múltiplo de 6 ;
4. A quantidade total de barras encontradas seja um múltiplo de 5 ;
5. A quantidade total de espaços encontrados seja um múltiplo de 4 ;
6. E que as somas das quantidades encontradas nos itens (2 e 3) e (4 e 5) sejam idênticas e ambas resultem na quantidade encontrada no item 1.

Embora a validação do símbolo impresso possa ser analisada por meio dos passos citados acima, é possível a adoção de um caractere opcional de verificação, que certifica se a decodificação foi realizada sem erros.

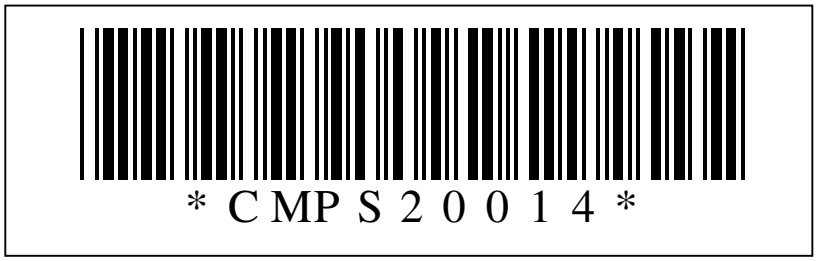
O cálculo do caractere de verificação é denominado verificação módulo 43, e segue o seguinte algoritmo:

1. Para cada caractere presente na tabela padrão adota-se um valor correspondente ao seu posicionamento. Os valores variam de zero até 42 em ordem ascendente de posicionamento. O caractere correspondente ao asterisco,

por definição, não possui valor, dessa forma, o caractere referente ao ‘espaço’ possui valor “38” e o caractere correspondente ao símbolo ‘ $ ’ possui valor “39”.

2. Em seguida, soma-se os valores de todos os caracteres decodificados antes do caractere de verificação ;
3. Divide-se o resultado da soma anterior por 43 e verifica-se o resto;
4. Através da tabela padrão, verifica-se qual é o caractere correspondente ao resto da divisão, obtido no passo anterior. Este será o caractere verificador.

Vejamos como exemplo, o mesmo símbolo das figuras 2.9.a e 2.9.b., presente na Figura 2.10 (a seguir), porém com a inserção do dígito verificador:



Vemos que o contexto codificado foi:

C M P S 2 0 0 1

Somando os valores dos caracteres, teremos:

C M P S 2 0 0 1

12 + 22 + 25 + 28 + 2 + 0 + 0 + 1 = **90**

90 ÷ 43 = 2 *resto 4* ➔ O caractere de verificação é o caractere que possui valor “4”, ou seja, pela tabela, quem possui valor 4 é o algarismo numérico **4**.

Capítulo 3

Processos para Decodificação

Em Automação Industrial, os códigos de barras são analisados e interpretados segundo diversos dispositivos de leitura: desde leitores manuais de feixe pontual aos dispositivos CCD (Charge Coupled Device).

Mediante a sua popularização e vasta aplicação para fins de economia de tempo, o código de barra necessita de um sistema de análise de imagens aprimorado, a fim de permitir uma decodificação diretamente a partir de uma imagem construída pelo dispositivo ótico. Este sistema envolve conceitos de reconhecimento ótico de padrões, classificação de objetos e controle de qualidade.

Para uma otimização de decodificação, é necessária a aplicação de algumas técnicas de processamento de imagens. No presente trabalho, o sistema de análise consiste em alguns conceitos de morfologia matemática e estatística.

3.1 Princípios de Análise

A partir da informação de entrada, que será uma imagem composta de um símbolo em uma posição aleatória, uma análise deve ser feita a fim de se obter uma série de informações contidas nesta imagem.

Independentemente da simbologia alvo de decodificação, uma seqüência de passos deve ser seguida. O passo inicial consiste na aquisição do símbolo pelo dispositivo ótico. Um pré-processamento é aplicado na imagem inicial capturada, a fim de solucionar

problemas de enquadramento do símbolo. A partir daí, a imagem, que inicialmente possui 256 níveis de cinza, é convertida em uma imagem binária (níveis preto e branco).

Após a binarização, conforme a simbologia presente na imagem inicial, é necessária a obtenção de medidas específicas para a classificação dos padrões encontrados na leitura do Código de Barras.

De forma geral, a figura 3.1 ilustra a seqüência seguida pelo algoritmo de decodificação.

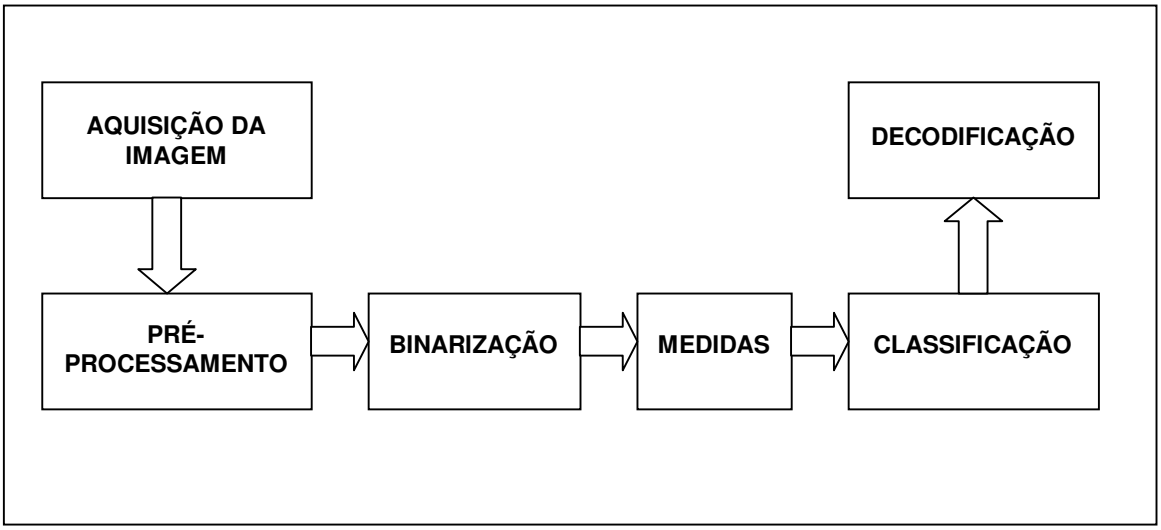


Figura 3.1 – Passos para decodificação

Consideraremos a aquisição da imagem de entrada por meio de um processo de digitalização do símbolo, utilizando um leitor ótico comum de mesa com resolução ótica máxima de 300x600 dpi. A imagem resultante da digitalização do símbolo por este leitor ótico possui 256 níveis de cinza.

A fim de garantir uma decodificação confiável, é necessário adotar uma resolução mínima tolerável, que é dada a partir da largura de elemento mais estreito do símbolo. Na maioria das imagens analisadas, a resolução mínima de um elemento estreito foi de 4 pixels.

É importante verificar se as zonas de silêncio nas laterais do símbolo estão sendo respeitadas, assim como os padrões de início e fim, pois estas partes do símbolo são responsáveis pelo processo de controle da decodificação.

3.1.1 Enquadramento

No momento da captura do símbolo pelo leitor ótico, um fator importante que deve ser levado em consideração é o posicionamento do eixo longitudinal do símbolo, ou seja, o ângulo do eixo longitudinal com a horizontal.

O desalinhamento característico do Código de Barras com a horizontal, no momento da digitalização deve ser determinado e corrigido. Existem algumas técnicas que detectam o ângulo de desalinhamento. No nosso trabalho, utilizaremos a técnica de estimação do ângulo preponderante do vetor gradiente pelas máscaras de Sobel, e será abordada mais adiante.

O ângulo do eixo longitudinal com relação à horizontal deve ser nulo ou muito próximo de zero, a fim de evitar problemas de serrilhados da imagem digitalizada, que podem provocar erros de decodificação. A Figura 3.2 ilustra o processo de enquadramento.

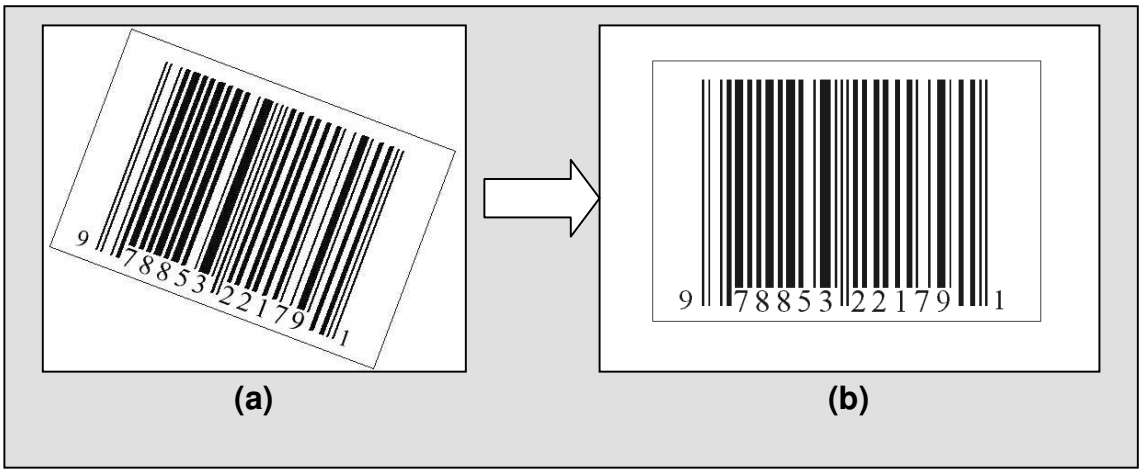


Figura 3.2 – Enquadramento de um símbolo capturado.
(a) Imagem de entrada; (b) Enquadramento almejado

3.1.2 Ilustração de Interferências

A correta decodificação de um Código de Barras está diretamente relacionada à qualidade do sinal gerado pelo leitor ótico. Como existem diversos fatores que influenciam a leitura, o sinal de saída do leitor ótico é caracterizado por uma forma de onda muitas vezes indecifrável, pois esses fatores impedem uma reconstrução perfeita da forma de onda ideal, que deveria ser um conjunto de pulsos retangulares, conforme ilustra a Figura 3.3.

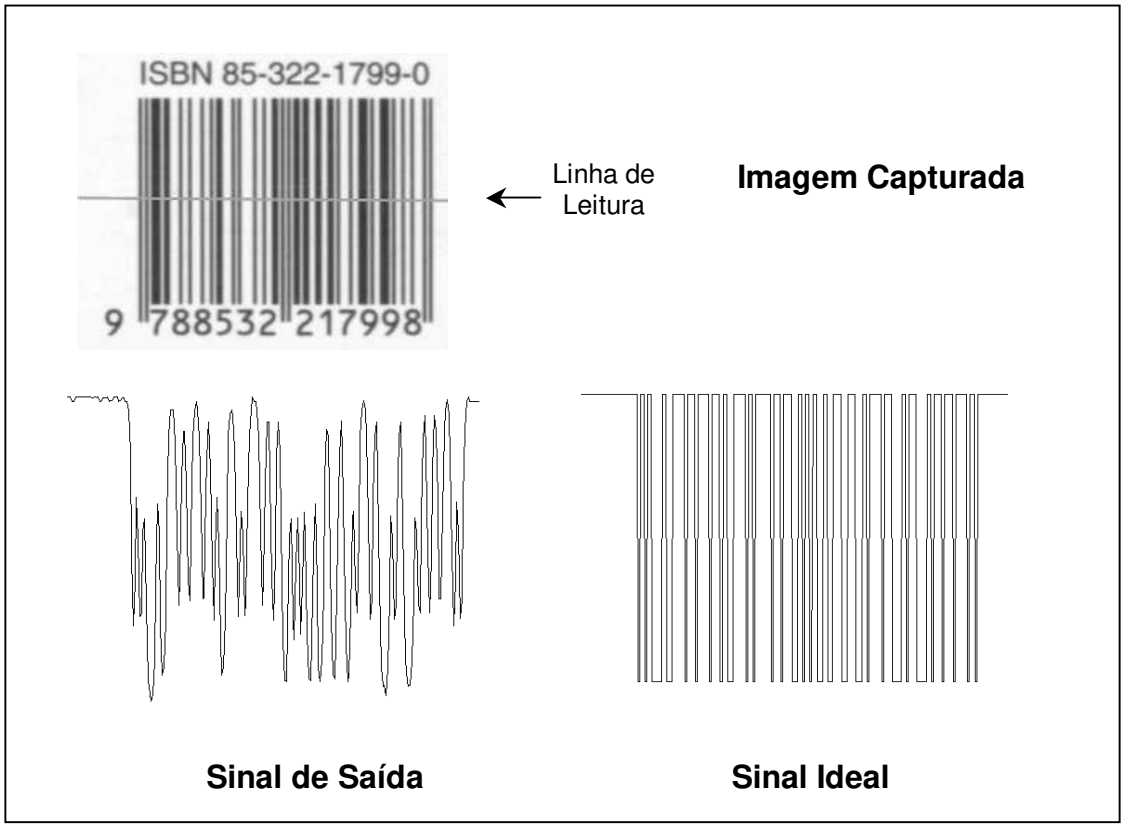


Figura 3.3

Observando o sinal de saída na Figura 3.3, vemos a presença acentuada de ruído ao longo do contorno da curva de onda no sinal de saída. Dessa forma, é necessário aplicar uma técnica eficiente de detecção de barras e espaços para interpretar de forma correta a forma de onda e decodificar a informação contida no símbolo.

O foco de nosso trabalho é a classificação das barras e espaços por meio de uma *Limiarização (Threshold Global)* eficiente, que deverá resultar em um símbolo representado através de uma imagem binária (apenas dois níveis de cores: preto e branco) com uma quantidade de barras plausível ao tipo de simbologia presente na imagem.

3.1.3. Ruídos e distorções

Além do ruído, os padrões da simbologia aplicada são influenciados pela presença de distorções na superfície de impressão ou até mesmo no equipamento de leitura.

É importante salientar a diferença entre *ruído* e *distorção*. Ao varrer o código mais de uma vez, os efeitos de distorção irão se repetir, embora os efeitos de ruído provavelmente serão distintos. Os efeitos de ruído significativos para a decodificação estão presentes entre barras e espaços.

Espalhamento de Tinta

Conforme mencionado no capítulo 2, o espalhamento de tinta é o efeito de distorção mais comum nos símbolos impressos.

Existem dois processos de impressão dos Códigos de Barras: um deles é realizado conforme critérios estabelecidos pelo usuário e um outro sob critérios de qualidade pré-definidos. Geralmente, no processo de impressão sob critérios do usuário, a qualidade da impressão é baixa, e portanto o efeito de espalhamento de tinta é bastante verificado. Entretanto, na impressão sob critérios pré-definidos, o efeito de espalhamento de tinta praticamente não existe, uma vez que a impressão é de alta qualidade, sendo exigido uma precisão de 0,01 polegada, praticamente imperceptível ao olho humano. O código UPC é o mais vulnerável ao efeito de espalhamento de tinta, conforme já mencionamos.

3.1.4 Técnicas de Binarização Avaliadas

A detecção de barras e espaços está sujeita a erros conforme a metodologia adotada para esse fim, a variação de contraste no momento da varredura, ou mesmo a iluminação do ambiente. Com isso, deve-se monitorar mudanças de intensidade de luz refletida durante a varredura, a fim de se estabelecer uma uniformidade das condições de leitura. Ou seja, é necessário um sistema de controle de ganho automático de luminosidade no dispositivo de leitura.

As técnicas de detecção de barras e espaços consistem em um processo de binarização, que transforma a imagem de 256 níveis adquirida pelo leitor ótico em uma nova imagem de 2 níveis de cinza (imagem binária: preto e branco), destacando as transições entre barras e espaços, necessárias para a decodificação. Este processo pode ser realizado através de alguns métodos, tais como *Limiarização* (Threshold global) e *Análise do vetor Gradiente ou pela Transição por zeros da segunda derivada.*

Em nosso trabalho, avaliamos três métodos de binarização: Limiarização de Otsu, Classificação por "*Watershed"* e Detecção de passagens por zeros .

3.1.4.1 Threshold (Limiarização)

O processo de *Limiarização* será a primeira, e talvez a mais crítica técnica de processamento de imagem aplicada antes do processo de decodificação do símbolo impresso.

Limiarização é um método de segmentação de imagens em níveis de cinza que extrai do plano de fundo de uma imagem, objetos de interesse, partindo-se do princípio que o objeto e o plano de fundo da imagem possam ser diferenciados entre si pelos seus níveis de cinza. Através da escolha de um valor de limiar "k^*", situado entre os valores de cinza preponderantes do objeto e do plano de fundo, os valores de níveis de cinza originais da

imagem podem ser transformados em uma forma binária, onde os pontos associados ao objeto assumem valor "um", e aqueles associados ao plano de fundo assumem valor "zero".

A determinação de um valor para a aplicação de uma *Limiarização* na imagem de um Código de Barras requer critérios bastante flexíveis, pois existem muitos fatores que influenciam de forma diversificada o sinal de saída no leitor ótico. O problema de espalhamento de tinta é um desses fatores e que dificulta muito a eficiência da escolha do valor de limiar, pois a intensidade do espalhamento de tinta não é um dado exato.

A fim de se obter um resultado satisfatório desta técnica, é necessário um estudo dos critérios de escolha para este valor, pois o valor "***k****" será responsável pela distinção entre o símbolo do Código de Barras (objeto de interesse) e o plano de fundo. No nosso estudo, a técnica de *Limiarização* adotada foi aquela desenvolvida por Nobuyuki Otsu [9].

Threshold de Otsu

A técnica de limiarização de Otsu é um método automático de segmentação de imagens em níveis de cinza, usando dados estatísticos do histograma da imagem como base para escolha de um valor de limiar eficaz.

Sejam os pixels[3.1] de uma imagem representados por **L** níveis de cinza [1,2,3,...,**L**]. O número de pixels presente na imagem com intensidade de cinza ***i*** é denotado por $\boldsymbol{n_i}$ e o total de pixels presentes na imagem por $\mathbf{N} = n_1 + n_2 + n_3 + \ldots + n_L$. Normaliza-se o histograma, acumulando-o como uma distribuição de probabilidade:

$$p_i = n_i / N \quad , \quad p_i \geq 0 \quad , \quad \sum_{i=1}^{L} p_i = 1 . \tag{3.1}$$

Em seguida, dividimos os pixels em duas classes C_0 e C_1 (plano de fundo e objetos) por meio de um valor de limiar ***k***. A classe C_0 indica os pixels com níveis em [1,2,...,***k***] e C_1 denota os pixels com níveis em [***k***+1, ***k***+2,...**L**]. A probabilidade de ocorrência das classes C_0 e C_1 e suas médias são dadas por:

[3.1] Pixel: Contração de Picture (Pix) Element. Menor elemento de uma imagem gráfica.

$$\omega_0 = \Pr(C_0) = \sum_{i=1}^{k} P_i = \omega(k) \qquad \omega_1 = \Pr(C_1) = \sum_{i=k+1}^{L} P_i = 1 - \omega(k) \qquad (3.2)$$

$$\mu_0 = \sum_{i=1}^{k} i \Pr(i \mid C_0) = \sum_{i=1}^{k} i p_i / \omega_0 = \mu(k) / \omega(k) \qquad (3.3)$$

$$\mu_1 = \sum_{i=k+1}^{L} i \Pr(i \mid C_1) = \sum_{i=k+1}^{L} i p_i / \omega_1 = \frac{\mu_T - \mu(k)}{1 - \omega(k)} \qquad (3.4)$$

Onde os momentos acumulados de ordem "zero" e "um" do histograma (para níveis de cinza até k) são dados respectivamente por:

$$\omega(k) = \sum_{i=1}^{k} p_i \qquad \text{e} \qquad \mu(k) = \sum_{i=1}^{k} i p_i \qquad (3.5)$$

E a média total de níveis na imagem é dada por:

$$\mu_T = \mu(L) = \sum_{i=1}^{L} i p_i \qquad (3.6)$$

Assim: $\omega_0 \mu_0 + \omega_1 \mu_1 = \mu_T$

$$\omega_0 + \omega_1 = 1 \qquad (3.7)$$

As variâncias das classes C_0 e C_1 são dadas por:

$$\sigma_0^2 = \sum_{i=1}^{k} (i - \mu_0)^2 \Pr(i \mid C_0) = \sum_{i=1}^{k} (i - \mu_0)^2 p_i / \omega_0 \qquad (3.8)$$

$$\sigma_1^2 = \sum_{i=k+1}^{L} (i - \mu_1)^2 \Pr(i \mid C_1) = \sum_{i=k+1}^{L} (i - \mu_1)^2 p_i / \omega_1 \qquad (3.9)$$

A partir daí, são estabelecidas medidas indicadoras de separabilidade, ou seja, fatores que impõem critérios para a determinação de um valor de limiar eficaz; dadas por:

$$\lambda = \sigma_B^2 + \sigma_W^2, \qquad \kappa = \sigma_T^2 / \sigma_W^2, \qquad \eta = \sigma_B^2 / \sigma_T^2 \qquad (3.10)$$

Onde:

$$\sigma_W^2 = \omega_0 \sigma_0^2 + \omega_1 \sigma_1^2 \qquad \textit{(Variância das classes)} \qquad (3.11)$$

$$\sigma_B^2 = \omega_0(\mu_0 - \mu_T)^2 + \omega_1(\mu_1 - \mu_T)^2$$
$$= \omega_0\omega_1(\mu_1 - \mu_0)^2 \qquad \textit{(Variância entre as classes)} \qquad (3.12)$$

Dessa forma:

$$\sigma_T^2 = \sum_{i=1}^{L}(i - \mu_T)^2 p_i \qquad \textit{(Variância Total dos níveis)} \qquad (3.13)$$

Adotando η como critério de eficiência da escolha de um valor k como limiar, devemos considerar que o valor otimizado ***(k*)*** será aquele que maximiza η, ou de forma equivalente, maximiza σ_B^2.

Portanto:

$$\eta(k) = \sigma_B^2(k) / \sigma_T^2 \qquad (3.14)$$

$$\sigma_B^2(k) = \frac{[\mu_T\omega(k) - \mu(k)]^2}{\omega(k)[1 - \omega(k)]} \qquad (3.15)$$

Finalmente, o valor de limiar ótimo ***(k*)*** será dado por:

$$\sigma_B^2(k^*) = \max_{1 \le k < L} \sigma_B^2(k)\,. \qquad (3.16)$$

Dentre as três técnicas de binarização, o método de limiarização de Otsu mostrou-se mais robusto nos aspectos de tempo de processamento e qualidade de imagem de saída. A Limiarização de Otsu apresentou uma adaptação eficaz na presença dos ruídos inerentes ao processo de digitalização da imagem.

Na aplicação desta técnica, o cálculo de um fator importante η (3.14), avalia a separabilidade entre as duas classes da imagem (objeto e fundo), ou seja, a bimodalidade do

histograma da imagem original. Com este fator, podemos estimar a eficiência com que o valor de limiar obtido "***k***", discrimina um objeto do plano de fundo na imagem. Um histograma com pouca variação de intensidade de cinza resulta em um fator de bimodalidade baixo, ao contrário de um histograma com níveis de cinza distintos e distribuídos em dois grupos definidos (Figura 3.4).

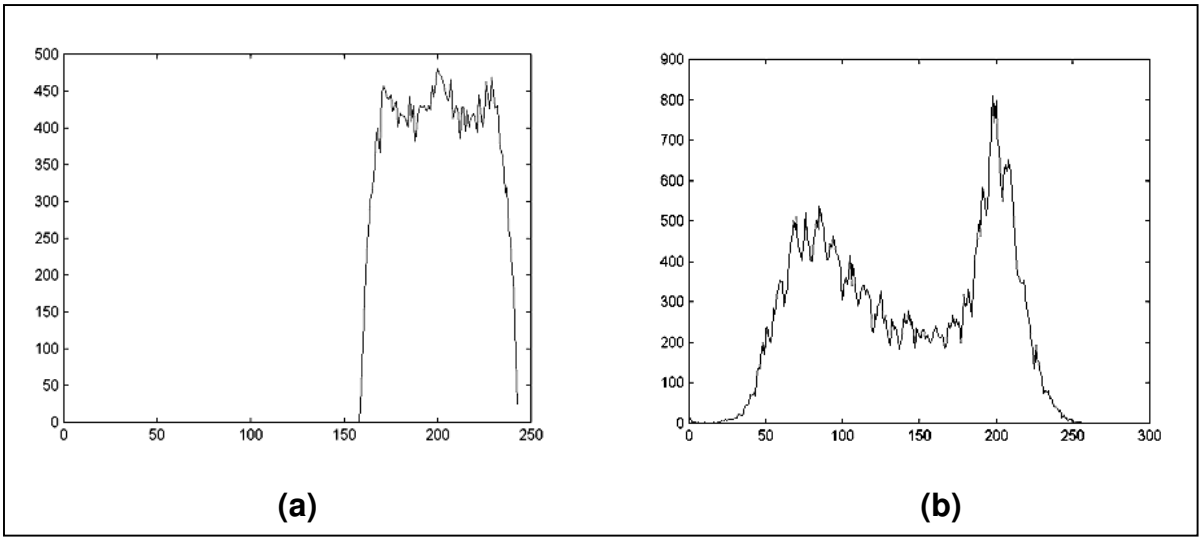


Figura 3.4 – (a) Histograma com bimodalidade baixa ; (b) Histograma com bimodalidade alta

3.1.4.2 Operadores Diferenciais

O método de segmentação de imagens por detecção de bordas consiste na análise de descontinuidades dos níveis de cinza presentes na imagem por meio de operadores diferenciais. Os operadores diferenciais mais usados são o Gradiente e o Laplaciano.

Borda em uma imagem, é caracterizada pelo limite entre duas regiões cujos níveis de cinza possuem propriedades distintas. No nosso estudo, onde imagem é caracterizada pela presença de um símbolo de Código de Barras, as bordas resultantes serão as delimitações das barras e dos caracteres impressos. Significativamente, as bordas de maior interesse na decodificação serão aquelas relacionadas às barras do símbolo. Como a imagem analisada,

do ponto de vista global, possui características relativamente homogêneas, as descontinuidades dos níveis de cinza caracterizam bem o limite entre regiões distintas, quando a imagem capturada é de alta qualidade.

De forma básica, a Figura 3.5 ilustra a técnica de detecção de bordas.

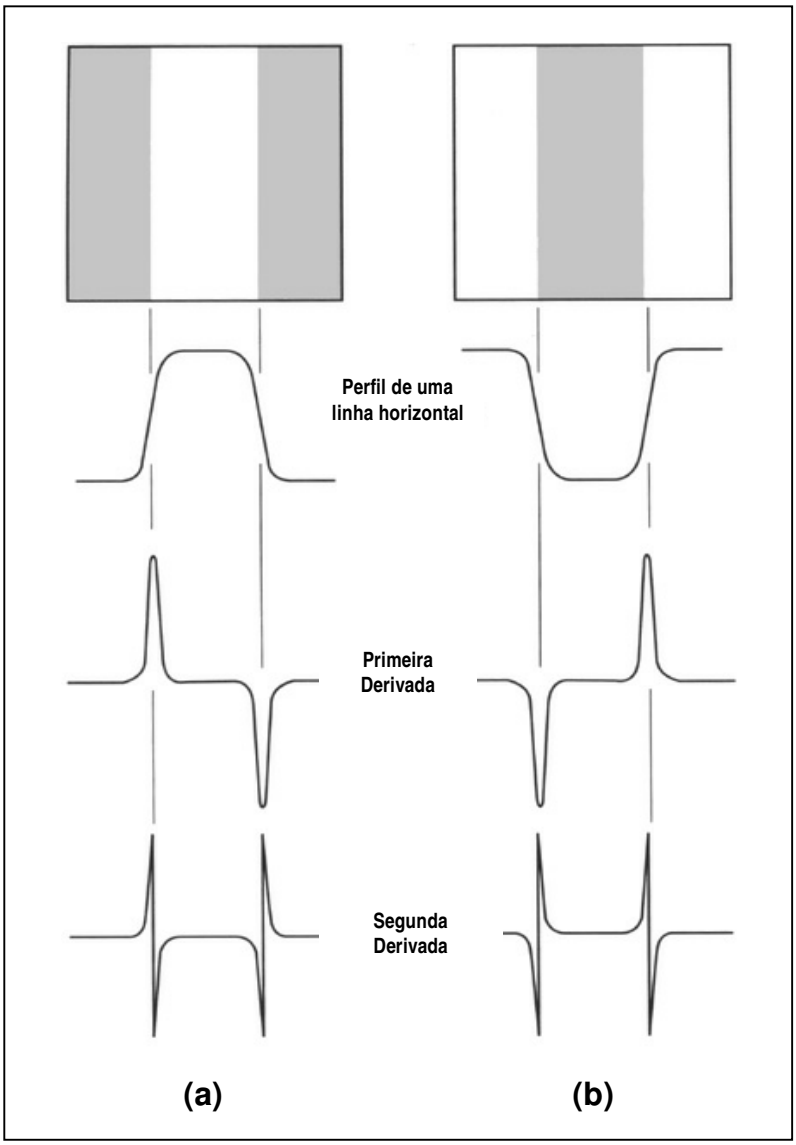


Figura 3.5 - Detecção de Bordas por Operadores Derivativos
(Fonte: [10] - R. C. Gonzalez and R. E. Woods, "Digital Image Processing")

Através dos perfis das imagens na Figura 3.5, observamos uma transição suave nos níveis de cinza, que é uma característica geralmente presente nas imagens digitais, uma vez que o processo de digitalização provoca um leve borramento.

Os extremos da magnitude da primeira derivada são usados na detecção de bordas enquanto que para a segunda derivada, as transições por zeros são os melhores indicadores de bordas.

Gradiente

A primeira derivada em um ponto qualquer de uma imagem é dada pela magnitude do gradiente no ponto considerado.

É através da orientação do vetor gradiente que determinamos a direção da transição mais abrupta de uma função, que no nosso estudo é uma imagem. Dessa forma, as informações que o vetor gradiente oferece em uma imagem são as transições abruptas de níveis de cinza e a distribuição dos seus ângulos de inclinação que será analisada mais adiante.

O vetor gradiente de uma imagem $f(x,y)$ é dado por:

$$\nabla \mathbf{f} = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \dfrac{\partial f}{\partial x} \\ \\ \dfrac{\partial f}{\partial y} \end{bmatrix} \tag{3.17}$$

E sua magnitude é dada por:

$$\nabla f = mag(\nabla \mathbf{f}) = \left[G_x^2 + G_y^2 \right]^{1/2} \tag{3.18}$$

Esta magnitude corresponde à maior taxa de aumento de $f(x,y)$ por unidade de distância na direção de $\nabla \mathbf{f}$.

No cálculo das derivadas parciais G_x e G_x, são utilizadas máscaras denominadas operadores de Sobel, dados por:

-1	-2	-1
0	0	0
1	2	1

Máscara de G_x

-1	0	1
-2	0	2
-1	0	1

Máscara de G_y

Assim, o resultado obtido no cálculo de G_x é dado através da convolução da imagem original $\boldsymbol{f}$ pela máscara de G_x, e aquele obtido no cálculo de G_y é dado pela convolução de f pela máscara de G_y.

Os pontos de mínimos e máximos da primeira derivada caracterizam a presença de bordas, uma vez que nesses pontos ocorrem uma transição acentuada de nível de cinza na imagem.

No caso de uma imagem contendo um símbolo de Código de Barras, o resultado da análise pelo gradiente é dado pela equação 3.18, e ilustrado na Figura 3.6.

Figura 3.6 – (a) Imagem Original $\boldsymbol{f}$; (b) Convolução de $\boldsymbol{f}$ por $\boldsymbol{G_x}$; (c) Convolução de $\boldsymbol{f}$ por $\boldsymbol{Gy}$; (d) Imagem da magnitude do gradiente

Conforme mencionado na seção 3.1.1, a aplicação de um pré-processamento na imagem capturada pelo dispositivo ótico é dada pela rotação por um ângulo que o eixo longitudinal do símbolo faz com a horizontal. Este ângulo é calculado baseado na direção do vetor gradiente. Para um ponto (x,y), a direção do vetor gradiente é dada por:

$$\alpha(x, y) = \tan^{-1}\left(\frac{G_x}{G_y}\right) \tag{3.20}$$

De fato, o desalinhamento do eixo longitudinal do símbolo será determinado pela análise do histograma dos diversos ângulos de $\nabla\mathbf{f}$, presentes na imagem, ou seja, um ângulo preponderante é encontrado mediante a curva do histograma de ângulos.

Laplaciano

Uma outra metodologia para detecção de bordas baseia-se na convolução da imagem original por uma máscara composta pelo laplaciano de uma função gaussiana. Esta máscara geralmente é denominada filtro **LoG** (Laplacian of Gaussian).

O laplaciano de uma função $f(x,y)$ é dado por:

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \tag{3.20}$$

Uma vez que o laplaciano gera bordas duplas, além de possuir sensibilidade ao ruído, ele se torna um operador bastante ruidoso. Dessa forma, a aplicação do laplaciano para detecção de bordas baseia-se na propriedade de cruzamentos por zero, que proporciona uma determinação confiável das bordas.

Existem diversas maneiras de se implementar a equação 3.20 e aplicar a técnica de cruzamentos por zero, que consiste na convolução da imagem com o laplaciano de uma gaussiana bidimensional dada por:

$$h(x, y) = \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right) \tag{3.21}$$

Onde σ é o desvio padrão.

Seja $r^2 = x^2 + y^2$. A partir de (3.20) temos:

$$\nabla^2 h = \left(\frac{r^2 - \sigma^2}{\sigma^4} \right) \exp\left(- \frac{r^2}{2\sigma^2} \right) \tag{3.22}$$

A máscara de convolução resultante da equação 3.22 dependerá do valor de σ . Inicialmente, convoluímos a imagem original pela função $\nabla^2 h$, definindo um valor para σ. Esta operação resultará em uma imagem borrada cujos níveis de cinza mais escuros indicam valores mais negativos obtidos, os níveis brancos representam valores mais positivos e consequentemente os níveis intermediários indicam zeros. A partir daí, os valores positivos são rotulados como branco e os negativos como zero, binarizando a imagem. Dessa forma, as bordas serão dadas pelos pontos de transição entre os níveis de preto e branco, que representam os pontos de cruzamento por zeros.

3.1.4.3 Detecção de Passagem Por Zeros

Esta técnica, consiste na classificação dos pixels da imagem em duas classes (objeto e fundo) , mediante a convolução da imagem pelo filtro LoG (Laplaciano de uma função Gaussiana).

A binarização é obtida pela análise dos valores dos pixels após o processo de convolução, onde o conjunto de pixels com valores positivos serão definidos como objeto da imagem resultante, e o conjunto de pixels com valores não-positivos serão classificados como plano de fundo (ou simplesmente fundo) da imagem resultante.

O resultado da convolução da imagem original com a função $\nabla^2 h$ (3.22), apresenta um borramento intenso (Figura 3.7) . Dessa forma, a imagem binarizada pela detecção de passagem por zeros apresenta-se extremamente ruidosa, inviabilizando a princípio, a decodificação do símbolo.

3.1.4.4 Segmentação por Watershed

O método Watershed [11][12] é uma das ferramentas mais usadas em segmentação de imagens, e seu conceito é baseado na interpretação topográfica da imagem e sua inundação.

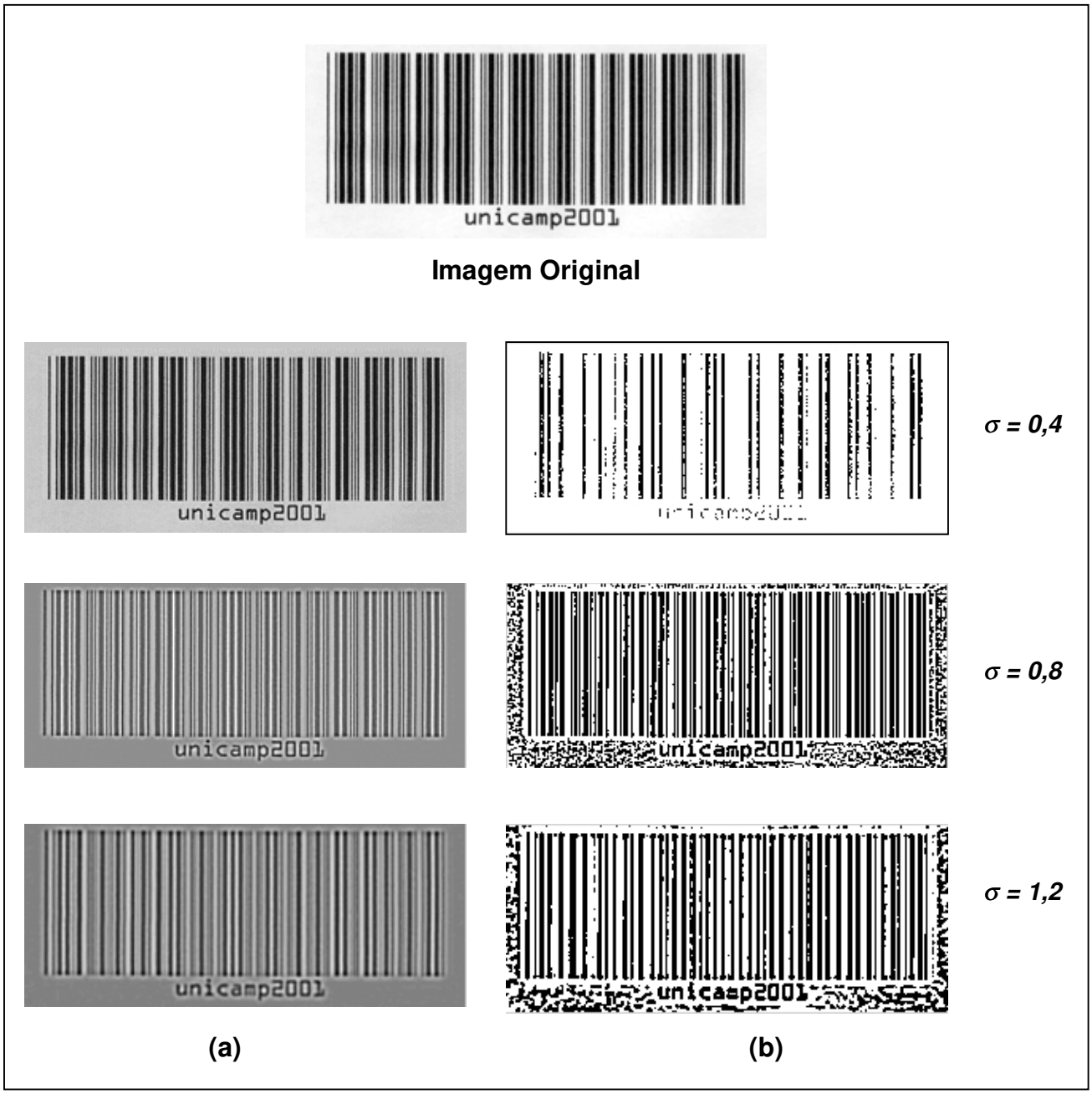


Figura 3.7 – (a) Convolução da imagem original com $\nabla^2 h$; (b) Limiarização por Zero

Seja uma imagem f (em nível de cinza), interpretada como uma superfície topográfica $\boldsymbol{S}$. Se imaginarmos gotas de água caindo sobre $\boldsymbol{S}$, elas irão escoar dos pontos mais altos da superfície em direção aos vales, que correspondem aos ***mínimos regionais*** da superfície $\boldsymbol{S}$.

Em outras palavras, um mínimo regional **M**, de imagem é um componente conexo de pixels com valor **v**, tal que todo pixel vizinho a **M** possui valor estritamente acima de **v**. Dessa forma, as águas acumular-se-ão em regiões denominadas ***bacias hidrográficas*** $\boldsymbol{H_f(M)}$ (Figura 3.8). Cada bacia hidrográfica referente a uma imagem f está associada a um mínimo regional **M**.

Inicialmente, para interpretar o processo de Watershed, façamos furos em cada mínimo regional presente na superfície topográfica $\boldsymbol{S}$, que representa de forma física a imagem inicial. Em seguida, imergimos $\boldsymbol{S}$ em um lago com velocidade vertical uniforme, provocando inundações nas bacias $\boldsymbol{H_f(M)}$. No decorrer das inundações, as águas provenientes de bacias distintas tendem a se encontrar. Entretanto, o nosso objetivo é evitar este encontro. Então, na iminência do encontro das águas, erguemos paredes (diques) a fim de evitar que elas se misturem. Construímos estes diques até o momento em que não haja mais ameaça de encontro das águas. No final deste processo, a coleção de todos os diques erguidos caracterizam as linhas do Watershed.

A descrição do *Watershed* também pode ser dada pelas regiões delimitadas pelos diques.

Os mínimos regionais são denominados marcadores, cuja função é identificar os locais onde o processo de inundação irá começar, ou seja, os locais dos furos feitos nas bacias $\boldsymbol{H_f(M)}$. A princípio, estes marcadores são dados pelos mínimos regionais como descrito acima. Normalmente, o *Watershed* é aplicado na imagem do gradiente ($\boldsymbol{g}$). Entretanto, existem muitas bacias hidrográficas na superfície $\boldsymbol{S}$ de $\boldsymbol{g}$ que não são expressas de forma visível na imagem original, pois, embora a imagem do gradiente evidencie de maneira relativamente clara as delimitações dos *objetos* na imagem, existem muitos mínimos regionais no plano de fundo que são visualizados de forma homogênea. Estes mínimos (ocultos a olho nú), são responsáveis por uma super-segmentação indesejada da imagem, inviabilizando a decodificação do símbolo (Figura 3.9).

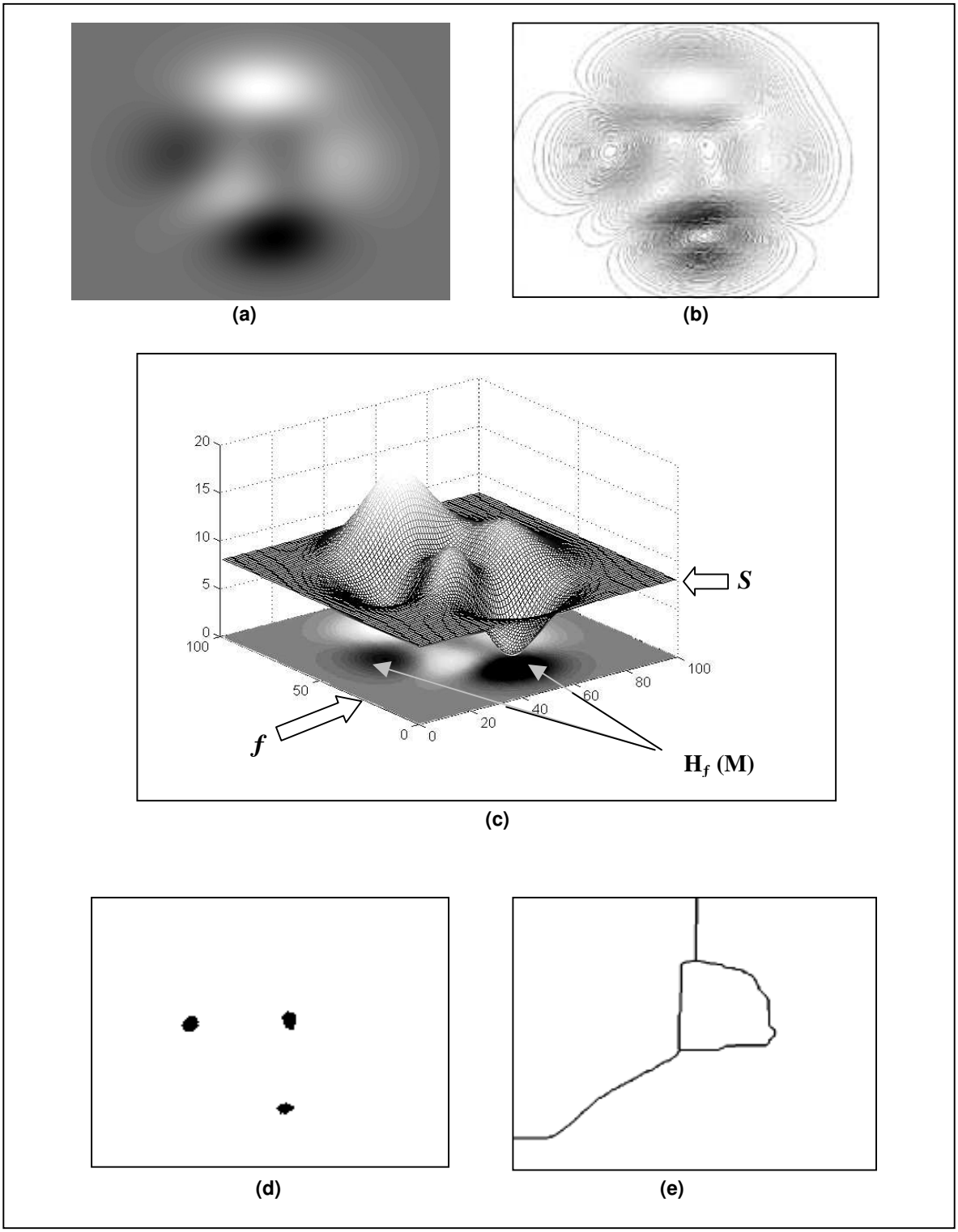


Figura 3.8 – Interpretação Física do Watershed
(a) Imagem Original ; (b) Curvas de nível ; (c) Representação topográfica da imagem original ; (d) Mínimos regionais ; (e) Linhas do Watershed.

A fim de evitar esta super-segmentação da imagem, podemos adotar marcadores obtidos por algum tipo de pré-processamento da imagem dos mínimos regionais do gradiente, obtendo-se assim $\mathfrak{I}_g(\boldsymbol{M})$. O que necessitamos, é na realidade a determinação de uma quantidade de marcadores que seja adequada a uma boa segmentação.

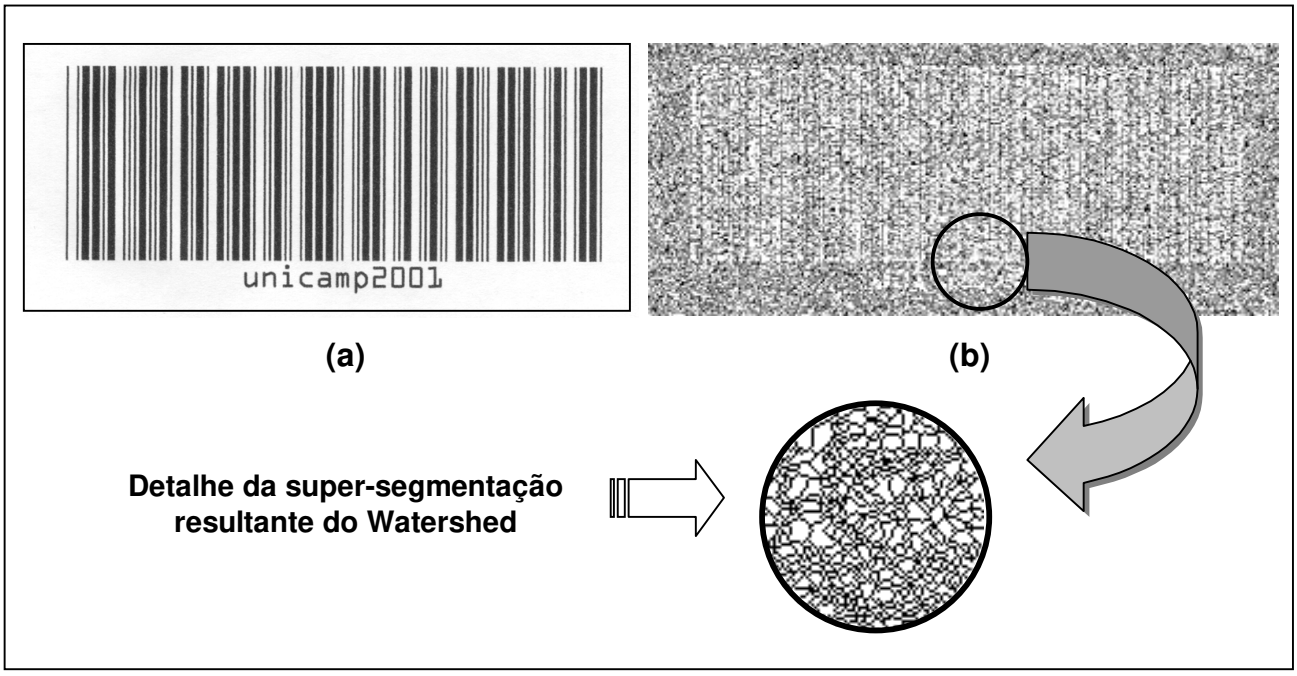


Figura 3.9 – (a) Imagem Original ; (b) Watershed

Considerando uma abordagem bidimensional, para uma delimitação clara das barras presentes na imagem do código de barras, devemos ter marcadores no interior de cada barra preta e uma externa a elas. A obtenção exata desta quantidade de marcadores exige um pré-processamento extra da imagem do gradiente, que varia de acordo com a resolução da imagem capturada e do ruído presente nela. Para as imagens estudadas, o pré-processamento aplicado baseou-se na remoção de mínimos regionais de $\boldsymbol{g}$, cujo contraste eram inferiores a um valor $\boldsymbol{k}$. Conforme a resolução de digitalização da imagem varia, este valor deve variar. A Figura 3.10 ilustra, a dificuldade em estipular um valor eficaz para $\boldsymbol{k}$, tomando-se como ponto de partida, a imagem original da Figura 3.7.

Através da Figura 3.10, notamos que para $\boldsymbol{k}$=35 ocorre uma junção acentuada entre barras. Se $\boldsymbol{k}$=40, ocorre uma diminuição de junções, embora já ocorra o início de uma

deterioração de barras. Entretanto, se ***k***=50, a junção entre barras cessa, mas a deterioração das barras é muito acentuada e resulta em uma decodificação incorreta.

A quantidade eficaz de marcadores a serem impostos e a determinação do valor ***k*** são fatores bastante subjetivos, inviabilizando o método de Watershed como processo de binarização para o Código de Barras .

Figura 3.10

3.1.5 Resultados Obtidos

Através da Figura 3.11, observamos que o método de limiarização de Otsu é o mais eficaz para a binarização da imagem do código.

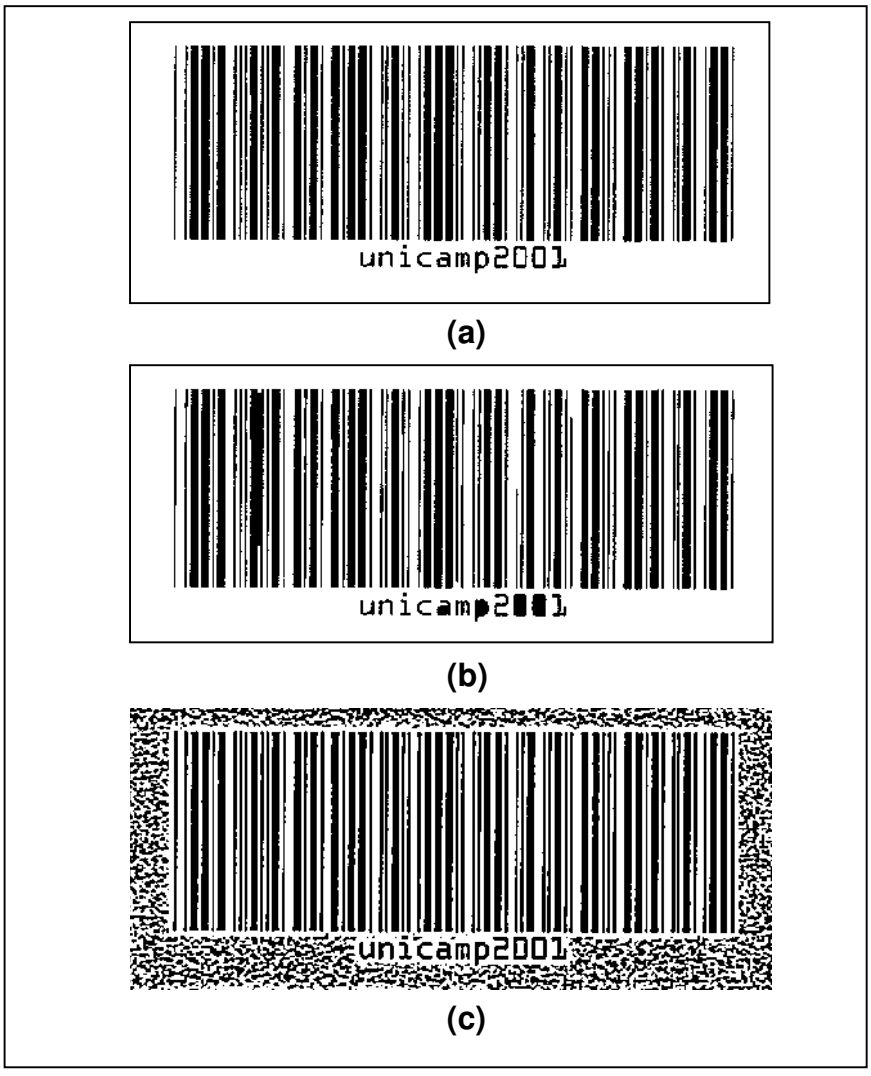


Figura 3.11 – Binarização (a) Por Limiarização de Otsu ; (b) Pelo método de Watershed ; (c) Por Detecção de Passagem por Zeros

Capítulo 4

Descrição de Técnicas Aplicadas

Para cada uma das simbologias abordadas no nosso trabalho, um algoritmo de decodificação foi desenvolvido com critérios distintos para obtenção de medidas e classificação, uma vez que cada simbologia possui características particulares de codificação no símbolo impresso. Inicialmente, são seguidos os três primeiros passos mostrados na figura 3.1, independentemente da simbologia analisada.

4.1 Code 39 (Código 3 de 9)

Para esta simbologia, após a aquisição da imagem de entrada, um vetor padrão é obtido, através de um processo particular de classificação. Este vetor corresponde a uma seqüência de palavras-código presentes na tabela padrão.

Os passos para a determinação deste vetor são explicados a seguir.

BINARIZAÇÃO

Mediante a imagem obtida a partir do processo de enquadramento, o passo seguinte é a binarização, descrito na seção 3.1.4.

A partir da imagem binarizada, realizamos então uma varredura longitudinal, obtendo um vetor linha de padrões $\boldsymbol{v_p}$, conforme ilustra a Figura 4.1.

ROTULAÇÃO

A partir do vetor linha de padrões v_p, rotulamos todos os seus componentes conexos (seqüências de elementos com mesmo nível de cinza) com valores ascendentes. Um novo vetor resultante r_p é obtido, representando cada elemento da informação codificada através de rótulos. A figura 4.2 ilustra este passo.

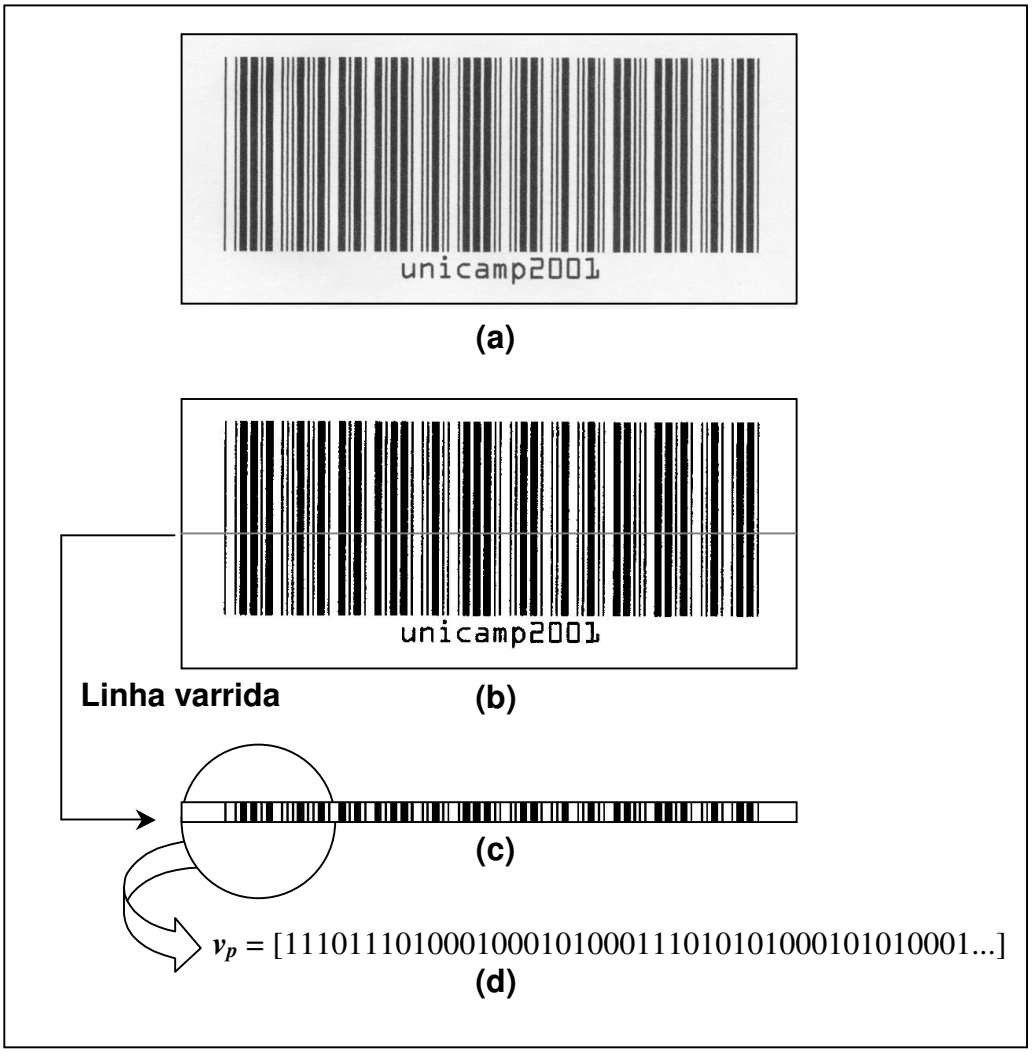


Figura 4.1 - (a) Imagem após o processo de Enquadramento (256 níveis de cinza); (b) Imagem Binarizada por Limiarização de Otsu (2 níveis de cinza: "0" e "1"); (c) Representação da linha varrida, ampliada na vertical; (d) Vetor linha de padrões vp que representa a linha varrida.

Através da Figura 4.2, vemos que, para cada elemento (barra ou espaço) subseqüente, um rótulo é imposto de forma ascendente. É obedecida a razão de 3:1 entre elementos largos e estreitos, e a margem de silêncio possui 11 vezes a largura de um elemento estreito.

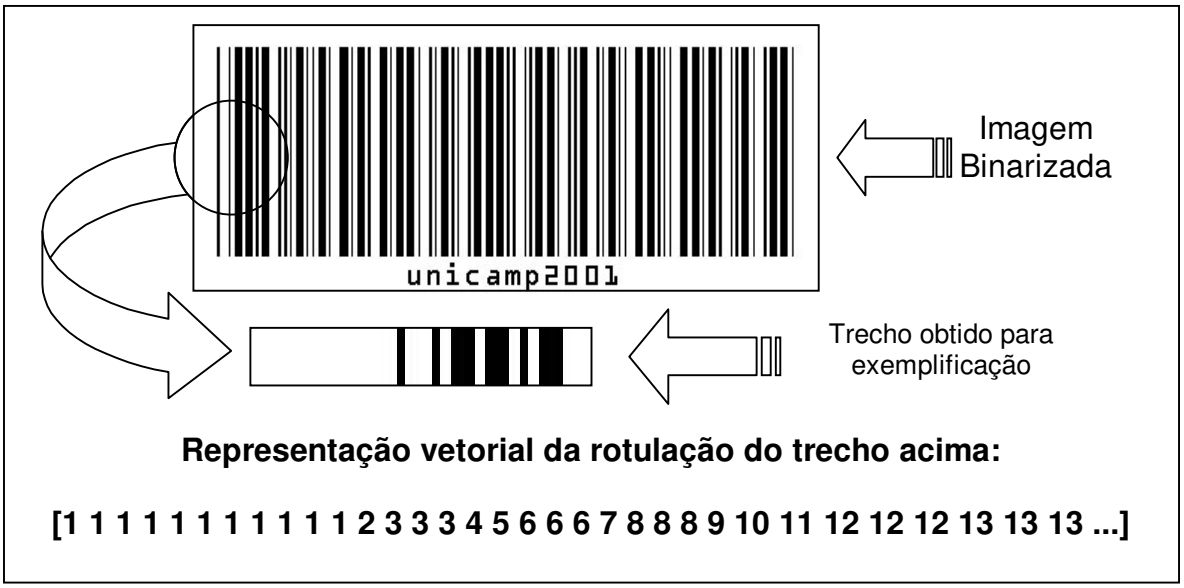


FIGURA 4.2 – Ilustração de uma rotulação

ESTIMATIVA DA LARGURA DO ELEMENTO MAIS ESTREITO

Após a rotulação dos elementos encontrados, achamos o histograma do vetor de rótulos r_p, a fim de obtermos uma estimativa da largura (em pixels) do elemento mais estreito, l_e. Esta estimativa é necessária, pois indica se a resolução está igual ou acima da mínima tolerável e determina qual é a dimensão da área responsável pela abertura que aplicaremos no passo seguinte.

A largura do elemento mais estreito l_e, encontrado na varredura, é determinada analisando a função distribuição do histograma do vetor de rótulos. A princípio, sabemos que a quantidade de elementos estreitos é maior que a quantidade dos elementos largos, pois

para cada palavra-código existem seis elementos estreitos e três largos. Dessa forma, pela função distribuição obtemos qual é o maior valor de ocorrência de elementos rotulados. Para este valor de ocorrência preponderante, achamos qual é a largura correspondente, pois ela corresponde à largura do elemento mais estreito.

A Figura 4.3 ilustra o histograma do vetor de rótulos obtido no passo anterior, (cuja imagem de entrada corresponde à imagem original da Figura 3.7) e a função distribuição das larguras dos elementos rotulados.

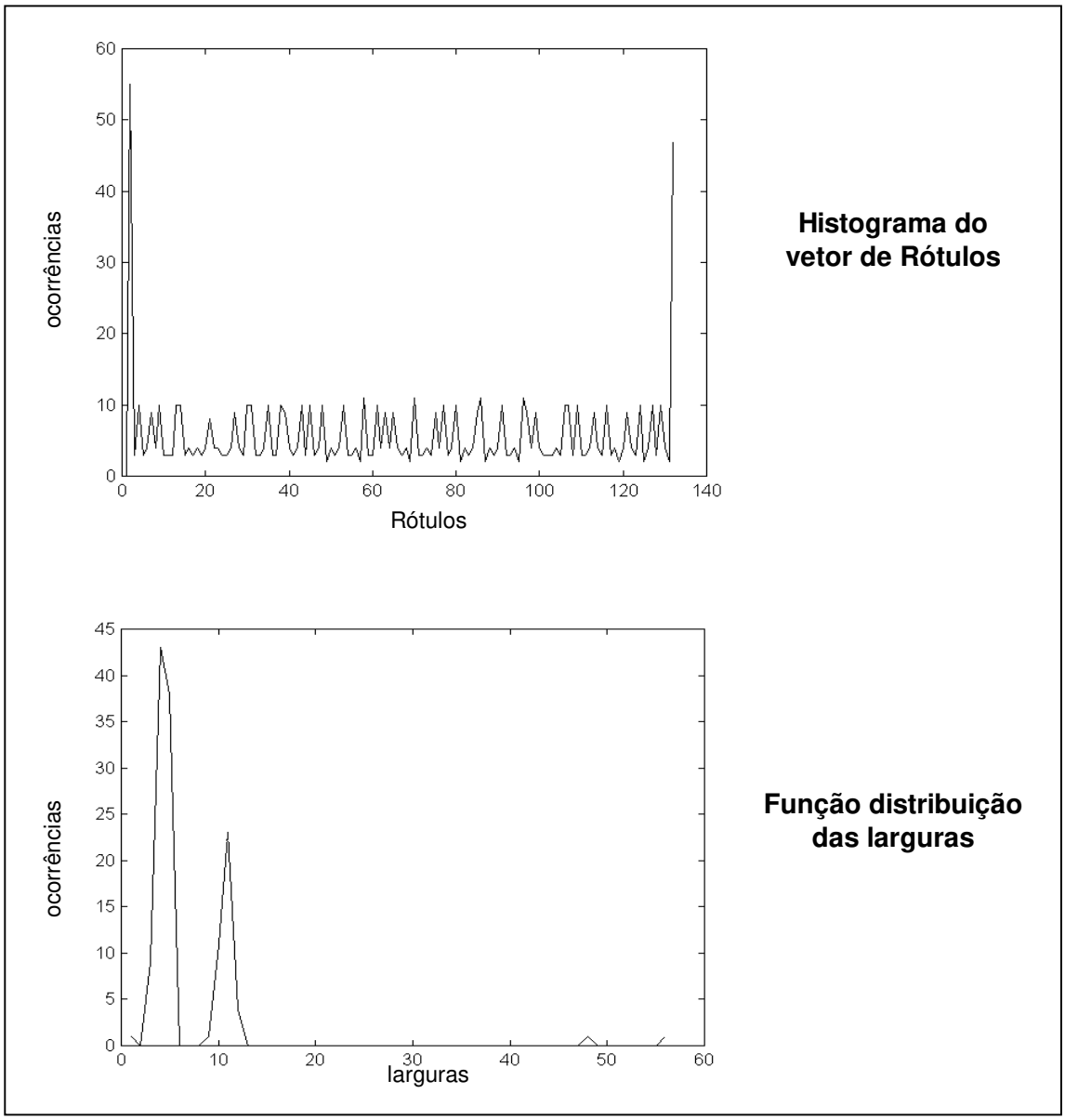


Figura 4.3

Pelos gráficos da Figura 4.3, vemos que a largura com maior índice de ocorrência é aquela com valor igual a 4. Assim, $l_e = 4$.

LOCALIZAÇÃO DO CAMPO DE PADRÕES

Uma vez definida a largura l_e, encontrada na varredura, necessitamos aplicar a técnica de abertura por área no vetor v_p para localizar com precisão o campo de dados que contém os padrões das palavras-código, ou seja, devemos extrair o campo de padrões ($\mathbf{C_P}$) entre as zonas de silêncio, pois apenas ele contém as barras e espaços que codificam a informação a ser obtida.

A imagem de análise para a aplicação da abertura por área é, na realidade, o vetor v_p, contendo seqüências de elementos com valores 0 (zero) e 1 (um). Os segmentos de valor "1" correspondem às margens de silêncio (presentes à esquerda e direita do símbolo) e aos espaços entre as barras (veja Figura 4.4).

O princípio da abertura por área baseia-se na idéia de comparar as áreas de componentes conexos correspondentes ao objeto da imagem, com um valor de área pré-definido, A . Caso a área de cada componente conexo seja igual ou superior ao valor A, este componente será objeto da imagem resultante, caso contrário pertencerá ao plano de fundo. Na nossa análise, o valor de área torna-se um valor de comprimento, pois a imagem de entrada possui apenas uma linha, caracterizada pelo vetor linha de padrões v_p.

O objetivo deste processo consiste na remoção dos elementos, presentes no vetor v_p, cujas larguras sejam inferiores ao valor de comparação A . Obtermos então como resultado, um vetor constituído apenas pelas margens de silêncio que serão o objeto (valor "1") deste vetor resultante, e o fundo (valor "0") indica o local do campo $\mathbf{C_P}$. A Figura 4.4 ilustra de forma mais clara este processo.

O valor "A" é um fator importante, pois garante uma extração correta do campo de padrões, que é o nosso alvo de análise neste passo. É necessária uma escolha de um valor que favoreça a obtenção de apenas três componentes conexos (dois componentes com valor "1", e um componente com valor "0") como resultado desta operação. Os dois componentes

de valor "1" corresponderão às margens de silêncio (esquerda e direita), e o componente com valor "0", ao campo de padrões. Portanto, o valor de comparação de área escolhido deve ser um múltiplo do valor da largura mais estreita l_e. No algoritmo implementado, utilizamos um valor 5 vezes maior que a largura mais estreita ($A = 5 \cdot l_e$), garantindo uma extração correta de $\mathbf{C_P}$. Notamos que todos os espaços (componentes de valor "1" presentes no campo $\mathbf{C_P}$) serão "excluídos" após a abertura por área, recebendo valor "0". Eventualmente, devido a imperfeições de impressão, manchas ou outros tipos de interferências na leitura do código, após a aplicação da abertura por área, o vetor resultante poderá apresentar mais ou menos de três componentes conexos. Se o vetor resultante contiver menos de três componentes, provavelmente a largura das zonas de silêncio não foram respeitadas no momento de impressão da simbologia, pois o valor correto da largura das zonas de silêncio deve ser aproximadamente 11 vezes o valor da largura do elemento mais estreito presente no símbolo.

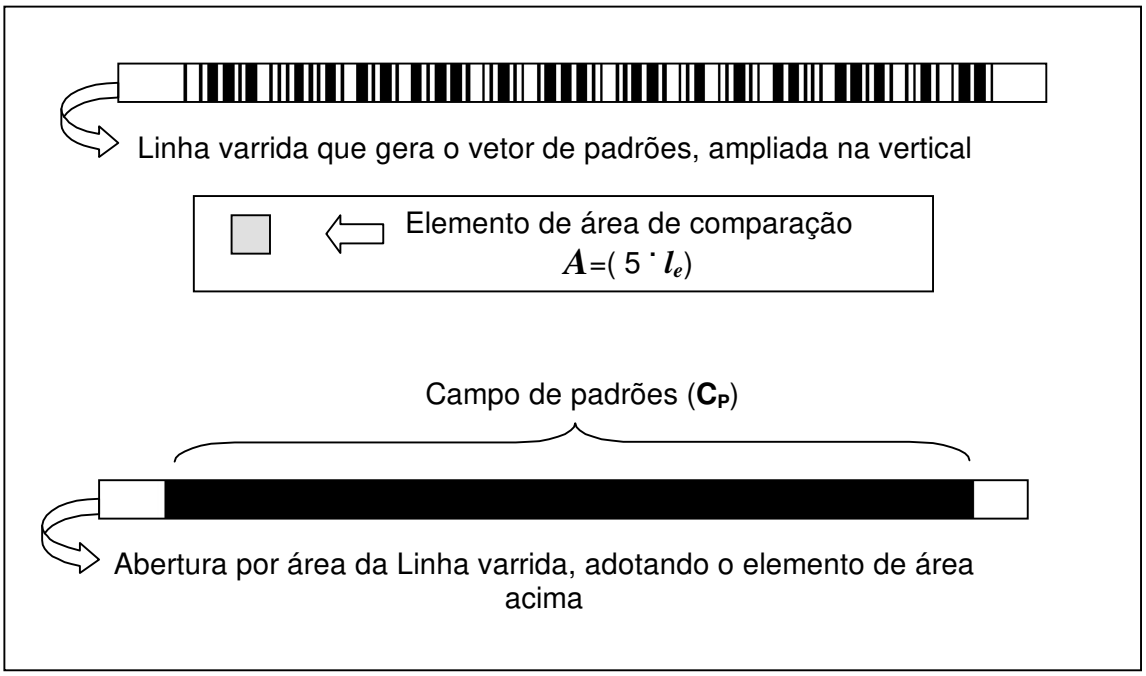


FIGURA 4.4 – Ilustração da Abertura por Área

Analisando a Figura 4.4, notamos que o elemento de área adotado (s_e) só "encaixa" nos locais das margens de silêncio. Como o elemento s_e é mais largo que os espaços (componentes brancos) entre as barras pretas, estes espaços são excluídos, ou seja, recebem valor "0", tornando-se fundo do vetor resultante. Apenas as margens de silêncio permanecem com valor "1" (branco), indicando pixels do objeto.

VERIFICAÇÃO DAS MARGENS E OBTENÇÃO DE MEDIDAS

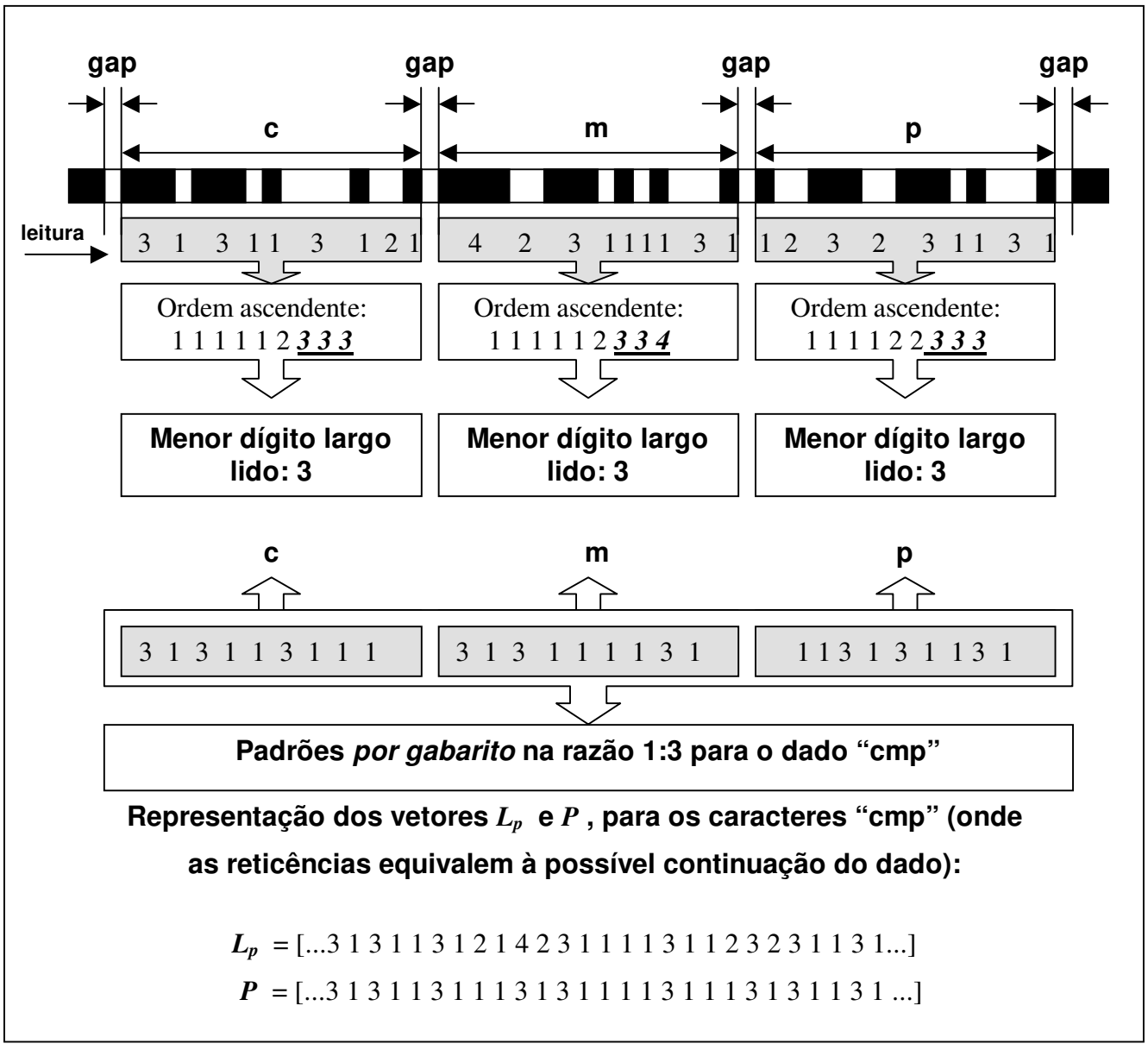
Uma vez verificado que a imagem resultante da abertura por área apresenta três componentes conexos (dois componentes com valor "1" e um componente de valor "0"), extraímos aqueles situados no campo $\mathbf{C_P}$ que foram rotulados anteriormente a fim de determinarmos quais são as larguras de cada elemento que compõe o dado armazenado.

CLASSIFICAÇÃO E DECODIFICAÇÃO

A partir dos elementos rotulados presentes no campo $\mathbf{C_P}$, obtemos o histograma dos rótulos, que nos fornece uma seqüência de larguras consecutivas por meio de um novo vetor de larguras, $\boldsymbol{L_P}$. O vetor é particionado em subconjuntos de nove elementos, correspondendo às palavras-código do símbolo, uma vez que elas possuem 9 elementos (barras e espaços). A quantidade de subconjuntos de nove elementos corresponde à quantidade de palavras-código presentes no símbolo e que codifica a informação.

Devemos identificar a largura de cada elemento presente em cada subconjunto, que deve possuir 6 elementos estreitos e 3 elementos largos. Através do vetor $\boldsymbol{L_P}$, adotando passos de 9, ordenamos em ordem ascendente, os valores de larguras dos subconjuntos. Os três últimos valores de cada subconjunto correspondem aos três elementos mais largos da palavra-código; os remanescentes correspondem aos seis elementos mais estreitos. Dessa forma, para cada valor de largura inferior ao menor dos três elementos mais largos, interpretamos como sendo um elemento estreito, e os outros três elementos com valores iguais ou acima a este terceiro elemento mais largo, são interpretados como elementos largos. A figura 4.5 ilustra este passo de classificação.

Após interpretar cada elemento do vetor L_P, criamos um vetor de padrão final (P), a partir daquele. Os elementos identificados como largos receberão uma seqüência de 3 dígitos binários com rótulo correspondente ao seu valor original (se barra, recebe rótulo "0" ; se espaço, recebe rótulo "1"). Em contrapartida, os elementos estreitos recebem um único dígito binário, correspondente ao seu valor original, proporcionando dessa forma uma razão entre elementos largos e estreitos de 3:1.



Representação dos vetores L_p e P , para os caracteres "cmp" (onde as reticências equivalem à possível continuação do dado):

L_p = [...3 1 3 1 1 3 1 2 1 4 2 3 1 1 1 1 3 1 1 2 3 2 3 1 1 3 1...]

P = [...3 1 3 1 1 3 1 1 1 3 1 3 1 1 1 1 3 1 1 1 3 1 3 1 1 3 1 ...]

Figura 4.5 – Classificação

Cada seqüência de 15 bits (correspondentes a 6 elementos estreitos de 1 bit e 3 elementos largos de 3 bits) do vetor ***P*** são comparados com a matriz de padrões do Código 39 (ver Tabela II), que contém 44 linhas de padrões correspondentes aos 44 caracteres que esta simbologia é capaz de codificar. Através desta comparação, ocorrerá um casamento entre o padrão da seqüência de 15 bits e alguma linha da matriz. A linha da matriz onde ocorrer o casamento dos padrões, indica qual é o caractere codificado no símbolo.

4.2 Código UPC/EAN

Como os códigos UPC e EAN possuem a mesma estrutura de codificação, o processo de decodificação para ambos é idêntico. A análise dos Códigos UPC/EAN difere daquela referente ao código 39, pois suas formas de decodificação não estão relacionadas apenas à interpretação de duas larguras, mas sim de quatro larguras distintas. Com isso, é necessário uma normalização das larguras dos elementos que compõem as palavras-código.

Alguns procedimentos seguidos para decodificar estes códigos são semelhantes àqueles necessários à decodificação do código 39.

PROCESSAMENTOS INICIAIS

O passo de limiarização consiste, assim como no caso do código 39, no passo inicial após o enquadramento do símbolo.

Para a simbologia UPC/EAN, o valor de limiar é um fator bastante crítico, pois os elementos que compõem uma palavra código podem diferir entre si de quatro maneiras distintas, exigindo uma boa precisão na determinação de cada largura.

EXTRAÇÃO DE UMA FAIXA DE PADRÕES

Uma vez que estas simbologias exigem uma precisão mais apurada das larguras dos elementos, extraímos uma faixa de padrões a fim de capturarmos mais informações a respeito da distribuição de larguras. Em outras palavras, as divergências entre uma largura lida daquela ideal são amenizadas pela captura de diversas linhas de varredura e

determinação de uma média para cada largura subseqüente. A Figura 4.6 ilustra o procedimento de captura de uma faixa do símbolo, que será analisada para a obtenção da informação codificada. A faixa extraída da imagem do símbolo é interpretada como um empilhamento de linhas que o compõem.

Após a extração da faixa, uma verificação da quantidade de elementos para cada linha deve ser feita, a fim de garantir uma quantidade correta de barras e espaços que uma simbologia UPC/EAN deve possuir. O símbolo do UPC ou EAN é formado de doze padrões compostos de quatro elementos (barras e espaços) cada, dois padrões laterais com três elementos e um padrão central de cinco elementos, portanto em uma leitura longitudinal da imagem são encontrados cinqüenta e nove elementos correspondentes ao símbolo e mais dois referentes às margens de silêncio. Assim, uma análise é feita nas linhas empilhadas que compõem a faixa extraída. Caso uma linha não apresente a quantidade correta de elementos ela será removida da faixa de análise . A presença de linhas com uma quantidade errada de elementos ocorre devido às interferências no momento da digitalização do código.

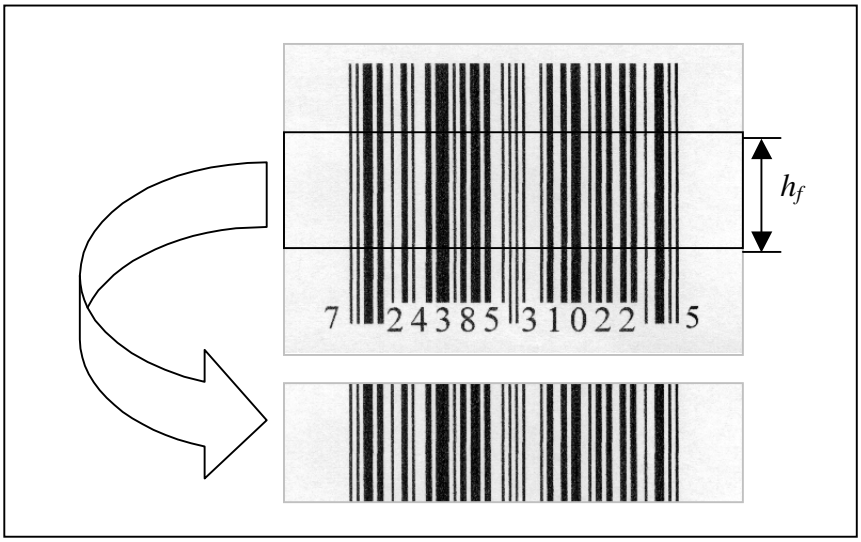


Figura 4.6 – Faixa Extraída

A quantidade de elementos encontrados em uma linha corresponde à quantidade de rótulos resultantes após o processo de rotulação.

DISTRIBUIÇÃO DE LARGURAS E EXTRAÇÃO DO CAMPO DE PADRÕES

Após a eliminação de linhas ruidosas da faixa extraída, obtemos uma nova faixa filtrada que eventualmente pode conter a mesma quantidade de linhas da faixa inicial (caso todas as linhas verificadas possuam a quantidade correta de elementos), ou uma quantidade inferior de linhas (caso sejam detectadas linhas ruidosas). Para imagens com boa resolução, não haverá linhas ruidosas, e a faixa escolhida permanece inalterada.

Uma rotulação é aplicada na faixa filtrada a fim de medirmos as áreas correspondentes aos blocos dos elementos concatenados. A partir daí, o artifício para determinar a largura dos elementos é dado pela divisão da área de cada elemento pela altura h_f da faixa filtrada. Esta forma de determinação das larguras equilibra pequenas discrepâncias entre as linhas empilhadas, devido ao ruído da leitura. Encontramos então um vetor de padrões $\boldsymbol{v_{UPC}}$ com seqüências correspondentes aos elementos de larguras concatenadas. Veja figura 4.7.

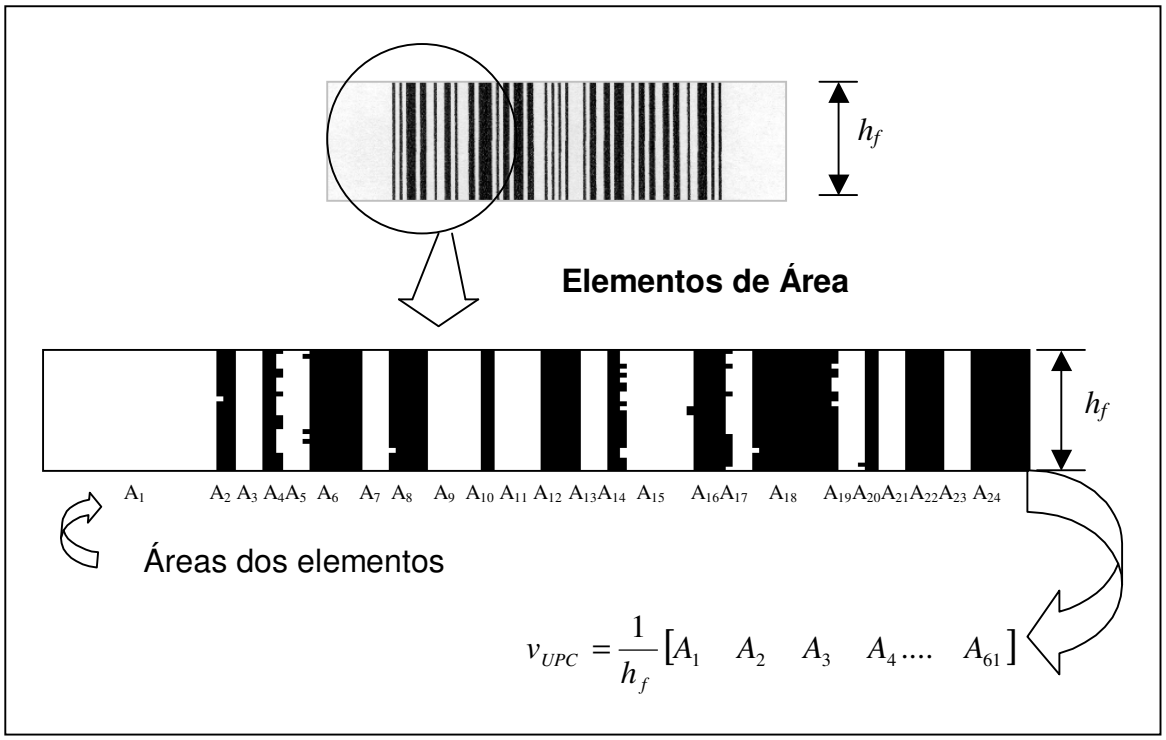


Figura 4.7 – Obtenção do vetor de padrões

Por meio do histograma do vetor $\boldsymbol{v_{UPC}}$ achamos o gráfico de distribuição de larguras que informa o valor estimado do elemento mais estreito. De forma idêntica ao código 39, um processo de abertura por área é aplicado no vetor $\boldsymbol{v_{UPC}}$, a fim de localizar o campo de padrões.

NORMALIZAÇÃO E CLASSIFICAÇÃO

Cada palavra código possui quatro elementos dispostos em sete módulos (ver seção 2.4.2). Assim, após os três elementos correspondentes ao padrão lateral (010), selecionamos seis subconjuntos sucessivos compostos de quatro elementos, referentes aos seis dígitos do lado esquerdo do padrão central. Em seguida, selecionamos mais seis subconjuntos de quatro elementos, após o padrão central (01010), que correspondem aos outros seis dígitos remanescentes.

Um valor de largura de módulo $\boldsymbol{l_m}$ deve ser determinado, para que os elementos de largura que compõem um dígito sejam interpretados como múltiplo de $\boldsymbol{l_m}$. Este valor pode ser determinado de duas maneiras: analisando todas as larguras encontradas na varredura do símbolo, ou através de cada palavra-código referente a um dígito codificado.

Um símbolo UPC ou EAN possui noventa e cinco módulos que correspondem aos 84 módulos dos doze dígitos (12 x 7 = 84) somados aos onze módulos correspondentes aos dois padrões laterais de três módulos (cada) e cinco módulos do padrão central (Figura 2.3). Dessa forma, para estimar o valor de largura de um módulo, dividimos a largura total (em pixels) da linha varrida no campo de padrões por 95. A eficiência da obtenção de $\boldsymbol{l_m}$ por este método é afetada pela falta de uniformidade entre os elementos. Ou seja, para um determinado valor de largura de módulo **X** encontrado em uma palavra-código lida, existirá um outro valor de largura de módulo que se distancia de **X**, encontrado em uma palavra posterior. Entretanto, estes valores de largura de módulos devem ser idênticos no caso ideal. Portanto, o aumento do domínio de análise para definir $\boldsymbol{l_m}$ diminui a eficiência do valor calculado, favorecendo a erros de decodificação conforme verificamos em testes implementados.

A forma mais confiável de determinar o valor de módulo $\boldsymbol{l_m}$ é dada a partir de cada palavra-código encontrada, ou seja, para cada conjunto de quatro elementos correspondentes a um dígito, é calculado um valor $\boldsymbol{l_m}$. Inicialmente, achamos a soma das larguras dos 4 elementos que compõem um dígito e em seguida dividimos este resultado por 7, obtendo-se então $\boldsymbol{l_m}$. Como o valor da largura de cada elemento está sob ponto flutuante, o valor de $\boldsymbol{l_m}$ estará também sob esta representação.

Durante a análise de cada palavra-código, após a determinação de $\boldsymbol{l_m}$, os elementos que compõem a palavra serão normalizados com base no valor $\boldsymbol{l_m}$, e em seguida obtemos 6 parâmetros, referentes aos padrões lidos, apresentados mais adiante.

Para definir os dígitos codificados, aplicamos uma ou duas classificações. Uma segunda classificação será realizada, caso a primeira resulte em um dos dígitos dos pares (1 e 7) ou (2 e 8), uma vez que estes pares de dígitos apresentam os parâmetros t_1 e t_2 idênticos; caso contrário, a primeira classificação decodifica com eficiência os dígitos presentes no símbolo. Uma tabela de classificação (Tabela III) é usada para a primeira classificação das palavras-código. Os parâmetros que compõem esta tabela são os coeficientes c_1, c_2, c_3, c_4, as distâncias t_1, t_2 (analisados na Tabela 2.1) e um parâmetro adicional ***CA*** utilizado para uma eventual segunda classificação .

Obtemos a partir da Tabela III**,** os vetores-padrão de características dados por:

$$x(d) = [c_1 \quad c_2 \quad c_3 \quad c_4 \quad t_1 \quad t_2] \tag{3.23}$$

A primeira classificação dos dígitos codificados é dada por meio da comparação entre o vetor *x(d)*, cujos atributos são dados pelos valores da Tabela III, e o vetor de características *x'*, cujos atributos são obtidos a partir da varredura do símbolo.

$$x' = [C'_1 \quad C'_2 \quad C'_3 \quad C'_4 \quad t'_1 \quad t'_2] \tag{3.24}$$

Uma vez calculado o vetor *x'*, o dígito *"**d**"* da Tabela III para o qual ocorra o menor valor para a distância:

$$D = \sum_{i=1}^{6} \left(x(d) - x \right)^2 \qquad (3.25)$$

indicará o dígito codificado.

Dígito	Larguras dos Elementos				Dígitos da Esquerda		Dígitos da Direita		*CA*
	c_1	c_2	c_3	c_4	t_1	t_2	t_1	t_2	c_1+c_3 / c_2+c_4
1	2	2	2	1	3	4	4	4	4/3
2	2	1	2	2	4	3	3	3	4/3
3	1	4	1	1	2	5	5	5	2/5
4	1	1	3	2	3	4	2	4	4/3
5	1	2	3	1	4	5	3	5	4/3
6	1	1	1	4	5	2	2	2	2/5
7	1	3	1	2	3	4	4	4	2/5
8	1	2	1	3	4	3	3	3	2/5
9	3	1	1	2	3	2	4	2	4/3
10	3	2	1	1	4	3	5	3	4/3

Tabela III – Tabela para Classificação dos padrões

A Figura 4.8 ilustra um exemplo para a primeira classificação.

Mais um parâmetro adicional ***CA*** = (c_1+c_3 / c_2 +c_4) é utilizado na segunda classificação, caso a primeira classificação resulte em um dos dígitos 1, 2, 7 ou 8 .

Pela tabela de classificação (Tabela III) vemos que a razão entre os parâmetros ***CA*** para esses pares é dada por:

$$\boldsymbol{CA(1) = 3{,}33\ CA(7)}$$
$$\boldsymbol{CA(2) = 3{,}33\ CA(8)}$$

tanto para os dígitos com paridade ímpar ou par. Concluímos que o parâmetro ***CA*** distingue de forma eficaz os pares (1 e 7) e (2 e 8).

Analisando a Figura 4.8, vemos que o dígito para o qual ocorre a menor distância *"D"* é o dígito 7. Dessa forma, para a primeira classificação, o dígito lido é sete. Como esta

primeira classificação resultou no dígito 7, a segunda classificação é realizada, pois o dígito 7 pode ser resultante de uma interpretação equivocada do dígito 1. Comparando-se o resultado de ***CA'*** para o dígito lido, com os parâmetros ***CA's*** para os dígitos 1 e 7 da Tabela III, certificamos que o dígito lido realmente é o dígito 7, uma vez que:

$\boldsymbol{CA}(1) = 1{,}333$

$\boldsymbol{CA}(7) = 0{,}4$

$\boldsymbol{CA'} = C'_1 + C'_3 / C'_2 + C'_4 = 1{,}26+0{,}7/2{,}94+2{,}1 = 0{,}388888 \cong 0{,}4 = \boldsymbol{CA}(7)$

Características de um dígito lido no campo esquerdo

Larguras Obtidas	
A_1/h_f	1,8
A_2/h_f	4,2
A_3/h_f	1,0
A_4/h_f	3,0

$l_m = (1{,}8 + 4{,}2 + 1 + 3) / 7 = 1{,}4285714$

$C'_1 = 1{,}8 \ / \ l_m = 1{,}26$

$C'_2 = 4{,}2 \ / \ l_m = 2{,}94$

$C'_3 = 1{,}0 \ / \ l_m = 0{,}7$

$C'_4 = 3{,}0 \ / \ l_m = 2{,}1$

$t'_1\,(esquerda) = C'_3 + C'_4 = 2{,}8$

$t'_2 = C'_2 + C'_3 = 3{,}64$

$x' = [C'_1 \quad C'_2 \quad C'_3 \quad C'_4 \quad t'_1 \quad t'_2] = [1{,}26 \ \ 2{,}94 \ \ 0{,}7 \ \ 2{,}1 \ \ 2{,}8 \ \ 3{,}64]$

d	Campo Esquerdo					
	c_1	c_2	c_3	c_4	t_1	t_2
1	2	2	2	1	3	4
2	2	1	2	2	4	3
3	1	4	1	1	2	5
4	1	1	3	2	5	4
5	1	2	3	1	4	5
6	1	1	1	4	5	2
7	1	3	1	2	3	4
8	1	2	1	3	4	3
9	3	1	1	2	3	2
10	3	2	1	1	2	3

$$D = \sum_{i=1}^{6} (x(\boldsymbol{d}) - x)^2$$

$D_1 = (2\text{-}1{,}26)^2 + (2\text{-}2{,}94)^2 + (2\text{-}0{,}7)^2 + (1\text{-}2{,}1)^2 + (3\text{-}2{,}8)^2 + (4\text{-}3{,}64)^2 = 4.5008$

$D_2 = (2\text{-}1{,}26)^2 + (1\text{-}2{,}94)^2 + (2\text{-}0{,}7)^2 + (2\text{-}2{,}1)^2 + (4\text{-}2{,}8)^2 + (3\text{-}3{,}64)^2 = 7.8608$

$D_3 = (1\text{-}1{,}26)^2 + (4\text{-}2{,}94)^2 + (1\text{-}0{,}7)^2 + (1\text{-}2{,}1)^2 + (2\text{-}2{,}8)^2 + (5\text{-}3{,}64)^2 = 4.9808$

$D_4 = (1\text{-}1{,}26)^2 + (1\text{-}2{,}94)^2 + (3\text{-}0{,}7)^2 + (2\text{-}2{,}1)^2 + (5\text{-}2{,}8)^2 + (4\text{-}3{,}64)^2 = 14.1008$

$D_5 = (1\text{-}1{,}26)^2 + (2\text{-}2{,}94)^2 + (3\text{-}0{,}7)^2 + (1\text{-}2{,}1)^2 + (4\text{-}2{,}8)^2 + (5\text{-}3{,}64)^2 = 10.7408$

$D_6 = (1\text{-}1{,}26)^2 + (1\text{-}2{,}94)^2 + (1\text{-}0{,}7)^2 + (4\text{-}2{,}1)^2 + (5\text{-}2{,}8)^2 + (2\text{-}3{,}64)^2 = 15.0608$

$D_7 = (1\text{-}1{,}26)^2 + (3\text{-}2{,}94)^2 + (1\text{-}0{,}7)^2 + (2\text{-}2{,}1)^2 + (3\text{-}2{,}8)^2 + (4\text{-}3{,}64)^2 = \underline{\mathbf{0.3408}}$

$D_8 = (1\text{-}1{,}26)^2 + (2\text{-}2{,}94)^2 + (1\text{-}0{,}7)^2 + (3\text{-}2{,}1)^2 + (4\text{-}2{,}8)^2 + (3\text{-}3{,}64)^2 = 3.7008$

$D_9 = (3\text{-}1{,}26)^2 + (1\text{-}2{,}94)^2 + (1\text{-}0{,}7)^2 + (2\text{-}2{,}1)^2 + (3\text{-}2{,}8)^2 + (2\text{-}3{,}64)^2 = 9.6208$

$D_{10} = (3\text{-}1{,}26)^2 + (2\text{-}2{,}94)^2 + (1\text{-}0{,}7)^2 + (1\text{-}2{,}1)^2 + (2\text{-}2{,}8)^2 + (3\text{-}3{,}64)^2 = 6.2608$

Figura 4.8 – Exemplo da Primeira classificação para um dígito lido no campo esquerdo

Capítulo 5

Pré-processamentos

No nosso estudo, consideramos uma imagem digitalizada contendo um símbolo em uma posição aleatória. A captura do símbolo deve ser realizada em uma resolução de no mínimo 300 pontos por polegada, a fim de garantir uma estimativa de, no mínimo, 4 pixels para uma largura de módulo da simbologia presente na imagem.

5.1 Análise dimensional

A representação simbólica do código UPC-A no tamanho padrão em magnitude 1, apresenta uma largura de módulo de 0,33mm. Dessa forma, uma palavra-código representando um dígito codificado possui $7 \times 0{,}33\text{mm} = 2{,}31\text{mm}$.

A largura de todos os dígitos codificados, padrões de guarda laterais e central somam $95 \times 0{,}33\text{mm} = 31{,}35\text{mm}$. As margens de silêncio para o código UPC-A possuem como largura, 9 vezes a largura de um módulo, tanto na esquerda como na direita.

Acima das barras deve haver um espaçamento de 1 largura de módulo, assim como entre as barras e os números arábicos que expressam o dado armazenado. A Figura 5.1 ilustra um gabarito para o símbolo UPC-A.

Através da Figura 5.1, vemos que a área total de um símbolo ideal é dado por $37{,}29\text{mm} \times 26{,}26\text{mm} = 9{,}79\ \text{cm}^2$, que resulta em uma área de aproximadamente 440×310 pixels para uma digitalização em 300 pontos por polegada.

5.2 Etapas de Pré-Processamentos

O símbolo presente na imagem pode estar imerso em um plano de fundo que varia muito conforme o produto em questão. Para uma localização eficiente, a impressão do símbolo deve obedecer as normas dimensionais.

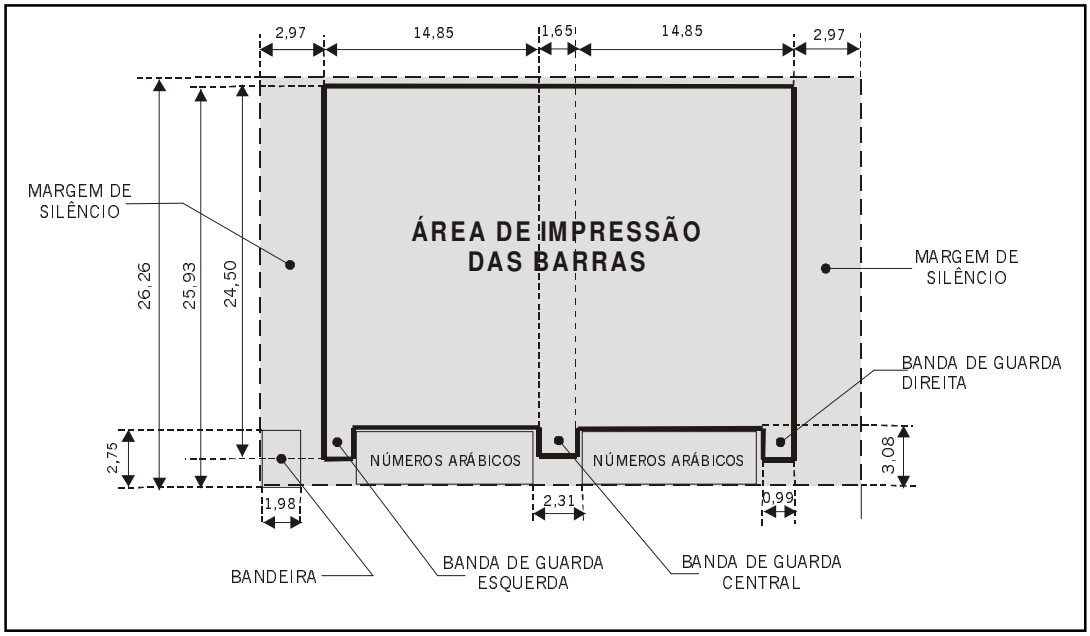


Figura 5.1 – Dimensões ideais para o símbolo UPC-A em tamanho nominal (Medidas em milímetros)

5.2.1 Captação de contornos

Inicialmente, a partir da imagem original em nível de cinza, obtemos a imagem da magnitude do vetor gradiente, que ressalta os contornos dos objetos presentes na imagem. Este processo é realizado para obtermos o conjunto de retângulos que circundam todas as barras do símbolo. A figura 5.2 ilustra o processo de determinação de contornos para uma imagem de 452 × 462 pixels.

5.2.2 Estimação da localização do símbolo

A partir da imagem com os contornos das barras, sabendo-se que os espaçamentos entre elas não devem ultrapassar 4 vezes o valor de uma largura de módulo, aplicamos uma abertura com elemento estruturante circular de diâmetro 20 pixels. A escolha deste valor como diâmetro garante um nivelamento de todas as barras em uma faixa de valor de cinza reduzido. A nuvem escura com forma retangular mostrada na Figura 5.3 ilustra este processo.

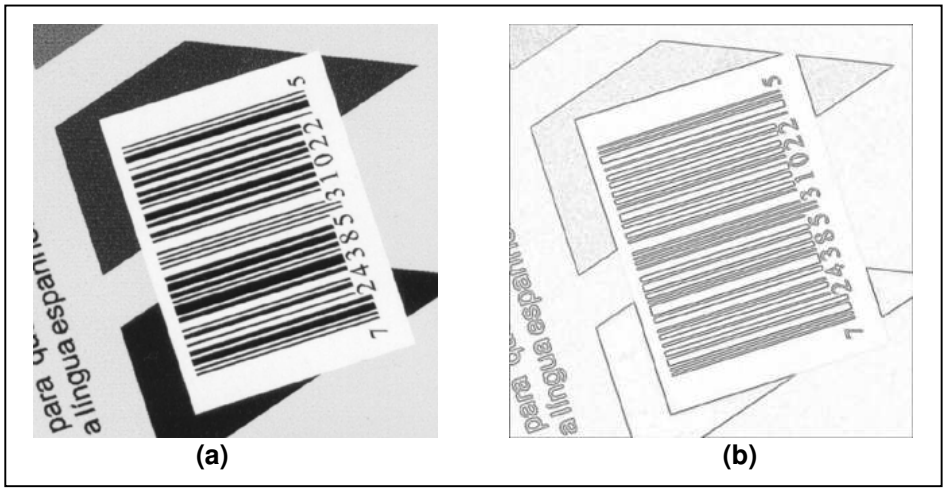


Figura 5.2 – (a) Imagem Original (636 x 651 pixels) ; (b) Magnitude do gradiente.

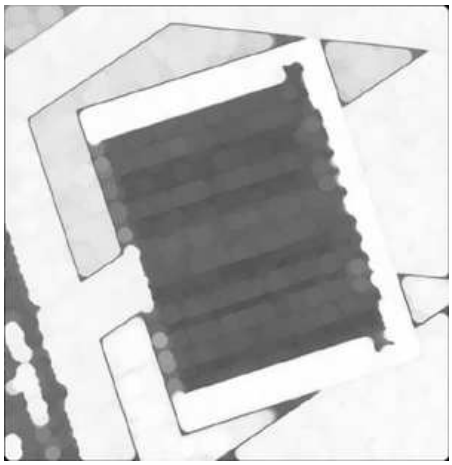


Figura 5.3 – Abertura da Figura 5.2 (b) por um disco de diâmetro 20 pixels

5.2.3 Localização Precisa do símbolo

A imagem obtida pelo processo de abertura (Figura 5.3) apresenta partes irrelevantes com baixos níveis de cinza que devem ser extraídas para proporcionar uma localização precisa do símbolo. Aplicamos inicialmente uma limiarização de Otsu para binarização da imagem, e assim obtemos uma imagem com diversos objetos, incluindo aquele referente ao local onde se encontra o símbolo do Código de Barras.

Para assegurar um enquadramento eficaz, que será aplicado posteriormente, é necessário que o símbolo esteja livre de cortes no seu corpo. Dessa forma, o símbolo não deve tocar as bordas da imagem, a fim de garantir a sua integridade. Assim, apenas serão considerados objetos que estejam posicionados no interior da imagem, sendo excluídos aqueles que tocam as bordas. A Figura 5.4 ilustra este passo.

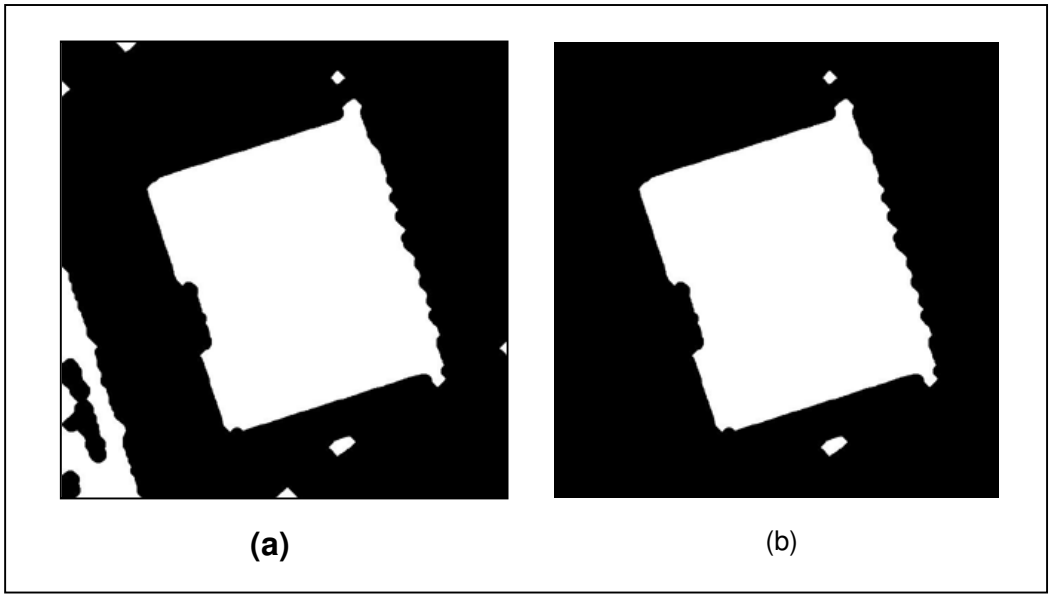


Figura 5.4 – (a) Binarização da Figura 5.3 ;
(b) Extração dos objetos que tocam as bordas da imagem

A partir da imagem mostrada na Figura 5.4 (b), devemos determinar o objeto que melhor indique a presença do símbolo. Aplicamos então uma abertura por área, cujo limiar

remova objetos irrelevantes de área pequena na imagem. O objeto remanescente desta abertura por área será aquele que indica a presença do símbolo.

5.2.4 Alinhamento do eixo Longitudinal

Focalizamos o posicionamento do objeto referente ao símbolo (Figura 5.5.a), e o enquadramos através de uma "janela imaginária" que o circunda mediante uma tolerância da ordem de 9 vezes a largura de módulo para a simbologia UPC, devido à presença das margens de silêncio. Caso a simbologia presente na imagem seja EAN-13, o valor desta tolerância deverá ser da ordem de 11 vezes o valor da largura de módulo, pois a largura da margem de silêncio esquerda nesta simbologia possui 11 vezes a largura de módulo. A fim de uma generalização, a adoção dos valores para a simbologia EAN-13 possibilita a decodificação dos códigos UPC-A, uma vez que estes códigos são versões específicas do EAN-13. Obtemos então uma imagem com dimensões reduzidas contendo o objeto do símbolo posicionado de forma centralizada (Figura 5.5.b).

Tomando a imagem da Figura 5.5.b como máscara, extraímos parcialmente a imagem original do símbolo (Figura 5.5.c) a fim de determinarmos de forma eficaz a inclinação do eixo longitudinal do símbolo.

Através da análise do histograma de ângulos do vetor gradiente, determinamos o ângulo preponderante, que indicará a inclinação das diversas barras que compõem o símbolo. Rotacionamos a imagem do enquadramento, colocando o eixo longitudinal do símbolo na posição horizontal.

Ocasionalmente, a rotação para o alinhamento pode posicionar o símbolo de cabeça para baixo, impedindo a leitura correta do código; entretanto, o dígito verificador (seção 2.3.5.1) certifica qual o sentido da leitura do símbolo, e uma rotação de 180º é realizada se necessário.

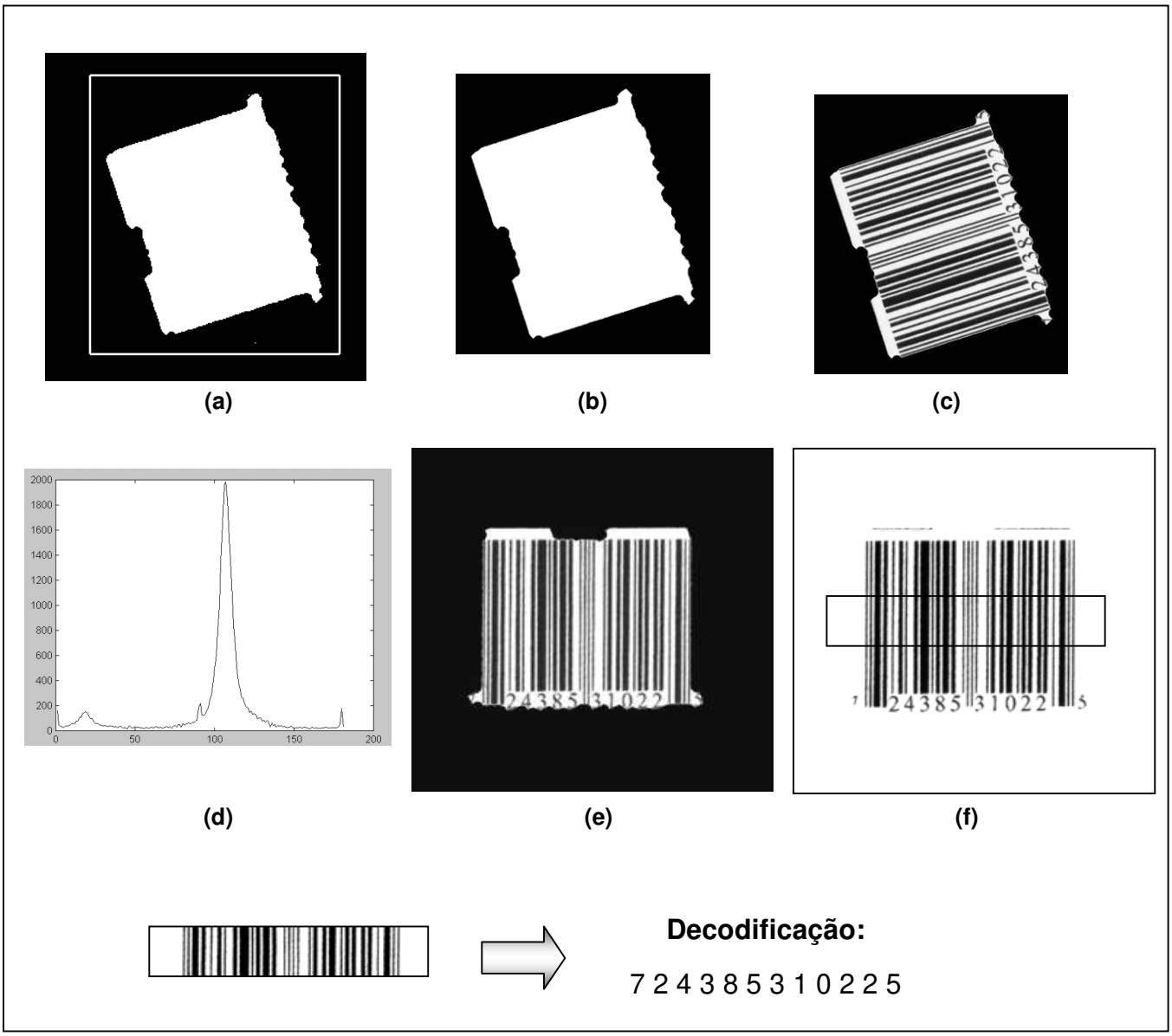


Figura 5.5 – a) Enquadramento por uma "janela imaginária"; b) Enquadramento preciso do objeto referente ao símbolo; c) Extração parcial da imagem original usando (b) como máscara; d) Histograma de Ângulos do Vetor Gradiente de (c); e) Alinhamento do eixo longitudinal; f) captura de uma faixa do símbolo.

Capítulo 6

Verificação

A fim de determinar a qualidade de um símbolo estudado, seja do ponto de vista de impressão ou de interpretação durante a análise da imagem capturada, calculamos alguns parâmetros que caracterizam o termo "verificação". Geralmente, um processo de verificação é realizado quando ocorre um erro na tentativa de decodificação do código. Quando a decodificação é realizada sem a detecção de erros, nenhum cálculo de parâmetros é feito. Os parâmetros de verificação seguem normas estabelecidas pelo ANSI (American National Standards Institute) e CEN (Comission for European Normalization).

Quanto maior a qualidade de impressão do símbolo, mais fácil e confiável serão a leitura e decodificação, respectivamente. Na nossa análise, a resolução de digitalização é um dos fatores mais importantes para uma decodificação livre de erros.

6.1 Parâmetros analisados na leitura

6.1.1 Deterioração do símbolo

Talvez, os maiores problemas encontrados no momento de leitura de um Código de Barras sejam os danos causados por arranhões na superfície de impressão ou mesmo a presença de defeitos que impossibilitam a leitura correta do código.

Ao longo do símbolo, podemos verificar muitas vezes, dois tipos de defeitos típicos: vazios, caracterizados por áreas claras internas às barras e manchas, caracterizadas por áreas escuras presentes nos espaçamentos. A Figura 6.1 ilustra os dois casos típicos de defeitos explicados.

Para o algoritmo de decodificação do código UPC/EAN, implementado no nosso trabalho, a faixa extraída (Figura 3.13) proporciona uma tolerância a esses tipos de defeitos, pois a

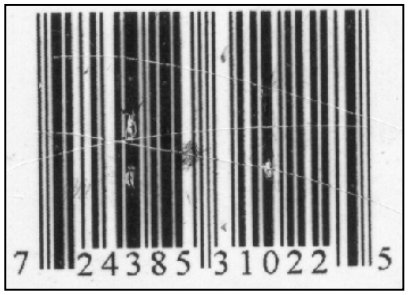


Figura 6.1 – Ilustração de defeitos típicos encontrados em um Código de Barras

cada varredura de linha da faixa, uma verificação da quantidade de elementos que compõem a simbologia é realizada, a fim de garantir uma confiabilidade de leitura.

6.1.2 Espalhamento de Tinta

Um outro problema bastante persistente na decodificação de Códigos de Barras é o efeito de espalhamento de tinta (Seção 2.2), que provoca impressões incorretas de barras e espaçamentos. Este espalhamento de tinta também é denominado ganho de ponto ou de impressão, que a princípio deve diferir de um valor muito pequeno para máquinas de impressão distintas, e além disso devem ser controláveis para qualquer processo de impressão.

O valor do ganho de ponto possui uma faixa de variação conforme seja o processo de impressão ou da máquina utilizada. Diversos fatores interferem na determinação do ganho de ponto, tais como a viscosidade da tinta, porosidade e absorção da superfície de impressão, pressão da máquina, velocidade de impressão etc. Alguns programas de impressão de

Códigos de Barras aplicam uma técnica de compensação de espalhamento de tinta, reduzindo a princípio o ganho de impressão.

Existem calibradores padrão destinados a ajustes de impressão, que consistem em gabaritos com quadrados compostos por linhas paralelas finas que se distanciam gradativamente para cada quadrado. Estes quadrados são impressos e uma vistoria é feita a fim de determinar se as impressões foram realizadas conforme regras pré-estabelecidas.

Uma forma de se garantir um efeito de espalhamento aceitável consiste na uniformidade de larguras de módulos encontrados para cada elemento varrido (seja barra ou espaço), isto é, para cada elemento com ***nx*** de largura (***n*** = 1,2,3,4 ; ***x*** = largura de módulo) os desvios padrão para as barras e espaços devem ser calculados e seus valores deverão ser reduzidos e diferirem entre si, de um valor muito pequeno para assegurar uma leitura correta, pois quanto menor for o valor do desvio padrão para barras ou espaços, maior será a uniformidade de cada elemento, e quanto menor a diferença entre as medianas das distribuições de larguras para barras e espaços, menor será o efeito de espalhamento de tinta. Este parâmetro é denominado gradual, pois não impede a realização correta da decodificação, apenas indica a intensidade de uma característica do símbolo impresso. Uma ilustração de análise destes parâmetros é apresentada na Figura 6.2.

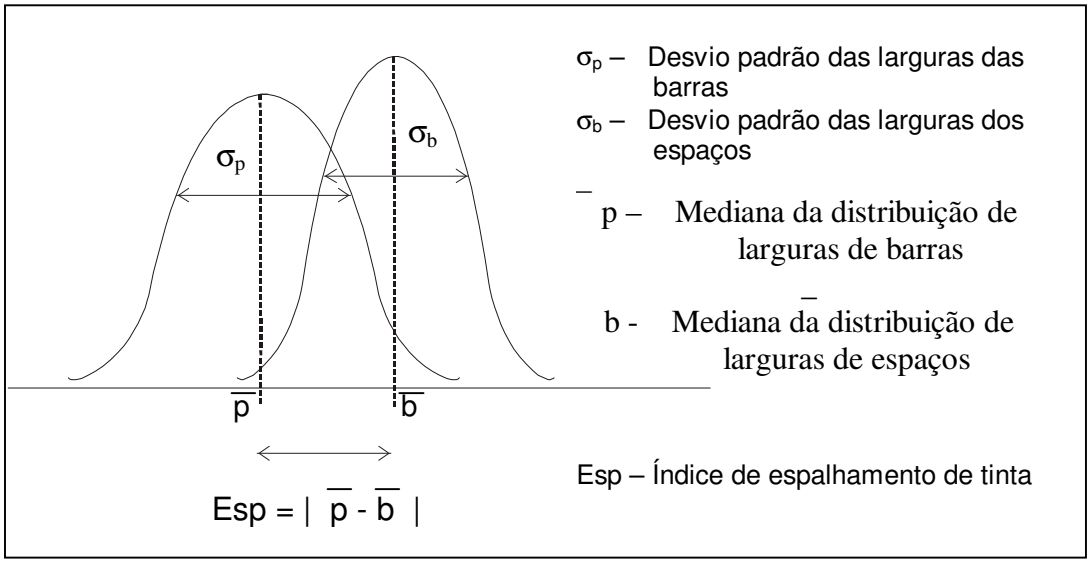


Figura 6.2 – Ilustração da análise do índice de espalhamento

6.1.3 Resolução de Digitalização

Diversos são os fatores que levam a uma decodificação incorreta ou até mesmo impossível de ser realizada. Inicialmente, a verificação da resolução na digitalização deve ser feita a fim de garantir uma tolerância eficaz na estimação da largura de um módulo, ou seja, da largura do mais estreito elemento presente no símbolo. Caso este valor de largura de módulo seja inferior a 4 pixels, um alerta deverá ser feito ou mesmo o impedimento de decodificação deverá ser realizado.

O parâmetro de resolução é denominado de permissão/impedimento, ou seja, conforme o resultado da análise das características avaliadas, a leitura será realizada ou não. Caso o resultado seja desfavorável à leitura, uma mensagem de erro deverá ser apresentada.

6.1.4 Distorção das barras

As superfícies mais vulneráveis a problemas de impressão e consequentemente de leitura são os plásticos, devido à possibilidade de deformações apresentadas em alguns casos. Atualmente, alguns fabricantes de produtos estão tentando solucionar este problema através do aumento da altura do símbolo, que facilita a leitura por meio de alguns leitores óticos.

A deformação da superfície de impressão pode causar a princípio, dois problemas na representação do símbolo: uma inclinação variável das barras e espaços, ou um aumento desordenado entre os elementos.

A inclinação dos elementos poderá ser imperceptível caso seja aplicada de forma uniformemente distribuída, ou seja, se ocorrer devido a uma força de cisalhamento, pois neste caso, a proporção entre as larguras poderá permanecer inalterada. As Figuras 6.3 e 6.4 exemplificam os casos citados acima.

Figura 6.3 – Exemplo de uma distorção por cisalhamento

Figura 6.4 – Exemplo de uma distorção desordenada

Para a simbologia UPC/EAN, os elementos podem apresentar 4 larguras possíveis, que são múltiplos da largura de um módulo. Assim, para cada um dos 4 múltiplos encontrados na varredura, as larguras nominais são avaliadas e por meio do cálculo do desvio padrão (tanto para as barras quanto para os espaços), determinamos a uniformidade das larguras, que representa a intensidade de deformação do símbolo. Quanto maior a uniformidade, menor será a distorção presente no símbolo.

6.1.5 Tolerância para margens de silêncio

Um parâmetro importante a ser analisado no processo de decodificação é a tolerância dada para as margens de silêncio dos símbolos.

Todas as simbologias de Códigos de Barras possuem os mesmos elementos básicos, embora muitos apresentem aspectos visuais diferentes. As simbologias apresentadas em nosso trabalho (Code 39, UPC e EAN) possuem os seguintes elementos comuns entre si:

- Margem de silêncio esquerda
- Caractere/Padrão de início
- Informação codificada (Dados)
- Caractere/Padrão de parada
- Margem de silêncio direita

Cada simbologia apresenta suas características próprias. As margens de silêncio do Código 39 são da ordem de 11 vezes a largura do elemento mais estreito, tanto no lado esquerdo como no direito. No código UPC, as margens de silêncio esquerda e direita são idênticas e devem possuir uma largura da ordem de 9 vezes a largura de um módulo (elemento mais estreito). Para o código EAN, as margens de silêncio esquerda e direita são estabelecidas na razão de 11 para 7 módulos. Muitas vezes a decodificação de um símbolo é impossível de ser realizada quando as tolerâncias estipuladas para as margens de silêncio não são seguidas.

Capítulo 7

Experimentos

Neste capítulo apresentamos o resultado de alguns experimentos através de ilustrações dos passos seguidos pelos algoritmos implementados.

Uma verificação de qualidade do símbolo capturado na digitalização foi feita por meio de um parâmetro que denominamos de "Taxa de Recuperação", que consiste na relação entre a altura da faixa extraída inicialmente e a altura da faixa resultante de sua filtragem (verificação da quantidade correta de elementos). Este parâmetro foi definido como a razão entre a altura da faixa filtrada e a altura da faixa inicialmente extraída, multiplicado por 100. Dessa forma, quanto maior a qualidade do símbolo impresso, maior será o valor da *Taxa de Recuperação*, pois as dimensões das alturas inicial e final aproximar-se-ão entre si.

Outros parâmetros de verificação são avaliados através dos desvios-padrão das larguras dos elementos (barras e espaços) encontrados em cada símbolo estudado.

7.1 Exemplo 1

Neste exemplo, a imagem de entrada possui 1170 x 1082 pixels, e ilustra a digitalização de dois símbolos presentes em produtos com embalagens plásticas.

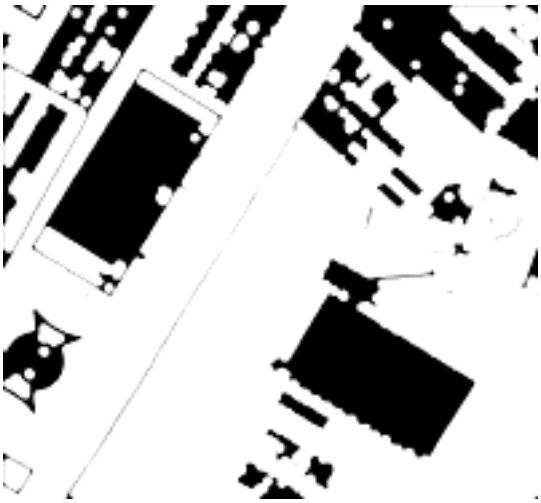
Notamos que, embora as digitalizações apresentem alguns reflexos devido ao material que compõe as embalagens, o *threshold* de Otsu mostrou-se como uma técnica bastante robusta para o processo de limiarização, proporcionando uma caracterização eficaz de ambos os símbolos.

Uma vez que a imagem de entrada possui uma quantidade considerável de pixels, o tempo de execução verificado foi longo.

Imagem Original

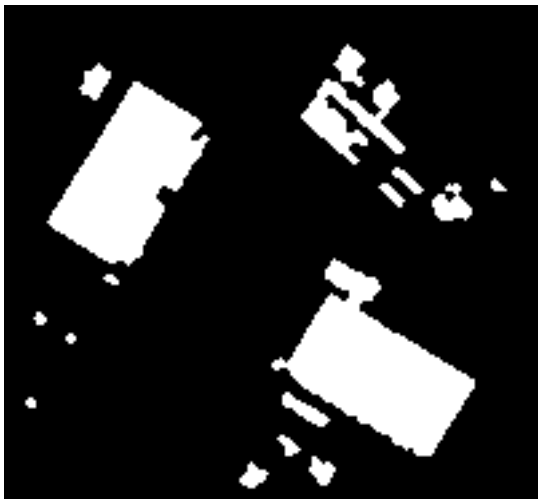
Imagem Inversa da Magnitude do Vetor Gradiente

Abertura Morfológica por um disco de diâmetro 20 pixels



Threshold de Otsu

Abertura Morfológica da inversa do Threshold de Otsu por um elemento cruz de raio 8



Remoção de componentes conexos que tocam as bordas

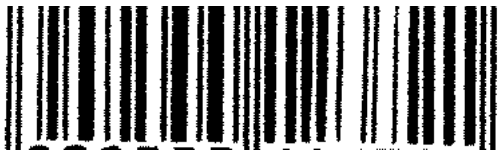
Localização aproximada dos elementos conexos correspondentes aos símbolos, por meio de uma análise de área

Primeiro Símbolo Extraído

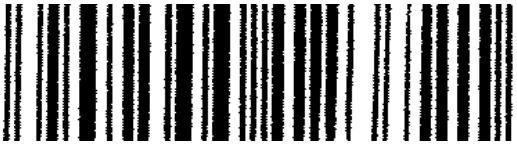
Alinhamento do eixo longitudinal
(θ =59º)

Enquadramento do símbolo

Threshold de Otsu



Faixa extraída inicialmente



Faixa final filtrada

Taxa de Recuperação: 91,67 %

Decodificação: 4 9 0 2 5 5 5 1 2 3 7 2 1

Segundo símbolo extraído

Alinhamento do eixo longitudinal
(θ =150º)

Enquadramento do símbolo

Threshold de Otsu

Faixa extraída inicialmente

Faixa final filtrada

Taxa de Recuperação: 88,51 %

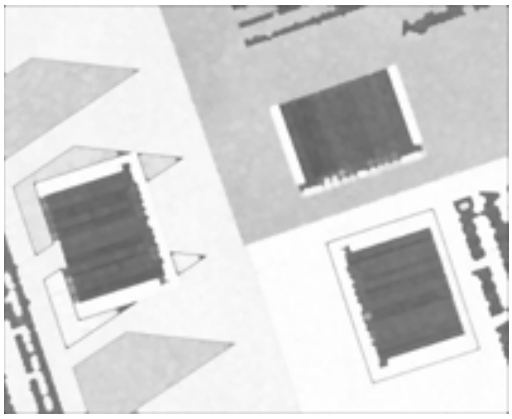
Decodificação: 4 9 0 3 3 3 3 0 1 7 8 7 4

7.2 Exemplo 2

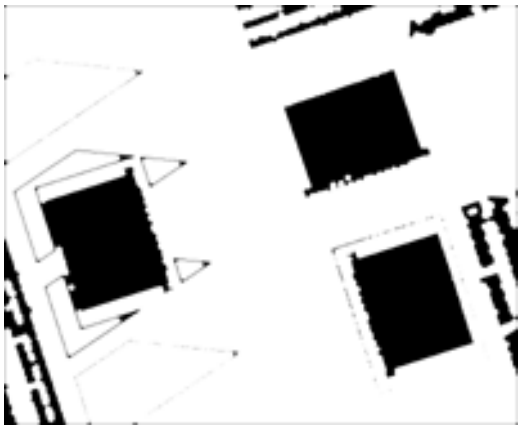
Neste exemplo, a imagem de entrada contém 3 símbolos, representados em 1700 x 1383 pixels.

Imagem Original

Imagem Inversa da Magnitude
do Vetor Gradiente

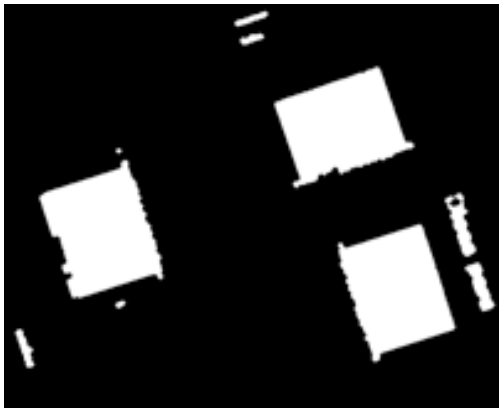


Abertura Morfológica por um
disco de diâmetro 20 pixels

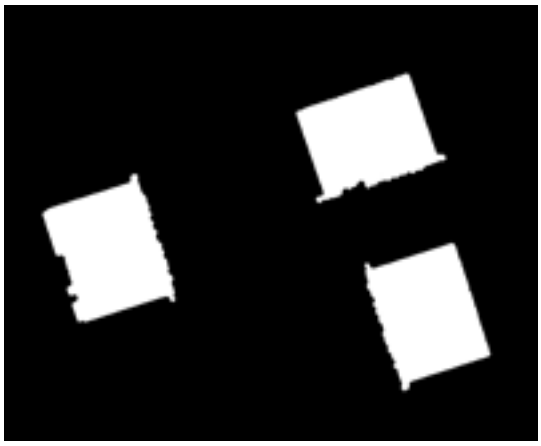


Threshold de Otsu

Abertura Morfológica da inversa do Threshold de Otsu por um elemento cruz de raio 8



Remoção de componentes conexos que tocam as bordas



Determinação dos elementos conexos

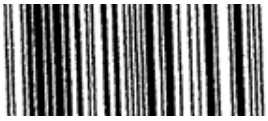
Primeiro símbolo capturado (θ=108º)

Faixa Final extraída

Taxa de recuperação: 98,81%

Decodificação: 724385310225

Segundo símbolo capturado
(θ =17º)



Faixa Final extraída
Taxa de recuperação: 98,74%
Decodificação:
977141447530

Terceiro símbolo capturado
(θ =108º)

Faixa Final extraída
Taxa de recuperação: 99,36%
Decodificação:
729180128792

7.3 Exemplo 3

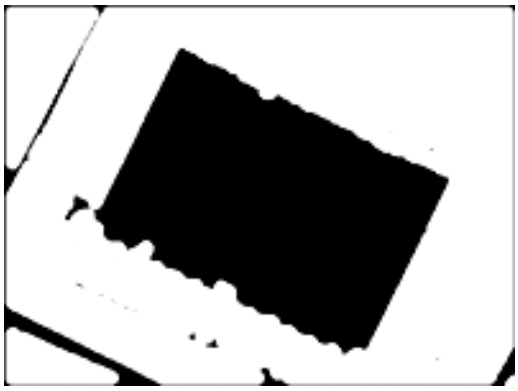
Para um caso onde ocorre manchas no corpo do símbolo, a imagem resultante do processo de limiarização de Otsu apresenta-se bastante ruidosa, necessitando de uma *filtragem adicional** para a decodificação correta. Esta filtragem consistiu em uma abertura morfológica por um elemento estruturante linha na posição vertical, conforme visto na imagem subseqüente àquela que mostra a primeira faixa extraída.

Neste terceiro exemplo, a imagem de entrada possui 700 x 521 pixels, contendo um símbolo.

Imagem Original

Imagem Inversa da Magnitude
do Vetor Gradiente

Abertura Morfológica por um disco
de diâmetro 20 pixels



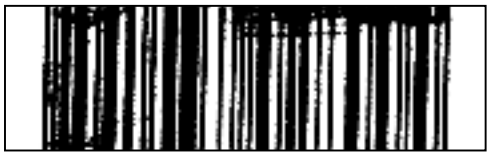
Threshold de Otsu

Enquadramento preciso e Captura
do símbolo

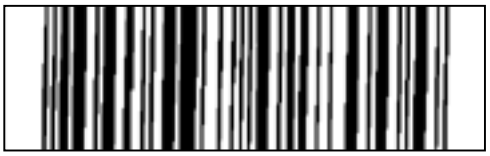
Alinhamento do eixo longitudinal
(θ =155º)

Extração do símbolo

Threshold de Otsu



Extração da Primeira faixa



*Filtragem adicional** da primeira faixa

Taxa de recuperação após a *filtragem adicional**: 97,91%
Decodificação:
7 8 9 1 0 3 8 0 2 8 0 0 4

Capítulo 8

Conclusões

Nosso trabalho descreve uma abordagem para decodificação de códigos de barras unidimensionais, usando análise de imagens.

Ao longo do desenvolvimento de nossa dissertação, elaboramos algoritmos destinados às etapas de decodificação, tais como captura de um símbolo presente em uma imagem, análise de validação do símbolo, alinhamento do eixo longitudinal do símbolo, verificação de características relevantes do símbolo, e determinação do dado codificado. Os algoritmos foram desenvolvidos em linguagem MATLAB®, utilizando diversas ferramentas de morfologia matemática presentes na *SDC Morphology Toolbox*[†] .

As imagens de estudo foram digitalizadas em um leitor ótico de mesa convencional, em resolução ótica de 300 dpi, garantiu uma largura de módulo para cada simbologia suficiente para uma leitura que proporcionasse a decodificação sem erros para um símbolo bem impresso.

Por ser um protótipo experimental, nosso objetivo não focou tempo de processamento competitivo aos dispositivos comuns de leitores de códigos. O tempo de processamento para uma imagem de 636 × 651 pixels, com o símbolo em uma posição aleatória, foi da ordem de 1 minuto e vinte segundos em um PC Pentium® MMX 233 MHz. Notamos que as imagens analisadas deveriam possuir dimensões elevadas, pois somente assim garantiam uma estimativa de no mínimo 3 pixels por largura de módulo, com isso foi exigido um tempo de processamento elevado, principalmente nas operações de abertura por área, cujo valor de limiar exigido foi alto para localização precisa do símbolo.

[†] www.mmorph.com

Outro “gargalo” do processo de decodificação foi encontrado no momento de decodificação das simbologias UPC/EAN, uma vez que exigiu etapas de comparações a diversas tabelas e cálculo do dígito verificador.

Como ponto de destaque em nossa proposta, citamos:

- Operações simples de análise de imagens e
- Aplicabilidade em processos de decodificação na indústria de automação comercial, utilizando câmeras CCD de boa resolução.

O ponto deficiente está centralizado no tempo de processamento exigido pelo algoritmo, decorrente da linguagem de programação e sobretudo do tamanho das imagens capturadas.

Como sugestão de trabalho de continuidade, destacamos a possibilidade de migração do algoritmo para uma outra linguagem que permita um tempo de processamento reduzido.

Referências

[1] Louis J. Galbiati, Jr. , *Machine Vision and Digital Image Processing Fundamentals*, Englewood Cliffs: Prentice Hall, 1990.

[2] Fábio Grossmann e Mauro Luiz Zyngier, *Código de barras: da teoria à prática,* São Paulo : Nobel, 1991.

[3] T. Pavlidis, J. Swartz, and Y. P. Wang, "Fundamentals of Bar Code Information Theory", *IEEE Computer,* pp. 74-86, Apr. 1990.

[4] Zuguang Cai, "A New Decode Algorithm for Binary Bar Codes", *Pattern Recognition Letters,* 15 , 1994, pp.1191-1199.

[5] J. S. Chen and G. Medioni, "Detection, localization and estimation of edges", *IEEE Trans. Pattern. Anal. Machine Intell.*, vol.11, pp. 191-198, 1989.

[6] E. Joseph and T. Pavlidis, "Bar Code Waveform Recognition Using Peak Locations", *IEEE Trans. Pattern. Anal. Machine Intell.*, vol 16, nº 6, June, 1994.

[7] Vera Lúcia Pinheiro da Silva, *Aplicações Práticas do Código de Barras,* São Paulo: Nobel, 1989.

[8] Roberto A. Lotufo, "Inspeção Visual Automática (Uma Abordagem computacional através de estudos de casos) – Leitor de Código de Barras ", *IA-330 Tópicos em Computação, 1º semestre 1991,* Notas de Aula

[9] N. Otsu, "A Threshold Selection Method from grey-level histograms". *IEEE Trans. Systems Man. Cybernet.,* 9, 62-66, 1979

[10] R. C. Gonzalez and R. E. Moods, "*Digital Image Processing*" , Addison-Wesley Publishing Company, Inc.

[11] S. Beucher and F. Meyer, "The Morphological Approach to Segmentation: The Watershed Transformation", *In E. R. Dougherty, editor, Mathematical Morphology in Image Processing*, 433-481. Marcel Dekker, New York City, New York, 1993.

[12] J. Goutsias and S. Batman, "Morphological Methods for Biomedical Image Analysis", *Handbook of Medical Imaging: Vol.3 – Progress in Image Processing and Analysis*. SPIE Optical Engineering Press, 1999.

[13] Zuguang Cai, "A New Decode Algorithm for Binary Bar Codes*", Pattern Recognition Letters,* 15 , 1191-1199, December 1994.

[14] E. Joseph and T. Pavlidis, "Peak Classifier for bar code waveforms", *SUNY at Stony Brook,* 0-8186-2915-0/92 © 1992 IEEE.

[15] T. Pavlidis , J. Swartz, and Ynjiun P. Wang, "Information Encoding with Two-Dimensional Bar Codes", *IEEE Computer,* pp. 18-28, June. 1992

Apêndice A

Tabela de Bandeiras para o Código EAN

00-13	Estados Unidos e Canadá
20-29	Reservado ao uso local do estabelecimento comercial
30 -37	França
400-440	Alemanha
45	Japão
46	Federação Russa
471	Taiwan
474	Estônia
475	Latvia
477	Lituânia
479	Sri Lanka
480	Filipinas
482	Ucrânia
484	Moldávia
485	Armênia
486	Georgia
487	Casaquistão
489	Hong Kong
49	Japão
50	Reino Unido
520	Grécia
528	Líbano
529	Chipre
531	Macedônia
535	Malta
539	Irlanda
54	Bélgica & Luxemburgo
560	Portugal
569	Islândia
57	Dinamarca
590	Polônia
594	Romênia
599	Hungria
600-601	África do Sul

609	Maurício
611	Marrocos
613	Argélia
619	Tunísia
622	Egito
625	Jordan
626	Irã
64	Finlândia
690-692	China
70	Noruega
729	Israel
73	Suécia
740-745	Guatemala, El Salvador, Honduras, Nicarágua, Costa Rica e Panamá
746	República Dominicana
750	México
759	Venezuela
76	Suíça
770	Colômbia
773	Uruguai
775	Perú
777	Bolívia
779	Argentina
780	Chile
784	Paraguai
785	Perú
786	Equador
789	Brasil
80 -83	Itália
84	Espanha
850	Cuba
858	Eslováquia
859	Thecoslováquia
860	Iugoslávia

869	Turquia
87	Países Baixos
880	Coréia do Sul
885	Tailândia
888	Cingapura
890	Índia
893	Vietnã
899	Indonésia
90 -91	Áustria
93	Austrália
94	Nova Zelândia
955	Malásia
977	ISSN (International Standard Serial Number)
978	ISBN (International Standard Book Number)
979	ISMN (International Standard Music Number)
980	Recibos de reembolso
99	Cupons

Apêndice B

Algoritmos implementados

Decodificação da simbologia EAN / UPC :

Captura do símbolo e decodificação:

```
warning off;
mmfreedom(2);

f=mmreadgray('nome_do_arquivo_da_imagem');

original=f;
[fmg,ft,ht]=sobel(f,50);
faux=mmneg(fmg);

f1=mmopen(faux,mmsedisk(10));

ot=iaotsu(f1);
f1o=(f1>ot);

f2o=mmopen(mmneg(f1o),mmsecross(8));

f2o2=mmneg(f2o);

hole1=mmclohole(f2o2);

blobsimbol=mmsubm(hole1,f2o2);

simbol=double(mmareaopen(blobsimbol,39000));

f7=mmlabel(simbol);
total=double(max(max(f7)));

   if total==0
   close all;
   disp('                                                              ');
   disp('--------------------------------------------------------------');
   disp('                       ATENÇÃO !!!                            ');
   disp('A digitalização da imagem que contém o símbolo foi realizada');
   disp('   em uma resolução inferior a 300 pontos por polegada.      ');
   disp('                                                              ');
   disp('  Realize uma nova digitalização, aumentando a resolução.    ');
   disp('                                                              ');
   disp('        ((( NÃO FOI POSSÍVEL REALIZAR A LEITURA )))           ');
   disp('--------------------------------------------------------------');
else
```

```
for n=1:total

   f7a=(f7==n);
   box=double(mmblob(f7a,'boundingbox','data'));

[fh fw]=size(f);
if (box(1,1)-30)<0 |(box(1,3)+30)>fh | (box(1,2)-30)<0 | (box(1,4)+30)>fw
   disp('                                                          ');
   disp('----------------------------------------------------------');
   disp('                       ATENÇÃO !!!                        ');
   disp(' O símbolo encontra-se próximo às bordas da imagem.  ');
   disp('                                                          ');
   disp('           Realize uma nova digitalização.           ');
   disp('                                                          ');
   disp('    ((( NÃO FOI POSSÍVEL DECODIFICAR O SÍMBOLO )))   ');
   disp('----------------------------------------------------------');
else


img=f(box(1,1)-30:(double(box(1,3))+30),box(1,2)-
30:(double(box(1,4))+30));

mask=simbol(box(1,1)-30:(double(box(1,3))+30),box(1,2)-
30:(double(box(1,4))+30));

place=double(img).*double(mask);

% INCLINAÇÃO DO CÓDIGO (Sobel) :

 [fmag, ftheta, htheta]=sobel(place,50);

% Extração de picos indesejados no histograma de Sobel:
h=mmopen(htheta,mmsecross(6));

% Máximo pico:
hm=max(h(:));

% Determinação Pontual do máximo pico no histograma:
h1=mmthreshad(h,hm,hm);

% Intersecção entre o ponto de máximo e a curva do histograma:
hh=mmintersec(mmgray(h1,'uint16'),htheta);

% Localização do ângulo cujo valor de máximo no histograma é verificado:
[hm theta]=max(hh(:));

%Alinhamento da máscara final:
align_mask=imrotate(mask,-theta);

%Imagem complemento da máscara
align_mask_comp=mmneg(align_mask);
```

```
% Alinhamento do símbolo:
alinhamento1=imrotate(place,-theta);


alinhamento2=mmthreshad(alinhamento1,1,255);
alinhamento=uint8(mmsymdif(alinhamento1,mmgray(alinhamento2,
'uint8',255)));
alinhamento=mmneg(alinhamento);

imwrite(alinhamento,'resultado8.jpg','jpg');

% Rotulação da Máscara Final
f8=mmlabel(align_mask);

box2=double(mmblob(f8,'boundingbox','data'));
hb2=(box2(1,3)-box2(1,1))+1;
wb2=(box2(1,4)-box2(1,2))+1;

% Plano de fundo branco:
scaux=255*(ones(hb2,wb2+66));
scaux(:,34:end-
33)=alinhamento(box2(1,1):(double(box2(1,3))),box2(1,2):(double(box2(1,4))
));
img2=scaux;


% Captura de uma faixa central ao corpo do símbolo:
[hmk wmk]=size(align_mask);
auxbox=align_mask(fix(4*hmk/9):fix(5*hmk/9),:);
figure;mmshow(auxbox);
box3=double(mmblob(auxbox,'boundingbox','data'));


imgextra=imrotate(img,-theta);
imgextra2=imgextra(box2(1,1):box2(1,3),:); % Parte da imagem original
[hx wx]=size(imgextra2);
stripeaux=imgextra2(fix(4*hx/9):fix(5*hx/9),:);

stot=iaotsu(stripeaux);
stoto=(stripeaux>stot);

colaux1=stripeaux(:,box3(1,2)-44:box3(1,2)); % Fatia da BG esquerda
colaux2=stripeaux(:,box3(1,4):box3(1,4)+44); % Fatia da BG direita
colaux1=colaux1>iaotsu(colaux1);
colaux2=colaux2>iaotsu(colaux2);

qualidade_margem_e=(1-(sum(sum(abs(double(colaux1)-
1)))/(size(colaux1,1)*(size(colaux1,2)))))*100
qualidade_margem_d=(1-(sum(sum(abs(double(colaux2)-
1)))/(size(colaux2,1)*(size(colaux2,2)))))*100
```

```
%*********************************************************************
%                          DECODIFICAÇÃO
%*********************************************************************
[codigo,inconsistencia]=upcia4(img2);

if inconsistencia==0
   dado=codigo'
else
   img2=imrotate(img2,180);
   [codigo,inconsistencia]=upcia4(img2);
   if inconsistencia==0
      dado=codigo'
   else
      disp('
');
      disp('-----------------------------------------------------');
      disp('                       ATENÇÃO !!!                   ');
      disp('     Ocorreram problemas na detecção do símbolo      ');
      disp('                                                     ');
      disp('           Realize uma nova digitalização.           ');
      disp('                                                     ');
      disp('    ((( NÃO FOI POSSÍVEL DECODIFICAR O SÍMBOLO )))   ');
      disp('-----------------------------------------------------');
      error(' ');
   end


end
end
end
end

-------------------------


function [codigo,inconsistencia,faixa]=upcia4(imagem)

mmfreedom(2);
% Inicialização dos vetores:
codigo=[];
erro=0;
v13=[];
d13=[];
desvio=[];
black1=[];
black2=[];
acumblack1=[];
acumblack2=[];
white1=[];
white2=[];
acumwhite1=[];
acumwhite2=[];
```

```
f=imagem;

[y x]=size(f);

[t,eta]=iaotsu(f);  %   THRESHOLD DE OTSU
f=(f>t);            %  IMAGEM LIMIARIZADA

faixa=f(round((1/4)*(y)):round((3/4)*(y)),:);

[yf xf]=size(faixa);

nlabel=61;
nlabel2=61;
lfaixa=[];

%<<<<<<  Filtragem Adicional* PARA SÍMBOLOS COM MANCHAS  >>>>>>>
%fn=mmneg(faixa);
%se1=ones((yf/2),2);
%se2=mmimg2se(se1); % Elemento estruturante
%fno=mmopen(fn,se2);% Filtragem para remoção de junções
                    % entre barras (elementos pretos).
%<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

for lyf=1:yf
   lfaixa2=mmlabelflat(faixa(lyf,:));   % Rotulação por "mmlabelflat"
   nlabel=double(max(max(lfaixa2)));    % Quantidade de rótulos
                                        % do empilhamento


   if nlabel==61 & nlabel2==61
      lfaixa=[lfaixa;faixa(lyf,:)];
      lfaixa3=mmlabelflat(lfaixa);
      nlabel2=double(max(max(lfaixa3)));

   else
   end
end

[ylf xlf]=size(lfaixa);

figure;mmshow(lfaixa);truesize;

if size(lfaixa,1)~=1

   lfaixa=lfaixa(1:end-1,:);

[hf lf]=size(lfaixa);

qualidade=(hf/yf)*100

llfaixa=mmlabelflat(lfaixa);         % Rotulação

hlfaixa=mmhistogram(llfaixa);        % Histograma da rotulação
```

```
dist2=[];
largura=(double(hlfaixa)/(hf))';     % Larguras em ponto flutuante

[yd xd]=size(largura);

for i=1:xd
   dist2=[dist2,repmat(rem(i,2),1,round(largura(1,i)))];
end

linha=(dist2==0);

lflinha=mmlabelflat(linha);           % Rotulação por "mmlabelflat"

hlflinha=mmhistogram(lflinha);        % Histograma da rotulação

hhlflinha=mmhistogram(hlflinha);      % Função distribuição

[dmax imax]= max(double(hhlflinha));  %[valor máx    nº do rótulo]

disp('                                                ');
disp('----------------------------------------        ');
fprintf('Um módulo tem aproximadamente: %d pixels\n', imax);
disp('----------------------------------------        ');

aolinha=mmareaopen(linha, 6 * imax);  % Abertura p/ extrair zona
                                      % de silêncio
k=max(max(mmlabelflat(aolinha)));

if k~=3
   disp('                                            ');
   disp('           (((( ATENÇÃO: ))))               ');
   disp('                                            ');
   disp('Margem de silêncio fora da especificação    ');
   disp('        ou Threshold inadequado             ');
   disp('--------------------------------------      ');
   error('Extração da zona de silêncio com problemas.');
else

localbar=mmneg(aolinha);              % Local das barras

linha2=linha(localbar);

lbl12=mmlabelflat(linha2);

k2=max(max(mmlabelflat(linha2)));
if k2~=59
   disp('                                              ');
   disp('ATENÇÃO:                                      ');
   disp('--------                                      ');
   disp('Quantidade de barras fora da especificação !!!');
   disp('       O Código pode estar danificado.        ');
   error('                                              ');

else
```

```
%t1  t2   c1  c2  c3  c4
mb=...
[4   4     1   2   2   2  ;
 3   3     2   2   1   2  ;
 5   5     1   1   4   1  ;
 2   4     2   3   1   1  ;
 3   5     1   3   2   1  ;
 2   2     4   1   1   1  ;
 4   4     2   1   3   1  ;
 3   3     3   1   2   1  ;
 4   2     2   1   1   3  ;
 5   3     1   1   2   3  ];

%t1  t2   c1  c2  c3  c4   t1  t2
mc=...
[3   4     2   2   2   1   4   4  ;
 4   3     2   1   2   2   3   3  ;
 2   5     1   4   1   1   5   5  ;
 5   4     1   1   3   2   2   4  ;
 4   5     1   2   3   1   3   5  ;
 5   2     1   1   1   4   2   2  ;
 3   4     1   3   1   2   4   4  ;
 4   3     1   2   1   3   3   3  ;
 3   2     3   1   1   2   4   2  ;
 2   3     3   2   1   1   5   3  ];


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                            CAMPO ESQUERDO
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

largura=largura(3:end-1);

   for n=4:4:24

      C1=double(largura(1,n));
      C2=double(largura(1,n+1));
      C3=double(largura(1,n+2));
      C4=double(largura(1,n+3));

      t1u=C3+C4;
      t1e=C3+C4;
      t2=C2+C3;

      fator=(C1+C2+C3+C4)/7;   % módulo estimado para o dígito

      rC1=C1/fator;     % Módulos ocupados pelo 1º Espaço
      rC2=C2/fator;     % Módulos ocupados pela 1ª Barra
      rC3=C3/fator;     % Módulos ocupados pelo 2º Espaço
      rC4=C4/fator;     % Módulos ocupados pela 2ª Barra
```

```
      rt1=rC3+rC4;
      rt2=rC2+rC3;

      rcsrate=(rC1+rC3)/(rC2+rC4);

      vc=[rt1 rt2 rC1 rC2 rC3 rC4];

%_____________________________________________________________________

      % 1ª Classificação

      dist1=zeros(10,1);
      dist2=zeros(10,1);
      for i=1:10
         dist1(i)=sum((abs((mc(i,1:end-2)-vc))).^2); %Tabela A
         dist2(i)=sum((abs((mb(i,1:end)-vc))).^2);   %Tabela B
      end
      minA=min(min(dist1));
      minB=min(min(dist2));
      if minA<minB
         v13=[v13 1];
         ls=find(dist1==minA);
         troubles1=[4/3 2/5];   % (c1+c3)/(c2+c4)
      else
         v13=[v13 double(0)];
         ls=find(dist2==minB);
         troubles1=[3/4 5/2];   % (c1+c3)/(c2+c4)
      end

      if size(ls)==1 % Certificação de que não há 2 distâncias iguais

      if ls==1|ls==2|ls==7|ls==8

      % 2ª Classificação

      dif=((abs((troubles1-rcsrate))).^2);

      ls2=find(dif==min(min(dif)));

      if size(ls2)==1 % Certificação de que não há 2 distâncias iguais

%---------Problemas da simbologia--------

      if ls==1|ls==7

         if ls2==1
            codigo=[codigo 1];
         else
            codigo=[codigo 7];
         end
      else

      if ls==2|ls==8
```

```
         if ls2==1
            codigo=[codigo 2];
         else
            codigo=[codigo 8];
         end
      end
      end
      else  % Caso ls2 seja 2
         erro=1;
         disp('A leitura apresentou redundâncias de decodificação');
         disp('              na banda esquerda !!!');
      end
   else
      codigo=[codigo ls];
   end

   else  % Certificação que há 2 distâncias iguais no 2° teste
      erro=1;
      disp('A leitura apresentou redundâncias de decodificação');
      disp('              na banda esquerda !!!');
   end
   end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                          CAMPO DIREITO
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   for n=33:4:53

      C1=double(largura(1,n));
      C2=double(largura(1,n+1));
      C3=double(largura(1,n+2));
      C4=double(largura(1,n+3));

      fator=(C1+C2+C3+C4)/7;   % módulo estimado para o dígito

      rC1=C1/fator;     % Módulos ocupados pela 1ª Barra
      rC2=C2/fator;     % Módulos ocupados pelo 1° Espaço
      rC3=C3/fator;     % Módulos ocupados pela 2ª Barra
      rC4=C4/fator;     % Módulos ocupados pelo 2° Espaço

      rt1=rC1+rC2;
      rt2=rC2+rC3;

      rcsrate=(rC1+rC3)/(rC2+rC4);

      vc=[rC1 rC2 rC3 rC4 rt1 rt2];

%_____________________________________________________________________

      % 1ª Classificação

      dist=zeros(10,1);
      for i=1:10
```

```
         dist(i)=sum((abs((mc(i,3:end)-vc))).^2);
      end
      ls=find(dist==min(min(dist)));

      if size(ls)==1  % Garantia de 1 distância mínima apenas

         if ls==1|ls==2|ls==7|ls==8   % Problemas da simbologia

      % 2ª Classificação

      troubles1=[4/3 2/5];

      dif=((abs((troubles1-rcsrate))).^2);

      ls2=find(dif==min(min(dif)));

      if size(ls2)==1 % Certificação de que não há 2 distâncias iguais

%---------Problemas da simbologia--------

      if ls==1|ls==7
         if ls2==1
            codigo=[codigo 1];
         else
            codigo=[codigo 7];
         end
      else
      if ls==2|ls==8

         if ls2==1
            codigo=[codigo 2];
         else
            codigo=[codigo 8];
         end
      end
      end
      else  % Caso ls2 seja 2
         erro=1;
         disp('A leitura apresentou redundâncias de decodificação');
         disp('              na banda direita !!!');
      end
   else % Caso os números detetados não forem 1, 2, 7 ou 8
      codigo=[codigo,ls];
   end
   else  % Certificação que há 2 distâncias iguais no 2° teste
      erro=1;
      disp('A leitura apresentou redundâncias de decodificação');
      disp('              na banda direita !!!');
   end
   end
end
end
if erro~=1
   codigo=codigo;
```

```
else
   codigo=[];
end

else  % Caso a quantidade de linhas de lfaixa seja realmente " 1 "
   codigo=[];
   disp('                                                              ');
   disp('¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤    ATENÇÃO !!!  ¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤');
   disp('                                                              ');
   disp('  A digitalização do código está excessivamente ruidosa !!!  ');
   disp('                                                              ');
   disp('¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤');
end

m13=...
  [1 1 0 1 0 0;...
   1 1 0 0 1 0;...
   1 1 0 0 0 1;...
   1 0 1 1 0 0;...
   1 0 0 1 1 0;...
   1 0 0 0 1 1;...
   1 0 1 0 1 0;...
   1 0 1 0 0 1;...
   1 0 0 1 0 1;...
   1 1 1 1 1 1];

rmv13=repmat(v13,10,1);
dif13=mmcmp(uint8(rmv13),'==',uint8(m13));
d13=find(sum(dif13,2)==6);

if isempty(d13)
   inconsistencia=1
   disp('    Problemas no símbolo     ');
   disp('FALHA NA TENTATIVA DE LEITURA');
else

codigo=[d13 codigo];

%    Dígito Verificador:
di=codigo(1,1)+codigo(1,3)+codigo(1,5)+codigo(1,7)+codigo(1,9)+codigo(1,11
);
dp=codigo(1,2)+codigo(1,4)+codigo(1,6)+codigo(1,8)+codigo(1,10)+codigo(1,1
2);
vd1=(dp*3)+di;
vd2=fix(vd1/10);
vd=rem(((vd2+1)*10),vd1);

if double(codigo(1,13))==vd
   inconsistencia=0
else
   inconsistencia=1
end

%*****************************
```

```
codigoaux=codigo(1,2:end);
for b=1:6
   dig=double(codigoaux(1,b));
   largaux=largura(1,4*b:(4*b)+3);
   tab=double(v13(b));
   if tab==1        % Uso da tabela A
      acum=zeros(4,4);
      acumblack1=zeros(4,4);
      acumwhite1=zeros(4,4);
      matriz_a=[mc(dig,3:end-2)];

      for z=1:4
         acum(matriz_a(1,z),z)=largaux(1,z);
         desvio=[desvio acum];
         if rem(z,2)==0
            acumblack1(matriz_a(1,z),z)=largaux(1,z);
            black1=[black1 acumblack1];
         else
            acumwhite1(matriz_a(1,z),z)=largaux(1,z);
            white1=[white1 acumwhite1];
         end
      end
   else             % Uso da tabela B

      acum=zeros(4,4);
      matriz_b=[mb(dig,3:end)];
      for z=1:4
         acum(matriz_b(1,z),z)=largaux(1,z);
         desvio=[desvio acum];
         if rem(z,2)==0
            acumblack1(matriz_b(1,z),z)=largaux(1,z);
            black1=[black1 acumblack1];
         else
            acumwhite1(matriz_b(1,z),z)=largaux(1,z);
            white1=[white1 acumwhite1];
         end
      end
   end
end

for b=7:12
   dig=double(codigoaux(1,b));
   largaux=largura(1,(4*b)+5:(4*b)+8);
   acum=zeros(4,4);
   matriz_a=[mc(dig,3:end-2)];
   acumblack2=zeros(4,4);
   acumwhite2=zeros(4,4);
      for z=1:4
         acum(matriz_a(1,z),z)=largaux(1,z);
         desvio=[desvio acum];
         if rem(z,2)==1
            acumblack2(matriz_a(1,z),z)=largaux(1,z);
            black2=[black2 acumblack2];
         else
```

```
            acumwhite2(matriz_a(1,z),z)=largaux(1,z);
            white2=[white2 acumwhite2];
         end
      end
end

guarda_aux=[largura(1,1:3) largura(1,28:32) largura(1,57:59)];
guarda=zeros(4,11);
guarda(1,1:11)=guarda_aux;

guardablack1=[largura(1,1:2:3) largura(1,29:2:31) largura(1,57:2:59)];
barblack1=zeros(4,6);
barblack1(1,1:6)=guardablack1;
black=[black1 black2 barblack1];
black=black';
bk1=black(find((black(:,1)~=0)),1);
bk2=black(find((black(:,2)~=0)),2);
bk3=black(find((black(:,3)~=0)),3);
bk4=black(find((black(:,4)~=0)),4);

std_black=[std(bk1,0,1) std(bk2,0,1) std(bk3,0,1) std(bk4,0,1)]

guardawhite1=[largura(1,2) largura(1,28:2:32) largura(1,58)];
barwhite1=zeros(4,5);
barwhite1(1,1:5)=guardawhite1;
white=[white1 white2 barwhite1];
white=white';
wt1=white(find((white(:,1)~=0)),1);
wt2=white(find((white(:,2)~=0)),2);
wt3=white(find((white(:,3)~=0)),3);
wt4=white(find((white(:,4)~=0)),4);
std_white=[std(wt1,0,1) std(wt2,0,1) std(wt3,0,1) std(wt4,0,1)]

desvio=[desvio guarda];
desv=desvio';

l1=desv(find((desv(:,1)~=0)),1);
l2=desv(find((desv(:,2)~=0)),2);
l3=desv(find((desv(:,3)~=0)),3);
l4=desv(find((desv(:,4)~=0)),4);

std_distorcao=[std(l1,0,1) std(l2,0,1) std(l3,0,1) std(l4,0,1)]

nlarg1=size(find(desvio(1,:)),2);
nlarg2=size(find(desvio(2,:)),2);
nlarg3=size(find(desvio(3,:)),2);
nlarg4=size(find(desvio(4,:)),2);

media1=sum(desvio(1,:))/nlarg1;
media2=sum(desvio(2,:))/nlarg2;
media3=sum(desvio(3,:))/nlarg3;
media4=sum(desvio(4,:))/nlarg4;

medias=[1 media1 ; 2 media2 ; 3 media3 ;4 media4]
```

```
  %*******************************
  occur10=find(codigo==10);
  codigo(occur10)=0;

  end % Final "isempty(d13)"
```

Decodificação da simbologia Code 39:

```
imagem='nome_do_arquivo_da_imagem';

f=mmreadgray(imagem);  % Leitura da imagem
[linha,padrao]=leitura(f);
[decode]=decod(padrao)

-------------------------

  function [linha,padrao]=leitura(f)
  mmfreedom(2);
  padrao=[1];
  [y x]=size(f);

  b=f>(iaotsu(f));              % Threshold

  linha=[b(fix(y/3),:)];        % Linha de análise

  lflinha=mmlabelflat(linha);

  mxlf=double(max(max(lflinha)));
  sile=double(find(lflinha==1));
  sild=double(find(lflinha==mxlf));
  mxs=double(max(sile)+1);
  mns=double(min(sild)-1);

  area=mmhistogram(lflinha);
  h=mmhistogram(area);
  [dmax imax]= max(double(h));
  fprintf('Estimativa da barra fina : %d pixels\n', imax);

  aolinha=mmareaopen(linha, 5 * imax);
  laolinha=mmlabelflat(aolinha);
  mxll=max(max(laolinha));
  if mxll~=3
     error('O símbolo está intensamente danificado !!!');
  else

  [yl xl]=size(linha);

  rotbarras=(double(lflinha(1,(mxs):(mns)))-1);

  mxr=max(uint8(rotbarras));
```

```
   h=mmhistogram(rotbarras);
   h=double(h);
   h=[h;1];
   haux=h;

   [hh wh]=size(h);

   for n=2:10:hh
      h(n:n+9)=sort(h(n:n+9));
      mnh=min(h(n+7:n+9));
         for i=n:n+9
         valuebar=abs(rem(i,2));
         if haux(i)>=mnh
            padrao=[padrao repmat(valuebar,1,3)];
         else
            padrao=[padrao valuebar];
         end
      end
   end

   end
   [hp wp]=size(padrao);
   padrao=padrao(1,17:wp-17);


-------------------------

function [decode]=decod(t)
k=[t];
decode=[];
erro=0;
cp=['1234567890abcdefghijklmnopqrstuvwxyz-. $/+%'];


m=[ 1 0 0 0 1 0 1 1 1 0 1 0 1 0 0 0;
1 0 1 0 0 0 1 1 1 0 1 0 1 0 0 0;
1 0 0 0 1 0 0 0 1 1 1 0 1 0 1 0;
1 0 1 0 1 1 1 0 0 0 1 0 1 0 0 0;
1 0 0 0 1 0 1 1 1 0 0 0 1 0 1 0;
1 0 1 0 0 0 1 1 1 0 0 0 1 0 1 0;
1 0 1 0 1 1 1 0 1 0 0 0 1 0 0 0;
1 0 0 0 1 0 1 1 1 0 1 0 0 0 1 0;
1 0 1 0 0 0 1 1 1 0 1 0 0 0 1 0;
1 0 1 0 1 1 1 0 0 0 1 0 0 0 1 0;
1 0 0 0 1 0 1 0 1 1 1 0 1 0 0 0;
1 0 0 0 1 0 1 0 1 1 1 0 1 0 0 0;
1 0 0 0 1 0 0 0 1 0 1 1 1 0 1 0;
1 0 1 0 1 0 0 0 1 1 1 0 1 0 0 0;
1 0 0 0 1 0 1 0 0 0 1 1 1 0 1 0;
1 0 1 0 0 0 1 0 0 0 1 1 1 0 1 0;
1 0 1 0 1 0 1 1 1 0 0 0 1 0 0 0;
1 0 0 0 1 0 1 0 1 1 1 0 0 0 1 0;
1 0 1 0 0 0 1 0 1 1 1 0 0 0 1 0;
```

```
1 0 1 0 1 0 0 0 1 1 1 0 0 0 1 0;
1 0 0 0 1 0 1 0 1 0 1 1 1 0 0 0;
1 0 1 0 0 0 1 0 1 0 1 1 1 0 0 0;
1 0 0 0 1 0 0 0 1 0 1 0 1 1 1 0;
1 0 1 0 1 0 0 0 1 0 1 1 1 0 0 0;
1 0 0 0 1 0 1 0 0 0 1 0 1 1 1 0;
1 0 1 0 0 0 1 0 0 0 1 0 1 1 1 0;
1 0 1 0 1 0 1 0 0 0 1 1 1 0 0 0;
1 0 0 0 1 0 1 0 1 0 0 0 1 1 1 0;
1 0 1 0 0 0 1 0 1 0 0 0 1 1 1 0;
1 0 1 0 1 0 0 0 1 0 0 0 1 1 1 0;
1 0 0 0 1 1 1 0 1 0 1 0 1 0 0 0;
1 0 1 1 1 0 0 0 1 0 1 0 1 0 0 0;
1 0 0 0 1 1 1 0 0 0 1 0 1 0 1 0;
1 0 1 1 1 0 1 0 0 0 1 0 1 0 0 0;
1 0 0 0 1 1 1 0 1 0 0 0 1 0 1 0;
1 0 1 1 1 0 0 0 1 0 0 0 1 0 1 0;
1 0 1 1 1 0 1 0 1 0 0 0 1 0 0 0;
1 0 0 0 1 1 1 0 1 0 1 0 0 0 1 0;
1 0 1 1 1 0 0 0 1 0 1 0 0 0 1 0;
1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 0;
1 0 1 1 1 0 1 1 1 0 1 0 1 1 1 0;
1 0 1 1 1 0 1 0 1 1 1 0 1 1 1 0;
1 0 1 0 1 1 1 0 1 1 1 0 1 1 1 0];

for i=1:16:size(k,2)



comp=mmcmp(uint16(double(m)),'==',uint16(double(repmat(k(1,i:i+15),43,1))));
   [x]=find(sum(comp,2)==16);

   if any(sum(comp,2)==16)
      decode=[decode cp(x(1),1)];
   else
      erro=erro+1;
      invalido=k(1,i:i+15);
      disp([' ']);
      disp(['                Padrao Inválido < 'int2str(invalido) ' >']);
      disp([' ']);

   end
end


if ~erro
  disp([' ']);
      disp(['                Decodificacao < 'decode ' >']);
      disp([' ']);
end
```