Algoritmos de afinamento de imagens 3-d: uma análise teórica e prática

Francisco Nivando Bezerra

Dissertação de Mestrado

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECA DO IMECC DA UNICAMP

Bezerra, Francisco Nivando

B469a Algoritmos de afinamento 3-d : uma análise teórica e prática / Francisco Nivando Bezerra -- Campinas, [S.P. :s.n.], 1997.

Orientador : Neucimar Jerônimo Leite

Dissertação (mestrado) - Universidade Estadual de Campinas, Instituto de Computação.

1. Processamento digital de imagens. I. Leite, Neucimar Jerônimo. II. Universidade Estadual de Campinas. Instituto de Computação. III. Título.

Tese de Mestrado defendida e aprovada em 11 de abril de 1997 pela Banca Examinadora composta pelos Professores Doutores

Prof. Dr. João Marques de Carvalho

Prof. Dr. João Meidanis

Prof. Dr. Neucimar Jerônimo Leite

Algoritmos de afinamento de imagens 3-d: uma análise teórica e prática

Este exemplar corresponde à redação final da Dissertação devidamente corrigida e defendida por Francisco Nivando Bezerra e aprovada pela Banca Examinadora.

Campinas, 28 de abril de 1997.

Neucimar Jerônimo Leite, Doutor
(Orientador)

Dissertação apresentada ao Instituto de Computação, UNICAMP, como requisito parcial para a obtenção do título de Mestre em Ciência da Computação.

Instituto de Computação
Universidade Estadual de Campinas

Algoritmos de afinamento de imagens 3-d: uma análise teórica e prática

Francisco Nivando Bezerra[1]

Abril de 1997

Banca Examinadora:

- Neucimar Jerônimo Leite, Doutor (Orientador)
- João Marques de Carvalho, Phd
- João Meidanis, Phd
- Paulo Lício de Geus, Phd

[1]Este trabalho teve apoio financeiro das seguintes instituições: FAPESP, CAPES e CNPq.

Resumo

O afinamento de imagens tem sido bastante estudado em Processamento Digital de Imagens. O caso de imagens bidimensionais, extensivamente considerado na literatura, apresenta um conjunto significativo de algoritmos com diferentes métodos e caracterizações. O mesmo não ocorre com imagens em três dimensões em que as limitações tecnológicas de alguns anos atrás refletem, ainda, a pouca abordagem neste sentido.

Esta dissertação apresenta uma análise comparativa dos algoritmos de afinamento de imagens tridimensionais. São discutidos problemas tais como performance, características geométricas das imagens resultantes, e apresentadas algumas equivalências entre os principais métodos. Algumas extensões de técnicas de processamento de imagens bidimensionais ao caso tridimensional são consideradas. Estas técnicas incluem eliminação de ruídos nas imagens afinadas, e definição de divisores de água de uma função. São apresentadas, ainda, técnicas de otimização para os algoritmos estudados.

Abstract

The problem of thinning binary images has been extensively studied in the digital image processing field. Particularly to the bidimensional images, there is a large number of methods leading to different results. In the three-dimensional case, due to the technological limitations, the number of such algorithms is rather limited.

This work discusses some problems concerned with the definition of thinning algorithms to three-dimensional images. It describes the main algorithms presented in the literature and analyses them in a comparative way. We consider different aspects of the thinning problem such as execution time, geometric characteristics, and define formally some equivalences between the methods. We also consider some extensions from the bidimensional image processing algorithms to the three-dimensional case. These extensions include noise reduction of thinned images and the watersheds of a function. Finally, some general optimization techniques are defined for the implementation of the algorithms.

Poeta, cantô da rua
Que na cidade nasceu,
Cante a cidade que é sua,
Que eu canto o sertão que é meu.
⋮

Você teve inducação,
Aprendeu munta ciença,
Mas das coisa do sertão
Não tem boa experiênça.
Nunca fez uma paioça,
Nunca trabaiou na roça,
Não pode conhecer bem,
Pois nesta penosa vida,
Só quem provou da comida
Sabe o gosto que ela tem.
⋮

Mas porém, eu não invejo
O grande tesôro seu,
Os livro do seu colejo,
Onde você aprendeu.
Pra gente aqui sê poeta
E fazê rima compreta,
Não precisa professô;
Basta vê no mês de maio,
Um poema em cada gaio
E um verso em cada fulô.

Patativa do Assaré.

Agradecimentos

Agradeço ao Professor Neucimar Jerônimo Leite pela confiança na minha capacidade de trabalho e pela valiosa orientação.

Agradeço ao Professor Gilles Bertrand da École Supérieure d'Ingénieurs en Électrotechnique et Életronique, Paris, França, que contribuiu para este trabalho fornecendo material bibliográfico e imagens digitais usadas em nossos experimentos.

Agradeço aos amigos Marcos André, Maria do Socorro, Delano, Elder, Ralph, Gisele, Aminadab, Jerônimo e Cleidson pela amizade. Dispenso especial gratidão à amiga Cristiane Reis pelo apoio e cumplicidade em muitas das minhas opiniões.

Agradeço à minha família e, especialmente, a meus pais que souberam me incentivar sempre.

Conteúdo

Capítulo 1

Introdução

1.1 Breve histórico

Um grande número de algoritmos de afinamento para imagens bidimensionais (2-d) tem sido desenvolvido para extrair características da forma dos objetos em análise de imagens. Estes algoritmos reduzem os objetos de uma imagem a um conjunto de linhas representativo da estrutura inicial. Em [1], podemos encontrar um *survey* sobre algoritmos de afinamento 2-d.

Apesar do grande número de algoritmos de afinamento para o caso 2-d, poucos trabalhos foram realizados para o caso tridimensional (3-d). Alguns motivos concorreram para tal fato, entre eles a pouca disponibilidade de dados digitais, até alguns anos atrás, sob a forma de imagens 3-d . Atualmente, dispomos de imagens 3-d geradas a partir de várias tecnologias: de tomografia computadorizada (CT), de ressonância magnética (MRI), de computação gráfica dentre outros. A inexistência deste tipo de dado implicava numa ausência de demanda por aplicações associadas. Na verdade, aplicações para o afinamento 3-d não são ainda tão variadas quanto no caso 2-d, que vão desde o estudo biológico de células até o reconhecimento de caracteres. Uma das aplicações típicas do afinamento 3-d é a análise de falhas de forja e fundição (ver [2]). Todavia, a principal dificuldade imposta no surgimento de algoritmos de afinamento 3-d refere-se ao fato de que as propriedades topológicas dos espaço digital 3-d são de tratamento bem mais difícil que as do caso 2-d. Atualmente, existem diversos trabalhos sobre conexidade, números de Euler, preservação de topologia, pontos simples e outros tópicos relacionados ao afinamento 3-d (ver [3, 4, 5, 6, 7, 8], por exemplo). Os resultados apresentados nestes trabalhos servem como ferramentas importantes para realização de estudos, análises e sínteses desse tipo de algoritmo.

1.2 Objetivos

O presente trabalho realiza um estudo sobre os principais algoritmos de afinamento 3-d com ênfase nos seguintes aspectos:

i. estudo de topologia digital e do problema de preservação de topologia em algoritmos de afinamento, considerando as diversas soluções encontradas na literatura;

ii. avaliação das características geométricas resultantes dos diversos algoritmos de afinamento;

iii. análise comparativa da performance destes algoritmos;

iv. estudo de tempo de execução, visando um melhor desempenho dos algoritmos sobre imagens de grandes dimensões;

v. extensão de técnicas de Processamento Digital de Imagens 2-d para o caso 3-d.

Será dada ênfase, nesta dissertação, aos trabalhos que apresentam algoritmos de afinamento completos e aspectos formais bem fundamentados.

1.3 Organização do texto

No capítulo 2, será feita uma breve apresentação de topologia digital, ferramenta fundamental para o estudo de algoritmos de afinamento, focalizando pontos de interesse deste estudo. Serão definidas, entre outros, relações de adjacência, imagens binárias, números de Euler.

A seguir, o capítulo 3 apresenta uma discussão sobre algoritmos de afinamento propriamente ditos. Na seção 3.1, são levantados vários aspectos relevantes ao estudo desses algoritmos (principalmente os relacionados à topologia digital, introduzida no capítulo 2). Nas seções seguintes, são descritos os principais métodos de afinamento 3-d apresentados na literatura.

O capítulo 4 contém uma análise comparativa dos algoritmos na qual são analisados aspectos topológicos, características geométricas e performance.

Finalmente, o capítulo 5 considera alguns problemas de processamento digital de imagens 2-d e suas extensões ao caso 3-d. O capítulo 6 apresenta as conclusões finais do trabalho.

Para facilitar a leitura desta dissertação, o apêndice A apresenta, de forma resumida, as notações usadas em todos os capítulos, de forma esquemática e de fácil consulta.

Capítulo 2

Princípios de Topologia Digital

Neste capítulo, apresentaremos aspectos teóricos da topologia digital, essenciais no estudo de algoritmos de afinamento apresentado nos capítulos seguintes. Abordaremos as topologias dos espaços bidimensional e tridimensional, sendo este último o alvo maior de nosso interesse.

As definições e conceitos de topologia digital são construídos, geralmente, sobre a definição de relações de adjacência entre pontos. Essas relações são apresentadas na seção 2.1. Na seção 2.2, aparecem os conceitos de imagem binária digital, componente, cavidade, túnel e outros relacionados a estes. A seção 2.3 discute o número de Euler de uma imagem, que é, como veremos a seguir, uma importante ferramenta na preservação da topologia durante o afinamento.

2.1 Relações de adjacência

Diversas grades e relações de adjacência têm sido consideradas na busca de melhores algoritmos para o processamento digital de imagens. A maioria das relações de adjacência consideradas baseia-se nas *vizinhanças de Voronoi* dos pontos da grade. A vizinhança de Voronoi de um ponto p em uma grade bidimensional (tridimensional) é o conjunto de pontos no plano Euclidiano (no espaço Euclidiano) que estão pelo menos tão próximos de p quanto de qualquer outro ponto da grade [6].

As *relações de adjacência de Voronoi* são as relações entre dois pontos cujas vizinhanças de Voronoi compartilham um vértice, compartilham uma aresta e, no caso 3-d, compartilham uma face. Uma relação de adjacência de Voronoi em que cada ponto da grade é adjacente a exatamente outros α pontos da grade é dita uma *relação de α-adjacência.*

Existem, na literatura, diversas grades: e.g. 2-d hexagonal, 2-d triangular, 3-d triangular, 3-d *face-centered cubic*, 3-d *body-centered cubic* (ver [6]). Limitaremo-nos à descrição daquelas mais comumente usadas nos algoritmos de afinamento, que são a grade 2-d quadrada e a 3-d cúbica.

Grade 2-d quadrada: um ponto de grade p possui coordenadas $(p_1, p_2) \in \mathbb{Z}^2$. A vizinhança Voronoi de cada ponto, nesse tipo de grade, é um quadrado unitário. As relações de

adjacência Voronoi são a 4-adjacência e a 8-adjacência. Se as vizinhanças Voronoi de dois pontos p e q compartilham uma aresta, então p e q são 4-adjacentes. Quando as vizinhanças Voronoi de p e q compartilham um vértice, p e q são 8-adjacentes. A figura 2.1 mostra os pontos 4-adjacentes e 8-adjacentes ao ponto central da grade.

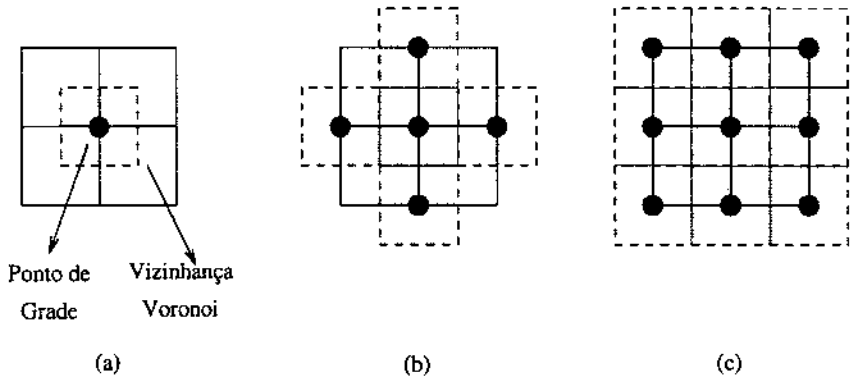


Figura 2.1: *Ponto de grade e sua vizinhança Voronoi (a), pontos 4-adjacentes a um ponto de grade e suas vizinhanças Voronoi (b), pontos 8-adjacentes a um ponto de grade e suas vizinhanças Voronoi (c).*

Grade 3-d cúbica: um ponto de grade p possui coordenadas $(p_1, p_2, p_3) \in \mathbb{Z}^3$. A vizinhança Voronoi de cada ponto é um cubo unitário. As relações de adjacência Voronoi são a 26-adjacência (quando as vizinhanças de Voronoi de dois pontos compartilham um vértice), a 18-adjacência (quando as vizinhanças de Voronoi de dois pontos compartilham uma aresta) e a 6-adjacência (quando as vizinhanças de Voronoi de dois pontos compartilham uma face). A figura 2.2 mostra as vizinhanças Voronoi para os pontos 26-adjacentes, 18-adjacentes e 6-adjacentes ao ponto central de uma grade, enquanto a figura 2.3 mostra os pontos representados na grade.

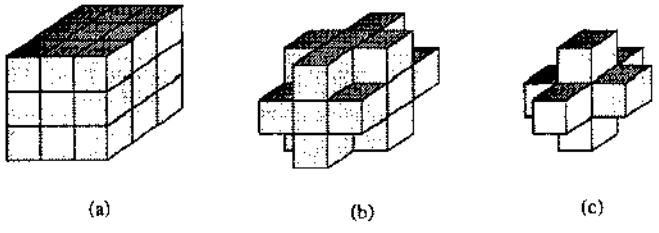
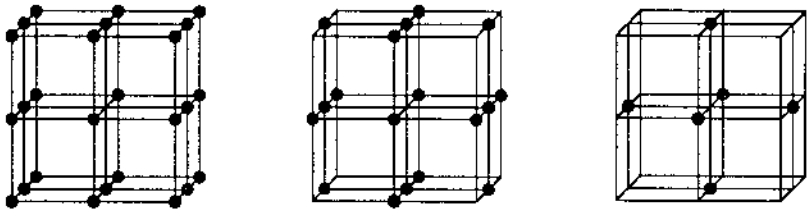


Figura 2.2: *Vizinhanças de Voronoi para os pontos 26-adjacentes (a), 18-adjacentes (b) e 6-adjacentes (c) a um mesmo ponto.*

As relações de adjacência descritas acima podem ser definidas de diversas maneiras. Vejamos uma destas formas alternativas.

Sejam dois pontos $p = (p_1, p_2)$ e $q = (q_1, q_2) \in \mathbb{Z}^2$. Dizemos que:



(a) (b) (c)

Figura 2.3: *Pontos 26-adjacentes (a), 18-adjacentes (b) e 6-adjacentes (c) ao ponto central da grade.*

i. se $|p_i - q_i| \leq 1$, para toda coordenada $i \in \{1, 2\}$, então p e q são *8-adjacentes*;

ii. se p e q são 8-adjacentes e diferem no máximo em uma coordenada, então p e q são *4-adjacentes*.

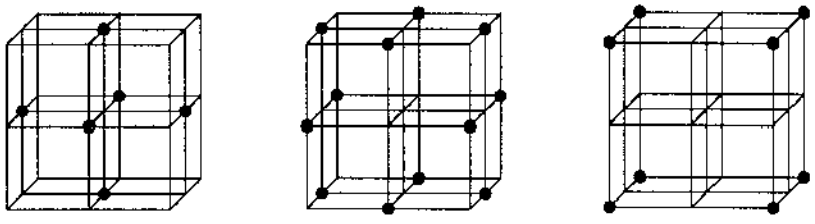
Sejam dois pontos $p = (p_1, p_2, p_3)$ e $q = (q_1, q_2, q_3) \in \mathbb{Z}^3$. Dizemos que:

i. se $|p_i - q_i| \leq 1$, para toda coordenada $i \in \{1, \ldots, 3\}$, então p e q são *26-adjacentes*;

ii. se p e q são 26-adjacentes e diferem no máximo em duas coordenadas, então p e q são *18-adjacentes*;

iii. se p e q são 26-adjacentes e diferem no máximo em uma coordenada, então p e q são *6-adjacentes*.

O conjunto de pontos α-adjacentes a um ponto p é chamado α-adjacência de p e denotado por $N^*_\alpha(p)$, sendo $\alpha \in \{6, 18, 26\}$ para o caso 3-d e $\alpha \in \{4, 8\}$ para o caso 2-d. Definimos $N_\alpha(p) = N^*_\alpha(p) \cup \{p\}$. Chamamos os conjuntos de pontos $N^*_6(p)$, $N^*_{18}(p) - N^*_6(p)$ e $N^*_{26}(p) - N^*_{18}(p)$ respectivamente de 6-vizinhos, 18-vizinhos e 26-vizinhos de p (figura 2.4).



(a) (b) (c)

Figura 2.4: *6-vizinhos (a), 18-vizinhos (b) e 26-vizinhos (c) do ponto central da grade.*

2.2 Imagens binárias digitais

Uma *imagem binária digital* (ou *imagem digital*, ou simplesmente *imagem*) $\Im$ é uma quádrupla $(\mathbb{Z}^n, \beta, \omega, B)$ na qual cada elemento em $\mathbb{Z}^n$ é chamado de um ponto (também chamado *voxel*

em $\mathbb{Z}^3$ e *pixel* em $\mathbb{Z}^2$) de $\Im$ e a ele está associado o valor 1 (um *ponto preto*) ou o valor 0 (um *ponto branco*); β e ω são relações de adjacência entre pontos e determinam as adjacências de dois pontos de maneira que dois pontos pretos são *adjacentes* se são β-adjacentes, e dois pontos brancos, ou um ponto branco e um ponto preto são *adjacentes* se são ω-adjacentes; $B \subseteq \mathbb{Z}^n$ é o conjunto de pontos pretos de $\Im$ e representa a *forma* ou os objetos da imagem. Os pontos brancos representam o *fundo* ou *background* da imagem. $\Im$ é também chamada uma *imagem n-dimensional* (β, ω). Consideraremos, no presente trabalho, apenas imagens bidimensionais e tridimensionais, ou seja, $n \in \{2, 3\}$.

Um *espaço digital de imagens* $\mathcal{E} = (\mathbb{Z}^n, \beta, \omega)$ é o conjunto formado por todas as imagens da quádrupla $(\mathbb{Z}^n, \beta, \omega, B)$ para qualquer $B \subset \mathbb{Z}^n$.

A *remoção* de um ponto p é a transformação que muda o valor de p de 1 (preto) para 0 (branco). Seja X um conjunto de pontos de uma imagem $\Im$; então, $\Im \cap X$ denota a imagem obtida de $\Im$ pela remoção de todos os pontos pretos em $\Im - X$, isto é, $\Im \cap X = (\mathbb{Z}^n, \beta, \omega, B \cap X)$. Seja Y um conjunto de pontos pretos de $\Im$; então, $\Im - Y$ denota a imagem obtida de $\Im$ pela remoção de todos os pontos pretos em Y, isto é, $\Im - Y = (\mathbb{Z}^n, \beta, \omega, B - Y)$. A *imagem complementar* de $\Im$ é a imagem $\overline{\Im} = (\mathbb{Z}^n, \omega, \beta, \mathbb{Z}^n - B)$.

Um *caminho* de X é uma seqüência $< x_1, ..., x_k >$ tal que cada $x_i \in X$ e x_j é adjacente a x_{j+1}, $1 \leq j < k$. Um *caminho preto* de X é um caminho constituído de pontos pretos de X. Se existe um caminho preto $< p = x_1, ..., x_k = q >$ entre os pontos p e q de X, estes são ditos *conectados*. As classes de equivalência desta relação são chamadas *componentes pretas de* X, i.e., uma componente preta é um subconjunto conectado maximal do conjunto de pontos pretos de X; uma *componente branca* de X é definida analogamente. Uma *cavidade* em uma imagem 3-d é uma componente branca finita. Por exemplo, uma esfera cujo interior é removido possui apenas uma cavidade.

Se um ponto preto p é adjacente a pelo menos uma componente branca, dizemos que p é um *ponto de borda*. Dizemos que um ponto preto p é um *ponto interior* se p não é adjacente à qualquer componente branca.

Além de componentes pretas e cavidades, túneis constituem estruturas topológicas do espaço 3-d. A definição precisa de túnel, todavia, não foi apresentada formalmente até agora. Podemos apenas dar uma idéia intuitiva. Por exemplo, uma esfera não possui túneis, enquanto um toro possui um túnel. A figura 2.5 apresenta exemplos de estruturas com túnel no espaço 3-d contínuo e no espaço 3-d discreto (grade 3-d cúbica). A figura 2.5(b) apresenta um túnel considerando a imagem (26, 6), (18, 6), (6, 26) ou (6, 18). Por outro lado, a figura 2.5(c) contém um túnel somente quando consideramos a imagem (26, 6) ou (18, 6).

Para um conjunto de pontos X, notaremos o número de componentes de X, o número de cavidades de X e o número de túneis de X, respectivamente por $C(X)$, $\mathcal{H}(X)$ e $T(X)$.

Um aspecto importante do trabalho com imagens binárias digitais é a escolha das relações de adjacência β e ω. Tal escolha deve ser feita cuidadosamente para se evitar certos paradoxos (ver [5, 6]). Vejamos alguns exemplos. Usando 6-adjacência para os pontos pretos e brancos ($\beta = \omega = 6$) da imagem da figura 2.3(c), teremos uma imagem em que não existem pontos pretos conectados, todavia, existem duas componentes brancas. Temos, com isso, pontos pretos

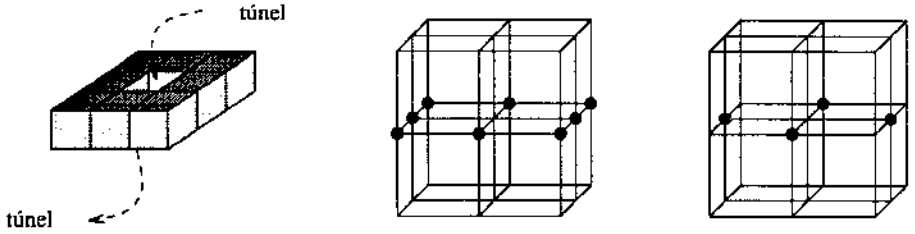


Figura 2.5: *Exemplo de túnel no espaço contínuo (a) e no espaço discreto da grade 3-d cúbica (b-c).*

não-conectados dividindo pontos brancos em duas componentes, o que é um paradoxo. Usando 18- ou 26-adjacência para os mesmos pontos, os pontos pretos formarão uma esfera de raio 1 cujo interior foi removido. No entanto, todos os pontos brancos da imagem estão conectados, formando uma única componente branca, o que é absurdo. Para evitar tais paradoxos, devemos usar 6-adjacência para os pontos pretos ($\beta = 6$) e 18- ou 26-adjacência para os pontos brancos ($\omega = 18$ ou $\omega = 26$), ou vice-versa [5, 6]. Temos, portanto, quatro possibilidades para o par (β, ω): (6, 18), (6, 26), (18, 6) e (26, 6).

2.3 Número de Euler *(Euler characteristc)*

2.3.1 Número de Euler no espaço contínuo

Um subconjunto S do plano ou do espaço 3-d é dito um *conjunto poliedral* se S pode ser expresso por uma união finita de pontos, segmentos de reta, triângulos fechados e tetraedros fechados.

O *número de Euler* de um conjunto poliedral finito S no espaço $\mathbb{R}^n$ pode ser definido de várias formas [5, 6]. Uma delas é a forma axiomática a seguir:

Axioma 2.1 $\chi(\emptyset) = 0$;

Axioma 2.2 $\chi(S) = 1$ *se* S *é não-vazio e convexo;*

Axioma 2.3 *para quaisquer poliedros* R *e* S, $\chi(R \cup S) = \chi(R) + \chi(S) - \chi(R \cap S)$.

Pode-se provar que estes axiomas são consistentes [5].

Uma definição alternativa é mostrada a seguir. Se S é um conjunto poliedral no plano $\mathbb{R}^2$, então $\chi(S)$ é igual ao número de componentes de S menos o número de cavidades de S. Se S é um conjunto poliedral no espaço $\mathbb{R}^3$, então $\chi(S)$ é igual ao número de componentes de S mais o número de cavidades de S menos o número de túneis de S (tabela 2.1). Por exemplo, uma circunferência vazia no plano $\mathbb{R}^2$ possui número de Euler igual a zero pois apresenta uma única componente e uma única cavidade. Já um cubo vazio no espaço possui número de Euler igual a 2 pois apresenta uma componente e uma cavidade, mas não possui túnel.

Definição 1 (axiomática)	$\chi(\emptyset) = 0$;
	$\chi(S) = 1$ para S não vazio e convexo;
	$\chi(R \cup S) = \chi(R) + \chi(S) - \chi(R \cap S)$
Definição 2 (plano)	$\chi(S) = C(S) - \mathcal{H}(S)$
Definição 2 (espaço)	$\chi(S) = C(S) + \mathcal{H}(S) - T(S)$

Tabela 2.1: *Algumas definições para o número de Euler.*

O número de Euler é uma invariante topológica e constitui uma valiosa ferramenta para realizar operações que exijam a preservação de topologia. Para maiores detalhes sobre as propriedades do número de Euler, consultar trabalhos sobre topologia algébrica, e.g., [9].

2.3.2 Análogo contínuo (*continuous analog*)

O número de Euler, como foi definido até agora, não pode ser usado na preservação de topologia de imagens dititais, que é uma das características de algoritmos de afinamento (como veremos oportunamente). A definição de número de Euler, como foi mostrada acima, é aplicável a poliedros no espaço contínuo, enquanto as imagens digitais encontram-se num espaço discreto.

Introduziremos uma quantidade que é definida para uma imagem digital $\Im$ e é análoga ao número de Euler e denotada $\chi(\Im)$. Para isso, associamos a cada imagem digital $\Im$ um poliedro $C(\Im)$, que denominaremos *análogo contínuo de* $\Im$. Definimos, então, $\chi(\Im) = \chi(C(\Im))$.

O análogo contínuo deve ter algumas propriedades, a saber:

i. o conjunto de pontos de grade em cada componente de $C(\Im)$ deve ser uma componente preta de $\Im$;

ii. o conjunto de pontos de grade em cada componente do complemento de $C(\Im)$ deve ser uma componente branca de $\Im$;

iii. uma componente preta P de $\Im$ é adjacente a uma componente branca Q de $\Im$ se, e somente se, os limites da componente de $C(\Im)$ que contém P e da componente do complemento de $C(\Im)$ que contém Q se tocarem.

A figura 2.6(a) apresenta um conjunto de pontos em $\mathbb{Z}^2$ e o respectivo análogo contínuo, considerando a imagem (8, 4) definida por esses pontos. Neste caso, o número de Euler da imagem é 0. Na figura 2.6, é apresentado o mesmo conjunto de pontos e o análogo contínuo correspondente à imagem (4, 8) determinada por esses pontos. O número de Euler da imagem é 1.

Detalhes sobre a determinação do análogo contínuo de uma imagem digital podem ser encontrados em [6]. A definição $\chi(\Im) = \chi(C(\Im))$ demonstra a utilização de conceitos de topologia

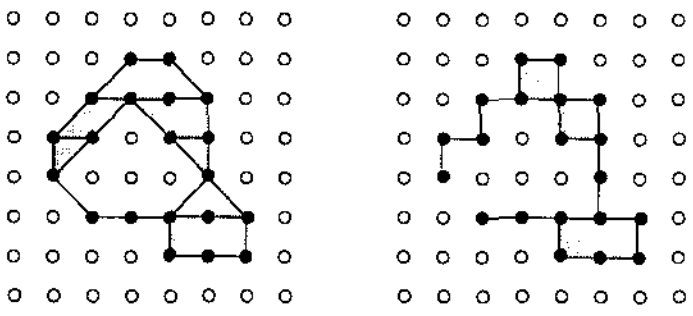


Figura 2.6: *Exemplo de análogo contínuo para imagem (8, 4) (a) e imagem (4, 8) (b).*

poliedral pela topologia digital, através do análogo contínuo. Outras técnicas e teoremas da topologia poliedral podem ser aplicados a imagens digitais através do análogo contínuo.

2.3.3 Cálculo do número de Euler

Mostraremos, a seguir, como o número de Euler pode ser calculado de maneira eficiente para imagens digitais. Esta técnica é apresentada em [10].

Para qualquer quadrado ou cubo unitário K da grade, definimos os seguintes conjuntos elementares:

i. K_0 é o conjunto de vértices de K;

ii. K_1 é o conjunto de arestas de K;

iii. se K é um cubo unitário, K_2 é a união das seis faces de K.

Seja $\Im$ uma imagem digital e $C = C(\Im)$. Se $\Im$ é bidimensional, definimos a *contribuição local* $\chi(\Im; K)$ para todo quadrado unitário K da seguinte forma:

$$\chi(\Im; K) = \chi(C \cap K) - \frac{\chi(C \cap K_1)}{2} - \frac{\chi(C \cap K_0)}{4}.$$

Se $\Im$ é tridimensional, definimos a contribuição local $\chi(\Im; K)$ para todo cubo unitário K da seguinte forma:

$$\chi(\Im; K) = \chi(C \cap K) - \frac{\chi(C \cap K_2)}{2} - \frac{\chi(C \cap K_1)}{4} - \frac{\chi(C \cap K_0)}{8}.$$

Assim, para uma imagem tridimensional $\Im$ e para K contendo apenas um ponto preto, $\chi(C(\Im); K) = 1 - \frac{1}{2} - \frac{1}{4} - \frac{1}{8} = \frac{1}{8}$. Se $\Im$ é uma imagem $(18, 6)$ ou $(26, 6)$ e se K contém apenas dois pontos pretos diagonalmente opostos em uma face de K, então $\chi(C(\Im); K) = 1 - \frac{1}{2} - \frac{2}{4} - \frac{2}{8} = -\frac{1}{4}$.

Teorema 2.1 *Se $\Im$ é uma imagem binária finita, então $\chi(\Im) = \sum_{K \in \Im} \chi(\Im; K)$.*

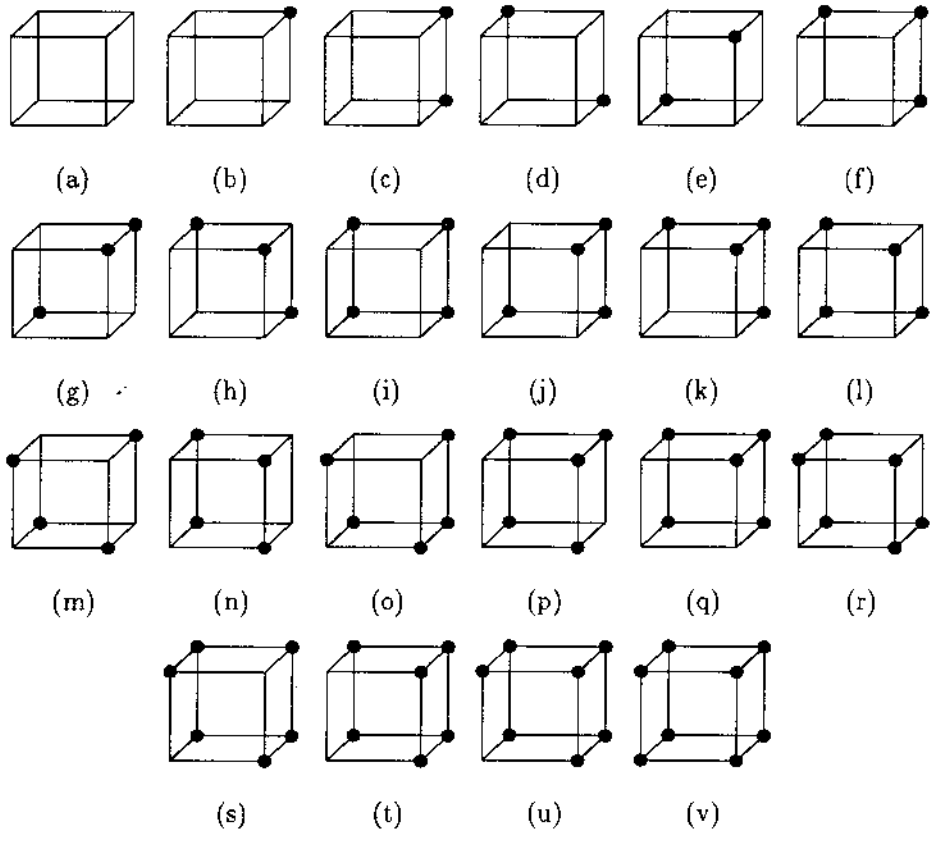


Figura 2.7: *Configurações básicas para cores dos vértices de um cubo unitário K.*

O teorema acima pode ser provado pelo princípio de inclusão-exclusão (ver [6]). Note que o valor de $\chi(\Im; K)$ é totalmente determinado pelas configurações de pontos pretos e brancos de K. Como existem, no caso tridimensional, somente 8 vértices para o cubo K e, portanto, somente $2^8 = 256$ configurações possíveis para as cores desses vértices, os valores de $\chi(\Im; K)$ podem ser previamente calculados e armazenados em uma tabela, ou seja, $\chi(\Im; K)$ pode ser facilmente calculado pois é determinado localmente. Apesar de sua natureza localizada, $\chi(\Im; K)$ constitui uma ferramenta para cálculo do número de Euler em nível global da imagem. Então, para obtermos $\chi(\Im)$, basta calcularmos $\sum_{K \in \Im} \chi(\Im; K)$ para todo cubo K de $\Im$ que possui pelo menos um vértice preto. Desse modo, as contribuições locais $\chi(\Im; K)$ são utilizadas para obter $\chi(\Im)$, uma informação em nível global.

A figura 2.3.3 apresenta as 22 configurações básicas. As demais configurações são obtidas por rotação ou reflexão de uma das configurações básicas para o cubo K. Os valores de $\chi(\Im; K)$, para estas configurações, são apresentados na tabela 2.2.

Configuração básica na figura 2.3.3	Relações de adjacência (β, ω) da imagem			
	(26, 6)	(18, 6)	(6, 26)	(6, 18)
a	0	0	0	0
b	1/8	1/8	1/8	1/8
c	0	0	0	0
d	-1/4	-1/4	1/4	1/4
e	-3/4	1/4	1/4	1/4
f	-1/8	-1/8	-1/8	-1/8
g	-3/8	-3/8	1/8	1/8
h	-1/8	-1/8	3/8	3/8
i	0	0	0	0
j	-1/4	-1/4	-1/4	-1/4
k	-1/4	-1/4	-1/4	-1/4
l	0	0	0	0
m	1/2	1/2	1/2	1/2
n	0	0	0	0
o	3/8	3/8	-1/8	-1/8
p	1/8	1/8	-3/8	-3/8
q	-1/8	-1/8	-1/8	-1/8
r	1/4	1/4	-3/4	1/4
s	1/4	1/4	-1/4	-1/4
t	0	0	0	0
u	1/8	1/8	1/8	1/8
v	0	0	0	0

Tabela 2.2: *Valores de* $\chi(\Im; K)$ *calculados para as configurações básicas de* K.

2.4 Conclusão

Neste capítulo, apresentamos o conceito de relação de adjacência entre pontos de grades 2-d e 3-d. A partir daí, pudemos definir imagem binária e outros conceitos como componentes, cavidades, túneis, etc. Foi apresentada também a definição de número de Euler relacionado a topologia digital e que, como veremos mais tarde, é de grande importância para o estudo de algoritmos de afinamento.

No próximo capítulo, introduziremos a noção de afinamento sobre imagens binárias. Além de uma discussão geral, serão descritos alguns dos principais algoritmos encontrados na literatura.

Capítulo 3

Algoritmos de Afinamento 3-d

O capítulo corrente introduz, em sua seção 3.1, conceitos e princípios básicos inerentes aos algoritmos de afinamento, por vezes recorrendo ao caso 2-d para introduzi-los de maneira mais simples, avançando, em seguida, para o caso 3-d. Serão abordados aspectos topológicos e não-topológicos, bem como paralelismo em algoritmos de afinamento. Em seguida, nas seções 3.2 a 3.6, serão decritos alguns dos principais algoritmos de afinamento 3-d encontrados na literatura.

3.1 Aspectos Gerais

Nesta seção, introduzimos a idéia de afinamento de imagens binárias e descrevemos vários aspectos desta operação, finalizando com o esboço de um algoritmo geral de afinamento.

3.1.1 Introdução e aspectos topológicos

Afinamento é a operação de remoção de pontos de uma imagem visando a obtenção de um esqueleto. Um algoritmo de afinamento funciona de modo iterativo no qual, a cada iteração, são removidos pontos de borda, segundo certos critérios, até que não seja possível mais nenhuma remoção, respeitando-se estes critérios. A figura 3.1 mostra um exemplo típico de afinamento 2-d.

Este exemplo mostra um afinamento "ideal" onde não há, entre outros problemas que podem ocorrer, ocorrência de ramos extras (muitas vezes devidos a ruídos). A figura 3.2 apresenta exemplos de afinamento (figura 3.2(b)-(d)) válidos para uma mesma imagem 2-d (figura 3.2(a)). Notamos que há ocorrência de ramos (figura 3.2(b)). As diferenças entre os diversos esqueletos podem ser claramente percebidas quando os mesmos são sobrepostos ao objeto original (representado em cinza nas figuras 3.2(b)-(d)).

Seja um espaço digital de imagens $\mathcal{E} = (\mathbb{Z}^n, \beta, \omega)$, uma imagem $\Im = (\mathbb{Z}^n, \beta, \omega, B) \in \mathcal{E}$ e um operador $\Psi : \mathcal{E} \mapsto \mathcal{E}$, tal que $\Psi(\Im) = (Z^n, \beta, \omega, B^\Psi)$. Se Ψ é um operador de afinamento, valem as seguintes propriedades:

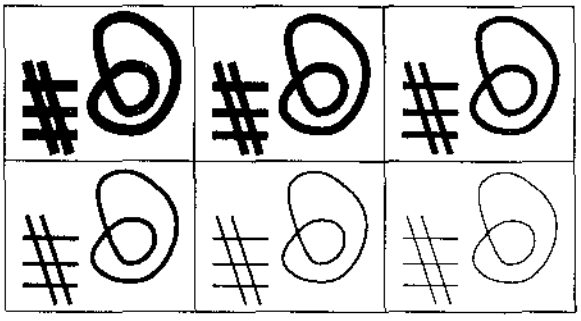
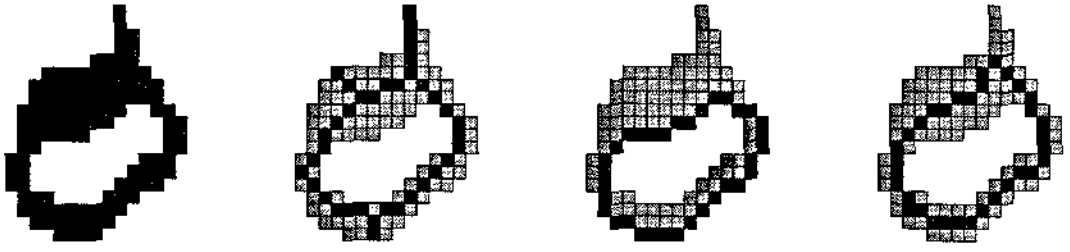


Figura 3.1: *Exemplo de um afinamento típico em 2-d.*



(a) (b) (c) (d)

Figura 3.2: *Imagens 2-d: objeto original (a) e diferentes esqueletos (b)-(d).*

i. Ψ é um operador *anti-extensivo*, i.e., $B^{\Psi} \subseteq B$;

ii. a aplicação de Ψ preserva topologia da imagem original, ou seja, $\Im$ e $\Psi(\Im)$ são *topologicamente equivalentes.*

Na literatura, algumas definições para "preservação de topologia" em afinamento 3-d têm sido encontradas (ver, por exemplo, [5]). Uma das primeiras idéias de preservação de topologia, conforme Tourlakis e Mylopoulos (ver [5]), considera que duas imagens $\Im_1$ e $\Im_2$ são topologicamente equivalentes se elas têm o mesmo número de componentes, cavidades e túneis; contudo, não estabelece nenhuma relação entre os posicionamentos das componentes, cavidades e túneis de $\Im_1$ e $\Im_2$.

Um significado mais rigoroso, neste aspecto, é aquele introduzido por Morgenthaler [5] e enunciado no critério 3.1.

Critério 3.1 *Seja a imagem* $\Im_1 = (\mathbb{Z}^3, \beta, \omega, B)$. *A remoção dos pontos de* $R \subset B$ *preserva topologia se, e somente se,* $\Im_2 = (\mathbb{Z}^3, \beta, \omega, B - R)$ *satisfaz as seguintes condições:*

i. cada componente preta de $\Im_1$ *contém exatamente uma componente preta de* $\Im_2$*;*

ii. cada componente branca de $\Im_2$ *contém exatamente uma componente branca de* $\Im_1$*;*

iii. cada caminho fechado em B *pode ser* digitalmente deformado *em* B *para um caminho fechado em* $B - R$*;*

iv. se um caminho fechado π_1 em $B - R$ pode ser digitalmente deformado em B para outro caminho fechado π_2 em $B - R$, então π_1 pode ser digitalmente deformado em $B - R$ para π_2.

As condições *i* e *ii* garantem que as componentes pretas e brancas de $\Im_1$ e $\Im_2$ estejam "no mesmo lugar". Condições *iii* e *iv* garantem o mesmo para os túneis.

Definição 3.1 *Seja $\Im = (\mathbb{Z}^n, \beta, \omega, B)$ uma imagem binária e $p \in B$ um ponto preto de $\Im$. Chamamos p um* ponto simples *se, e somente se, a remoção de p preserva a topologia de $\Im$, i.e., se $\Im$ e $\Im - \{p\}$ são topologicamente equivalentes.*

Algumas caracterizações de pontos simples têm sido propostas para os casos 2-d e 3-d (ver, por exemplo, [11, 5, 3, 12, 13]), visando melhores implementações de algoritmos de afinamento. É importante notar que o problema de determinação de pontos simples, no caso 3-d, tem se mostrado bem mais difícil que no caso 2-d, principalmente devido à ocorrência de túneis. Até recentemente, foram apresentadas supostas caracterizações de pontos simples para o caso 3-d, nas quais a preservação de topologia não é garantida devido à desconsideração da ocorrência de túneis no espaço 3-d, o que não é efetivamente correto.

Podemos constatar esta maior dificuldade em [4], em que uma suposta caracterização de ponto simples em 3-d atribuída a Srihari em [14] é apresentada. Esta caracterização pode ser facilmente verificada para um ponto mas, como veremos a seguir, não é uma caracterização correta.

Caracterização 3.1 (Srihari) *Seja $\Im = (\mathbb{Z}^3, 26, 6, B)$ uma imagem binária e $p \in B$ um ponto preto de $\Im$. Seja $N(p) = N_{26}(p) \cap \Im$ e $N'(p) = N(p) - \{p\}$. As componentes de $N(p)$ e $N'(p)$ são representadas por $C = \{c_1, \ldots, c_n\}$ e $C' = \{c'_1, \ldots, c'_{n'}\}$, respectivamente. Então, p é simples se p é ponto de borda e $n = n'$.*

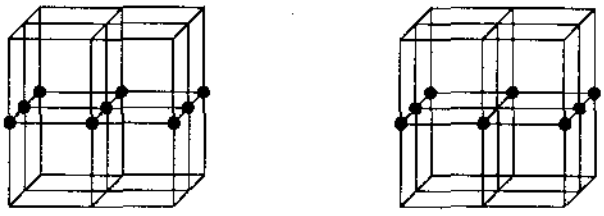
A figura 3.3 mostra determinada configuração de $N(p)$ e $N'(p)$. Para este caso, $N(p) = N'(p) = 1$ e p é ponto de borda. Então, p é considerado ponto simples pela caracterização 3.1. Todavia, um túnel é criado na imagem ao se remover p, o que modifica a topologia da imagem inicial e, portanto, a caracterização 3.1 não é correta.

A caracterização proposta por Morgenthaler é a primeira a caracterizar corretamente todos os pontos simples no espaço 3-d [5]. Tsao e Fu reapresentaram esta caracterização de forma simplificada [15], como está apresentada abaixo.

Caracterização 3.2 (Morgenthaler) *Seja $\Im = (\mathbb{Z}^n, \beta, \omega, B)$ uma imagem binária, $p \in B$ um ponto preto de $\Im$. Então, p é um ponto simples se, e somente se, as seguintes condições são satisfeitas:*

i. p é adjacente a apenas uma componente preta em $N^*_{26}(p) \cap B$;

ii. p é adjacente a apenas uma componente branca em $N_{26}(p) \cap \overline{B}$;



(a) (b)

Figura 3.3: *Contra-exemplo para caracterização 1. Configurações para $N(p)$ (a) e $N'(p)$ (b).*

iii. para o caso 3-d, $\chi(\Im \cap N_{26}(p)) = \chi(\Im \cap N^*_{26}(p))$.

Esta caracterização é válida para imagens (6, 26), (6, 18), (26, 6) e (18, 6). Ademais, para $(\beta,\omega) \in \{(26,6),(18,6)\}$, as condições *i* e *iii* implicam na condição *ii*. Nos casos em que $(\beta,\omega) \in \{(6,26),(6,18)\}$, as condições *ii* e *iii* implicam na condição *i*.

A caracterização 3.2 pode ser implementada usando algoritmos de contagem de componentes em teoria de grafos e a técnica de cálculo do número de Euler apresentada no capítulo anterior. Porém, buscando maior eficiência, diversas outras caracterizações têm sido propostas. Isto se deve ao fato de que a caracterização de pontos simples é um dos elementos cruciais em um algoritmo de afinamento, visto que cada ponto pode ter sua topologia verificada várias vezes até o fim do afinamento. Voltaremos a estas caracterizações durante a descrição dos algoritmos de afinamento disponíveis na literatura e considerados neste trabalho.

3.1.2 Características geométricas do esqueleto

Além da preservação da topologia, o afinamento deve apresentar outros aspectos de caráter não-topológico. Um algoritmo que meramente remove pontos, preservando a topologia, é chamado algoritmo de *encolhimento* ou *shrinking*. Aplicando encolhimento, um paralelepípedo sólido seria reduzido a um único ponto, por exemplo. No processo de afinamento, porém, é necessário evitar a remoção de pontos que proporcionem determinada geometria desejada no esqueleto final (o paralelepípedo seria reduzido a uma linha ou uma superfície, por exemplo).

Um ponto simples cuja remoção deve ser evitada a fim de preservar características da imagem em seu esqueleto é chamado de *ponto final*. No caso 2-d, o esqueleto é sempre composto de linhas, enquanto no caso 3-d, podemos ter esqueletos compostos por linhas e superfícies afinadas (chamados *esqueletos de superfícies*) ou um esqueleto composto apenas por linhas (chamado *esqueleto de linhas*). A definição de pontos finais em 3-d depende, portanto, do esqueleto desejado e das relações de adjacência β e ω das imagens a serem afinadas.

Vejamos um exemplo para imagens (6, 26). Inicialmente, consideramos como ponto final todo ponto simples p que possui apenas um ponto preto em $N^*_{26}(p)$. Com isso, pontos extremos de curvas afinadas serão mantidos na imagem e o resultado do afinamento será um esqueleto de linhas. Consideremos, agora, todo ponto simples p que não pertence a um cubo unitário

com todos os vértices preto de tamanho $2 \times 2 \times 2$ na imagem. Teremos, assim, um esqueleto de superfícies como resultado. A figura 3.4 mostra exemplos de afinamento segundo estas duas definições de ponto final.

(a) (b) (c)

Figura 3.4: *Objeto (a) e seus esqueletos de superfícies (b) e de linhas (c)*

3.1.3 Afinamento em paralelo

Nos últimos anos, tecnologias para obtenção de imagens 3-d têm sido aperfeiçoadas rapidamente e uma quantidade de dados cada vez maior é disponibilizada sob esta forma. Como conseqüência, é crescente a necessidade de técnicas eficientes e velozes para tratamento dos enormes volumes de dados contidos nestas imagens. Neste contexto, algoritmos paralelos se apresentam como uma solução para o afinamento.

O paralelismo pode ser obtido pela remoção de vários pontos simultaneamente. Isso torna mais difícil a determinação dos pontos que podem ser removidos sem alteração da topologia da imagem. Vejamos o exemplo da figura 3.5, considerando a imagem (6, 26). Os pontos p, q e r são pontos simples. Porém, a remoção destes três pontos, simultaneamente, desconecta a imagem, alterando a topologia.

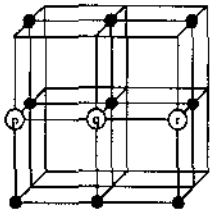


Figura 3.5: *Exemplo em que a remoção paralela de pontos simples numa conexidade (6, 26), modifica a topologia da imagem.*

A estratégia usada para vencer esta dificuldade é dividir cada iteração do afinamento em várias subiterações e a cada subiteração considerar, para remoção, apenas um subconjunto dos

pontos da imagem que satisfaça certas restrições. Essas restrições devem garantir que todos os pontos da imagem que as satisfaçam possam ser removidos sem alteração da topologia.

Existem, na literatura, duas estratégias utilizadas para dividir a imagem nestes subconjuntos: a direcional e a de subcampos, comentadas a seguir.

Estratégias direcionais

Em 2-d, uma idéia popular para se obter um conjunto de pontos para remoção em paralelo é a utilização de direções: apenas pontos que estão de um certo lado do objeto, i.e., pontos que possuem um vizinho branco em dada direção (Norte, Sul, Leste ou Oeste) são considerados para remoção simultânea em cada subiteração [11].

Mais formalmente, definimos $D_{norte}(\Im) = \{(x, y) \in B \,|\, (x+1, y) \notin B\}$ e $D_{sul}(\Im) = \{(x, y) \in B \,|\, (x-1, y) \notin B\}$ como o conjunto de pontos em $\Im = (\mathbb{Z}^n, \beta, \omega, B)$ que estão, respectivamente, nas direções Norte e Sul. Analogamente, definem-se os conjuntos de pontos nas direções Leste e Oeste, $D_{leste}(\Im)$ e $D_{oeste}(\Im)$. Cada subiteração do algoritmo de afinamento em paralelo consiste em remover (simultaneamente) os pontos simples de uma das quatro direções. A cada nova subiteração, nova direção é considerada. A ordem escolhida para as direções deve usar pares de direções opostas a fim de obter um esqueleto mais simétrico possível.

O algoritmo de afinamento definido dessa forma tem a propriedade de garantir a preservação da topologia para imagens $(8, 4)$ e $(4, 8)$, exceto pela possível remoção de componentes compostas por apenas dois pontos pretos. É suficiente verificar este caso especial para obter um algoritmo válido.

Temos, em 3-d, seis direções (Norte, Sul, Leste, Oeste, Acima e Abaixo), mostradas na figura 3.6(b) e definidas de modo análogo ao caso 2-d. Todavia, estratégias direcionais não apresentam boas propriedades topológicas em 3-d.

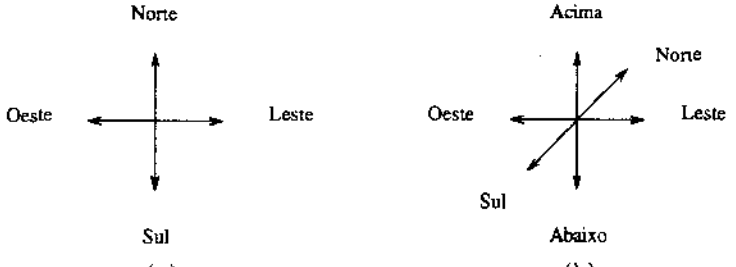


Figura 3.6: Direções em 2-d (a) e 3-d (b).

A figura 3.5 mostra uma imagem (6, 26) contendo três pontos simples p, q e r. Estes são pontos na direção Acima. Observamos que a remoção paralela de p, q e r resulta numa alteração da topologia da imagem. Como conseqüência, algoritmos paralelos de afinamento 3-d baseados em estratégia direcional precisam estabelecer condições adicionais para garantir a preservação da topologia.

Algoritmos de subcampos

Outra estratégia para remoção de pontos simples em paralelo é a estratégia baseada em *subcampos*: a imagem é particionada em subconjuntos chamados subcampos. Em uma dada subiteração, somente pontos de um dado subcampo são considerados para remoção. Diversos algoritmos de afinamento em 2-d têm sido propostos seguindo essa estratégia (ver, por exemplo, [16]).

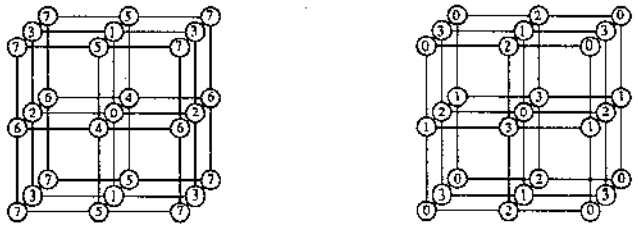
Consideremos o caso 3-d. Seja $p = (p_1, p_2, p_3) \in \mathbb{Z}^3$ e $P_i = p_i \, mod \, 2$, i.e., $P_i = 0$ ($P_i = 1$) se p_i é par (ímpar). Definimos, para $i \in [0,7]$, os subcampos $S_i = \{p \in \mathbb{Z}^3 | i = 4P_1 + 2P_2 + P_3\}$. Os oito subcampos S_i constituem uma partição de $\mathbb{Z}^3$, ilustrada na figura 3.7(a).

Pode ser mostrado que vale a seguinte propriedade: $\forall p \in S_i, \forall q \in S_i, p \notin N_{26}(q)$. Visto na caracterização 3.2 que a caracterização de ponto simples envolve apenas a sua 26-vizinhança, a remoção de um ponto não altera a natureza (simples ou não) de outros no mesmo subcampo. Então, temos que [8]:

Propriedade 3.1 *A remoção simultânea de pontos simples de um mesmo subcampo S_i não altera a topologia da imagem.*

Sejam os subcampos S'_i, para $i \in [0,3]$, definidos por $S'_i = \{p \in \mathbb{Z}^3 | \overline{P_1}(2P_2 + P_3) + P_1(2\overline{P_2} + \overline{P_3})\}$. Os quatro subcampos S'_i constituem uma partição de $\mathbb{Z}^3$, mostrada na figura 3.7(b). Temos a seguinte propriedade [8]:

Propriedade 3.2 *Para imagens (26, 6), a remoção simultânea de pontos simples de um dado sub-campo S'_i não muda a topologia da imagem exceto pela possível remoção de componentes compostas por apenas dois pontos pretos.*



(a) (b)

Figura 3.7: *Partição de $\mathbb{Z}^3$ em oito subcampos (a) e quatro subcampos (b).*

3.1.4 Esboço de um algoritmo geral de afinamento

Podemos resumir os aspectos discutidos nas seções anteriores no seguinte esboço de algoritmo de afinamento:

Algoritmo Afinamento($\Im$)
Entrada: Imagem $\Im = (\mathbb{Z}^3, \beta, \omega, B)$.
Saída: Esqueleto de $\Im$.

Repita
    pontos_removidos ←0;
    ParaTodo $p \in B$ **faça**
        Se $p \in B$ **e** p é simples **e** p não é ponto final **então**
            Remove(p, $\Im$);
            pontos_removidos ←pontos_removidos+1;
        Fim-Se
    Fim-ParaTodo
Até pontos_removidos = 0;
retorne $\Im$;
Fim-Algoritmo

Algoritmo 3.1: *Algoritmo geral de afinamento*

Para um algoritmo paralelo, a declaração **ParaTodo** da linha 3 pode ser executada simultaneamente para todos os pontos em B.

Discutimos, até aqui, vários aspectos importantes relativos aos problemas envolvidos na definição de algoritmos de afinamento 3-d. Nas próximas seções deste capítulo, descreveremos vários algoritmos de detecção de pontos simples e afinamento explicitando as soluções de implementação adotadas para cada aspecto levantado nas discussões da presente seção.

3.2 Algoritmo de Tsao e Fu

Tsao e Fu propõem, em [4], um algoritmo de afinamento em paralelo baseado em direções e no algoritmo de Lobregt et al. (ver [3]) para verificação de topologia. Este algoritmo é aplicável a imagens (26, 6).

3.2.1 Número de conexidade

Lobregt et al. em [3] utilizam um *número de conexidade* para determinação de pontos simples. Este número de conexidade corresponde à *característica de superfície* (igual a duas vezes o número de Euler) da imagem. Para calcular o número de conexidade foi usada a mesma técnica apresentada na seção 2.3 para cálculo do número de Euler. A vizinhança do ponto considerado é dividida em oito cubos ou *octantes* de tamanho $2 \times 2 \times 2$ e as contribuições locais (ver seção 2.3) de cada cubo são somadas. As contribuições das 256 configurações possíveis são calculadas

previamente e armazenadas em uma tabela. Um ponto é considerado simples se não há variação no número de conexidade após a remoção do mesmo.

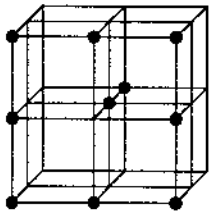


Figura 3.8: *Exemplo de configuração em que o número de Euler e o número de conexidade não se alteram após a remoção do ponto central, havendo alteração de topologia.*

Esta condição é, porém, insuficiente para preservação de topologia nos termos estabelecidos por Morgenthaler (ver seção 3.1). No exemplo da figura 3.8, a remoção do ponto central provoca a criação de um túnel e de uma componente preta a mais, o que deixa inalterados os valores do número de Euler e, conseqüentemente, do número de conexidade.

Como solução, Tsao e Fu adicionaram outras condições para que um ponto seja considerado simples. Como veremos, a seguir, estas condições dizem respeito à conexidade nos planos ortogonais à imagem, passando pelo ponto considerado. Elas garantem, também, uma definição de pontos finais para esqueletos de superfícies e esqueletos de linhas, além de contribuirem para a remoção de pontos em paralelo.

3.2.2 Pontos finais

Tsao e Fu apresentam, ainda, definições de pontos finais para esqueletos de linhas e esqueletos de superfícies com base na conexidade das imagens bidimensionais correspondentes aos planos ortogonais à imagem no ponto considerado.

Seja a imagem $\Im = (\mathbb{Z}^3, 26, 6, B)$ e $p = (p_x, p_y, p_z) \in B$. Definimos os *planos de checagem* P_x, P_y e P_z: $P_x(p) = (\mathbb{Z}^2, 8, 4, B_x)$, sendo $B_x = \{(y, z)|(p_x, y, z) \in B\}$; $P_y(p) = (\mathbb{Z}^2, 8, 4, B_y)$, sendo $B_y = \{(x, z)|(x, p_y, z) \in B\}$; $P_z(p) = (\mathbb{Z}^2, 8, 4, B_z)$, sendo $B_x = \{(x, y)|(x, y, p_z) \in B\}$. O ponto p é dito *removível em* P_x se, e somente se, a remoção de $p' = (p_y, p_z)$ de P_x não desconecta $P_x \cap N_8(p')$ e o número de pontos pretos em $P_x \cap N_8(p')$ é maior ou igual a 2. Analogamente, definimos ponto removível em P_y e ponto removível em P_z (ver exemplo na figura 3.9). Consideraremos as seis direções descritas na seção 3.1.3. Seja ponto simples $p = (p_x, p_y, p_z)$ em $\Im$ tal que $p \in D_{norte}(\Im)$ ou $p \in D_{sul}(\Im)$. Se p não é *removível* nos planos $P_y(p)$ ou $P_z(p)$, então p é considerado um ponto final. Para $p \in D_{leste}$ ou $p \in D_{oeste}$, p é ponto final se p não é removível em $P_x(p)$ ou em $P_z(p)$. Finalmente, para $p \in D_{acima}$ ou $p \in D_{abaixo}$, p é ponto final se p não é removível em $P_x(p)$ ou em $P_y(p)$.

Esta definição de pontos finais leva à obtenção de esqueletos de superfícies. É importante notar que esta definição permite a remoção em paralelo de todos os pontos simples que não sejam pontos finais em uma mesma direção. A prova é apresentada em [4].

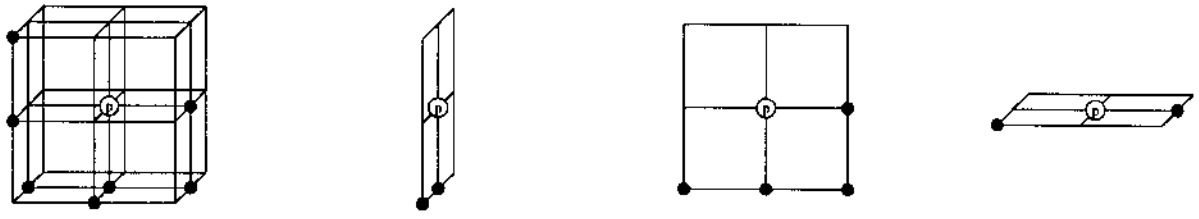


Figura 3.9: *Exemplo de configuração de $N_{26}(p)$ (a), P_x (b), P_y (c) e P_z (d). O ponto p é removível em P_x e P_y mas não é removível em P_z.*

Para obtenção do esqueleto de linhas, partimos do esqueleto de superfícies e repetimos o afinamento relaxando a condição de ponto final. Agora, um ponto p é considerado ponto final em uma direção α, se o número de pontos em cada plano de verificação é menor ou igual a 2 e o 6-vizinho de p na direção oposta a α é branco. A figura 3.10 mostra exemplo de ponto final para esqueleto de superfícies (a) e para esqueleto de linhas (b). Note que na figura 3.10(b) o ponto central não é considerado ponto final apenas quando forem verificados os pontos na direção Norte, pois o seu 6-vizinho na direção Sul apresenta cor preta.

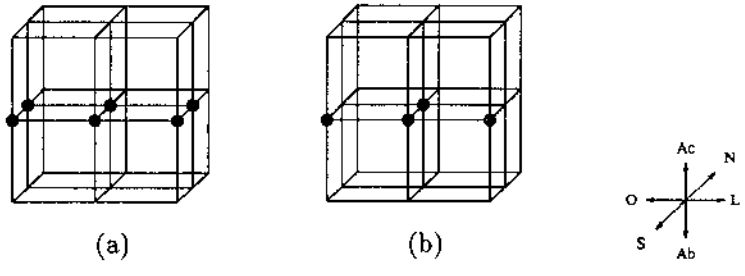


Figura 3.10: *Exemplo de ponto final para esqueleto de supefícies (a) e para esqueleto de linhas (b).*

3.2.3 Esboço do algoritmo de Tsao e Fu

O algoritmo 3.2 mostra um esboço do algoritmo de Tsao e Fu.

Algoritmo Tsao-Fu($\Im$)
Entrada: Imagem $\Im = (\mathbb{Z}^3, 26, 6, B)$.
Saída: Esqueleto de $\Im$ (algoritmo Tsao-Fu).

Repita
pontos_removidos ←0;
ParaTodo $\alpha \in \{norte, sul, leste, oeste, acima, abaixo\}$ **faça**
ParaTodo $p \in (B \cap D_\alpha(\Im))$ **faça em paralelo**
Se $p \in B$ **e** p é simples **e** p não é ponto final **então**
Remove(p, $\Im$);
pontos_removidos ←pontos_removidos+1;
Fim-Se
Fim-ParaTodo
Fim-ParaTodo
Até pontos_removidos = 0;
retorne $\Im$;
Fim-Algoritmo

Algoritmo 3.2: *Algoritmo de afinamento Tsao-Fu*

3.3 Algoritmo de Bertrand e Aktouf

Em [7], Bertrand e Malandain apresentam uma caracterização de pontos simples válida para imagens (26, 6) e (6, 26). Em [17], Bertrand estendeu esta caracterização para imagens (18, 6) e (6, 18). Tal caracterização é apresentada na próxima seção, juntamente com o algoritmo de afinamento para imagens (26, 6) baseado na mesma e apresentado em [8].

3.3.1 Números topológicos

Apresentamos, inicialmente, algumas definições básicas:

Definição 3.2 (Vizinhança geodésica) *Sejam $X \subset \mathbb{Z}^3$ e $p \in X$. A* vizinhança geodésica *de p em X de ordem k é o conjunto $N_\beta^k(p, X)$ definido indutivamente por:*

$$\begin{aligned} N_\beta^1(p, X) &= N_\beta^*(p) \cap X \\ N_\beta^k(p, X) &= \bigcup\{N_\beta(y) \cap N_{26}^*(p) \cap X,\ y \in N_\beta^{k-1}(p, X)\}. \end{aligned}$$

Em outras palavras, $N_\beta^k(p, X)$ é o conjunto composto por todo ponto $y \in N_{26}^*(p) \cap X$ tal que existe um caminho β-conexo π de p até y de tamanho menor ou igual a k e todos os pontos de π, exceto possivelmente p, pertencem a $N_{26}^*(p) \cap X$.

Em [17], Bertrand faz uma distinção entre 6-adjacência relativa a imagens (26, 6) e (6, 26) e 6-adjacência relativa a imagens (18, 6) e (6, 18), notadas respectivamente por 6-adjacência e 6^+-adjacência. Esta diferenciação simplifica a caracterização de ponto simples utilizando os números topológicos definidos a seguir.

Seja X um conjunto de pontos em $\mathbb{Z}^3$. Notamos por $\mathcal{C}_\beta(X)$ e $\mathcal{C}^p_\beta(X)$, respectivamente, o conjunto de componentes de X quando considerada a β-adjacência e o conjunto de componentes (também considerando β-adjacência) de X mas que sejam β-adjacentes ao ponto p.

Definição 3.3 (Números topológicos) *Sejam $X \subset \mathbb{Z}^3$ e $p \in X$. Os números topológicos relativos a p e X são:*

$$\begin{aligned} T_6(p, X) &= \#\mathcal{C}_6[N_6^2(p, X)], \\ T_{6+}(p, X) &= \#\mathcal{C}_6[N_6^3(p, X)], \\ T_{18}(p, X) &= \#\mathcal{C}_{18}[N_{18}^2(p, X)], \\ T_{26}(p, X) &= \#\mathcal{C}_{26}[N_{26}^1(p, X)]. \end{aligned}$$

A definição de números topológicos acima leva a uma caracterização concisa de ponto simples.

Caracterização 3.3 *Sejam $\Im = (\mathbb{Z}^3, \beta, \overline{\beta}, B)$ uma imagem binária onde $(\beta, \overline{\beta}) \in \{(26,6), (6,26), (18,6^+), (6^+,18)\}$ e $p \in B$ um ponto preto de $\Im$. O ponto p é simples se, e somente se, $T_\beta(p, B) = 1$ e $T_{\overline{\beta}}(p, \overline{B}) = 1$.*

Como podemos ver, a caracterização acima não necessita nenhuma informação direta sobre túneis e pode ser verificada rapidamente.

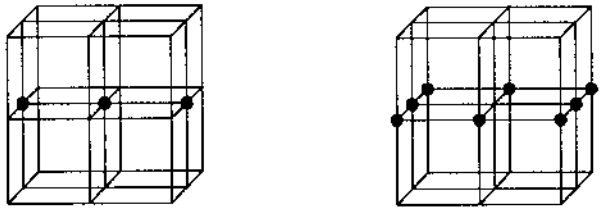
Sejam $X \subset \mathbb{Z}^3$ um conjunto de pontos pretos e $p \in X$. Denotamos $X^p_\beta = N^*_\beta(p) \cap X$ e $\overline{X}^p_\beta = N^*_\beta(p) \cap \overline{X}$, para $\beta \in \{6, 18, 26\}$.

Proposição 3.1 *Dados $X \subset \mathbb{Z}^3$ um conjunto de pontos pretos e $p \in X$, temos que:*

$$\begin{aligned} \#\mathcal{C}_6[N_6^2(p, X)] &= \#\mathcal{C}_6^p[X_{18}^p] \\ \#\mathcal{C}_{26}[N_{26}^1(p, X)] &= \#\mathcal{C}_{26}^p[X_{18}^p], \end{aligned}$$

o que representa uma implementação mais eficiente da caracterização de pontos simples para imagens (26, 6) e (6, 26).

Além da caracterização de ponto simples, os números topológicos encontram utilidade na definição de pontos finais. Um ponto p tal que $T_n(p, X) = 0$ é um *ponto isolado*. Se $T_{\overline{n}}(p, \overline{X}) = 0$, temos um ponto interior e pontos de borda são caracterizados por $T_{\overline{n}}(p, \overline{X}) \neq 0$. Consideremos um ponto p tal que $T_n(p, X) = 2$. Se removermos p, localmente desconectamos X e dizemos que p é um *istmo 1-d* (ver exemplo na figura 3.11). Do mesmo modo, se $T_{\overline{n}}(p, \overline{X}) \geq 2$, p é chamado *istmo 2-d*, visto que sua remoção leva a unir localmente componentes de $\overline{X}$. Um ponto p tal que



(a) (b)

Figura 3.11: *Exemplo de istmos 1-d (a) e istmo 2-d (b).*

Números topológicos	Classificação
$T = 0$	ponto isolado
$\overline{T} = 0$	ponto interior
$\overline{T} \neq 0$	ponto de borda
$T = 2, \overline{T} = 1$	istmo 1-d simples
$T = 1, \overline{T} = 2$	istmo 2-d simples
$T \geq 3, \overline{T} = 1$	junção 1-d simples
$T = 1, \overline{T} \geq 3$	junção 2-d simples
$T \geq 2, \overline{T} \geq 2$	istmo múltiplo
$T \geq 2$ ou $\overline{T} \geq 2$	istmo

Tabela 3.1: *Classificação de pontos usando números topológicos. Notamos* $T = T_n(p, X)$ *e* $\overline{T} = T_{\overline{n}}(p, \overline{X})$.

$T_n(p, X) = 2$ e $T_{\overline{n}}(p, \overline{X}) = 1$ pode ser visto como *istmo 1-d simples*. Também temos *istmos 2-d simples* quando $T_n(p, X) = 1$ e $T_{\overline{n}}(p, \overline{X}) = 2$. A tabela 3.1 mostra esta classificação topológica.

Os números topológicos exercem papel importante na caracterização de pontos de linha e pontos de superfície. Desde que o esqueleto de um objeto consiste de linhas e superfícies, podemos usar números topológicos não apenas para verificar se um ponto é simples, mas também para detectar pontos finais. Suponha que afinemos um objeto iterativamente: alguns pontos que inicialmente eram pontos de interior devem aparecer como istmos 1-d (istmos 2-d). Estes pontos têm características locais de uma curva (superfície). São, portanto, considerados pontos finais e sua remoção deve ser evitada no processo de afinamento.

3.3.2 Esboço do algoritmo de Bertrand-Aktouf

Apresentamos, a seguir, o algoritmo para afinamento considerando 8 sub-campos. Assim, cada iteração do algoritmo é dividida em 8 sub-iterações, cada sub-iteração correspondendo a um sub-campo S_i, como descrito na seção 3.1.3. Removemos todos os pontos simples no sub-campo S_i que não sejam pontos finais. Repetimos sucessivamente para $i = 1 \ldots 8$.

Para uma definição precisa do algoritmo, é necessário, ainda, uma definição exata de pontos finais. Como discutido na seção 3.3.1, usaremos números topológicos para detecção de pontos finais. Durante o processo de afinamento, consideraremos istmos como pontos finais, o que nos leva a um esqueleto de superfícies. Para esqueleto de linhas, apenas istmos 1-d serão considerados como pontos finais. Será necessário etiquetar os pontos finais, senão, superfícies serão encolhidas por suas bordas e linhas por suas extremidades. Estas considerações levam ao seguinte algoritmo:

Algoritmo Bertrand-Aktouf($\Im$)
Entrada: Imagem $\Im = (\mathbb{Z}^3, 26, 6, B)$.
Saída: Esqueleto de $\Im$ (algoritmo Bertrand-Aktouf).

ParaTodo $p \in B$ **faça**
    etiqueta[p] ←"não final";
Fim-ParaTodo
Repita
    pontos_removidos ←0;
    ParaTodo $i \in [0, 7]$ **faça**
        ParaTodo $p \in (B \cap S_i(\Im))$ **faça em paralelo**
            Se p é ponto final **então**
                etiqueta[p] ←"final";
            Fim-Se
            Senão
                Se p é simples **e** etiqueta[p] = "não final" **então**
                    Remove(p, $\Im$);
                    pontos_removidos ←pontos_removidos+1;
                Fim-Se
            Fim-Senão
        Fim-ParaTodo
    Fim-ParaTodo
Até pontos_removidos = 0;
retorne $\Im$;
Fim-Algoritmo

Algoritmo 3.3: *Algoritmo de afinamento Bertrand-Aktouf*

3.3.3 Implementação booleana

A implementação direta da definição de números topológicos envolve a computação de números de componentes conexas (seção 2.2). Portanto, alguns algoritmos da teoria dos grafos para contagem de componentes conexas podem ser usados. A complexidade destes algoritmos depende do tamanhos dos caminhos incluídos nas componentes conexas, requerem estruturas de dados tais como listas e é preciso fazer uma etiquetagem dos pontos. Há, portanto, necessidade de um método mais simples para caracterizar pontos simples e computar números topológicos.

Em [8], os autores apresentam um método que necessita apenas da checagem de 5 configurações básicas para verificação de ponto simples. No restante desta seção, usaremos 26-adjacência para os pontos pretos. Se desejarmos usar 6-adjacência, basta trocar as ocorrências de X e $\overline{X}$ entre si.

Definição 3.4 *Seja $X \subset \mathbb{Z}^3$ e $p \in X$. Definimos os seguintes conjuntos de pontos.*

i. $A_{18} = \{q \mid q$ *é um 18-vizinho de* p *tal que* $q \in X$ *e* $N^*_{26}(q) \cap X^p_{18} = \emptyset\}$;

ii. $A_{26} = \{q \mid q$ *é um 26-vizinho de* p *tal que* $q \in X$ *e* $N^*_{26}(q) \cap X^p_{18} = \emptyset\}$;

iii. $\overline{A_6} = \{q \mid q$ *é um 6-vizinho de* p *tal que* $q \in \overline{X}$ *e* $N^*_6(q) \cap \overline{X^p_{18}} = \emptyset\}$;

iv. $\overline{A_{18}} = \{q \mid q$ *é um 18-vizinho de* p *tal que* $q \in \overline{X}$ *e* $N^*_6(q) \cap X^p_6 = \emptyset\}$;

v. $\overline{A_{26}} = \{q \mid q$ *é um 6-vizinho de* p *tal que* $q \in \overline{X}$ *e* $N^*_{26}(q) \cap X^p_{18} = \emptyset\}$;

Propriedade 3.3 *Seja* $\#C = \#\overline{X^p_6} - \#\overline{A_6} - \#\overline{A_{18}} + \#\overline{A_{26}} + \#A_{18} + \#A_{26}$.
Se $\#C = 0$ *e* $\#\overline{X^p_6} - \#\overline{A_6} \neq 0$, *então* $T_{26}(p, X) = \#A_{18} + \#A_{26} + 2$ *e* $T_6(p, X) = \#\overline{A_6} + 1$.
Se $\#X^p_{18} = \#A_{18}$, *então* $T_{26}(p, X) = \#A_{18} + \#A_{26}$ *e* $T_6(p, X) = 1$.
Caso contrário, $T_{26}(p, X) = \#A_{18} + \#A_{26} + 1$ *e* $T_6(p, X) = \#\overline{A_6} + \#C$.

A partir da caracterização 3.3, as expressões acima descrevendo números topológicos podem ser simplificadas para caracterizar pontos simples, conduzindo a uma caracterização booleana dos mesmos, como a seguir:

Caracterização 3.4 *Sejam* $\Im = (\mathbb{Z}^3, 26, 6, B)$ *uma imagem binária e* $p \in B$ *um ponto preto de* $\Im$. *O ponto* p *é simples se, e somente se,*

$$
\begin{array}{l}
(\#\overline{X^p_6} = 1) \text{ ou} \\
(\#X^p_{26} = 1) \text{ ou} \\
(\#A_{26} = 0, \#X^p_{18} = 1) \text{ ou} \\
(\#\overline{A_6} = 0, \#A^p_{26} = 0, \#A_{18} = 0, \#\overline{X^p_6} - \#\overline{A_{18}} + \#\overline{A_{26}} = 1).
\end{array}
$$

O algoritmo de afinamento pode, enfim, ser implementado usando diretamente a caracterização 3.4. Consideremos um istmo. Para imagens (26, 6), podemos ver que: p é um istmo se, e somente se, $p \in B$, p não é simples, $\#X^p_{26} \neq 0$, e $\#\overline{X^p_6} \neq 0$. Isto nos permite implementar todo o algoritmo de afinamento com condições *booleanas*, para imagens (26, 6) e esqueletos de superfícies.

Não há uma condição tão simples para istmos 1-d. Então foi proposta uma condição aproximada para esqueleto de linhas: consideramos como ponto final todo ponto p que não é simples e tal que $\#\overline{X^p_6} = 2$.

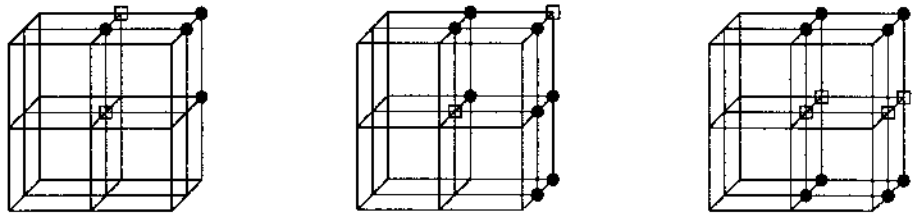
3.4 Esquema de afinamento em paralelo

Bertrand apresenta, em [18], uma estratégia geral para obtenção de algoritmos paralelos de afinamento 3-d. Esta estratégia, discutida a seguir, é válida para imagens (26, 6) e (6, 26) e funciona a partir da determinação de um conjunto de pontos candidatos à remoção simultânea de uma imagem sem alteração de topologia.

3.4.1 Pontos P-simples

Definição 3.5 (P-simples) *Sejam $X \subset \mathbb{Z}^3$, $P \subset X$ e $p \in P$. Dizemos que p é P-simples se $\forall S \subset P - \{p\}$, p é simples para $X - S$. Denotamos $S(P)$ o conjunto de todos os pontos P-simples em P. Um conjunto D é P-simples se $D \subset S(P)$.*

Alguns exemplos de ponto P-simples são mostrados na figura 3.12. Apenas a configuração em (a) corresponde a um ponto P-simples (6, 26) e somente as configurações (a) e (b) correspondem a configurações P-simples (26, 6).



(a) (b) (c)

Figura 3.12: *Exemplos de configurações de pontos P-Simples. Pontos pertencendo a $X-P$ e P são representados respectivamente por $\bullet$ e $\square$. Os demais pontos da grade pertencem a $\overline{X}$.*

Sejam $X \subset \mathbb{Z}^3$ e $p \in X$. Definimos $G_6(p, X) = N_6^2(p, X)$ e $G_{26}(p, X) = N_{26}^1(p, X)$.

Proposição 3.2 *Sejam $\Im = (\mathbb{Z}^3, \beta, \omega, B)$ e $P \subset B$. Se P é P-simples então $\Im$ e $\Im - P$ são topologicamente equivalentes [18].*

Seja $P = \{p_1, \ldots, p_k\}$ um conjunto P-simples de $\Im$. Suponha que sejam removidos seqüencialmente os pontos $p_1, \ldots, p_k$. Seja $P_i = \{p_1, \ldots, p_i\}$. No passo i, do procedimento de remoção, teremos que p_i é simples em $\Im - P_i$, pois P é P-simples e, portanto, a topologia não é alterada com a remoçãode p_i. Por indução, é fácil concluir que $\Im$ e $\Im - P$ são topologicamente equivalentes.

A caracterização a seguir permite uma verificação eficiente de pontos P-simples.

Caracterização 3.5 (Ponto P-simples) *Sejam $\Im = (\mathbb{Z}^3, \beta, \omega, B)$, sendo que $(\beta, \omega) \in \{(26,6), (6,26)\}$; $P \subset B$ e $p \in P$; denotamos $R = B - P$:*

$$p \text{ é } P\text{-simples} \Leftrightarrow \begin{cases} T_\beta(p, R) = 1 \\ T_\omega(p, \overline{B}) = 1 \\ \forall y \in N_\beta^*(p) \cap P,\ N_\beta^*(p) \cap G_\beta(p, R) \neq \emptyset \\ \forall y \in N_\omega^*(p) \cap P,\ N_\omega^*(p) \cap G_\omega(p, \overline{B}) \neq \emptyset \end{cases}$$

A demonstração para esta caracterização pode ser encontrada em [18]. A partir desta caracterização, diversos algoritmos de afinamento podem ser elaborados considerando diferentes definições do conjunto P.

3.4.2 Esquema de afinamento

Como exemplo de aplicação da noção de ponto P-simples, é proposto um esquema de afinamento 3-d paralelo usando a estratégia direcional [18]. Veremos que o esquema leva à construção de vários algoritmos de afinamento, de acordo com as diversas formas de definição de ponto final.

Sejam $\alpha(p)$, para $\alpha \in [0,\ldots,5]$, os 6-vizinhos do ponto p nas direções *Norte*, *Sul*, *Leste*, *Oeste*, *Acima*, *Abaixo*, respectivamente. Para $\Im = (\mathbb{Z}^3, \beta, \omega, B)$, definimos $D_\alpha(\Im) = \{p \in B | \alpha(p) \in \overline{B}\}$. O esquema de afinamento pode ser descrito da seguinte maneira:

$$B^{i+1} = B^i - S(P^i), \text{ com } P^i = \overline{E(B^i)} \cap D_{i \bmod 6}(B^i),$$

sendo que $B^0 = B$ e $E(X)$ denota o conjunto dos pontos finais em X. O esqueleto de $\Im$ será $\Im'(\mathbb{Z}^3, \beta, \omega, B^k)$, k sendo tal que $B^k = B^{k+6}$.

Vejamos um exemplo de algoritmo derivado deste esquema, para imagens (6, 26), onde precisamos apenas definir uma condição de ponto final para esta adjacência. Definimos um ponto final como um ponto preto que não pertence a nenhum cubo de tamanho $2 \times 2 \times 2$ contido em B. Esta definição, aplicada ao esquema acima, nos conduz à obtenção de esqueletos de superfície.

Em [18] são apresentadas outras aplicações de pontos P-simples, como a validação de operadores paralelos de afinamento, que não abordaremos aqui.

3.5 Algoritmo de Saha et al.

Saha et al. apresentaram em 1994 (ver [13]) uma caracterização de ponto simples, considerando as adjacências (26, 6). Não é apresentado, no trabalho, um algoritmo de afinamento completo mas, sim, um algoritmo para verificação de ponto simples implementado com o uso de tabelas.

A caracterização utilizada inicialmente para elaboração do algoritmo é a seguinte:

Caracterização 3.6 (Saha et al.) *Seja a imagem $\Im = (\mathbb{Z}^3, 26, 6, B)$. O ponto $p \in B$ é dito simples em $\Im$ se, e somente se, as seguintes condições forem satisfeitas:*

i. $N^*_{26}(p) \cap \overline{B}$ *possui pelo menos um 6-vizinho de* p;

ii. $N^*_{26}(p) \cap B \neq \emptyset$;

iii. o conjunto de pontos em $N^*_{26}(p) \cap B$ *é conectado;*

iv. o conjunto de pontos 6-vizinhos de p *em* $N^*_6(p) \cap \overline{B}$ *é 6-conectado em* $N^*_{18}(p) \cap \overline{B}$.

Com base na caracterização acima, é feita uma classificação geométrica das possíveis configurações de $N^*_{26}(p)$, visando a implementação do algoritmo para detecção de pontos simples. Esta classificação é apresentada na tabela 3.2. Obviamente, várias configurações pertencem a uma mesma classe. Para cada Classe i escolhemos uma configuração arbitrária denominada $base_i$, sendo que membros desta classe constituem rotação desta configuração.

Classe	Descrição	Pontos efetivos
classe 0	todos os 6-vizinhos são pretos	0
classe 1	cinco 6-vizinhos são pretos	0
classe 2	dois pares opostos de 6-vizinhos são pretos	0
classe 3	um par de 6-vizinhos opostos e dois 6-vizinhos não opostos são pretos	1
classe 4	um par de 6-vizinhos opostos e outro 6-vizinho são pretos	2
classe 5	três 6-vizinhos não opostos são pretos	4
classe 6	um par de 6-vizinho opostos é preto	4
classe 7	dois 6-vizinhos não opostos são pretos	7
classe 8	um 6-vizinho não é preto	12
classe 9	nenhum 6-vizinho é preto	20

Tabela 3.2: *Classificação geométrica de $N_{26}^{*}(p)$, segundo a vizinhança $N_{6}^{*}(p)$.*

O conjunto de *pontos efetivos* é aquele que, classificada a vizinhança $N_{26}(p)$, segundo a tabela 3.2, determina se p é simples ou não. Por exemplo, na figura 3.13 temos apenas um ponto efetivo.

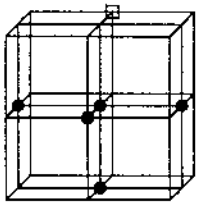


Figura 3.13: *O □ marca o ponto efetivo q. Se q tiver cor preta o ponto central p não é simples (uma nova componente preta é criada pela remoção do ponto central p), caso contrário p é simples. Os demais 18-vizinhos (não efetivos) não têm importância na determinação da natureza (simples ou não) de p.*

A partir da classificação de $N_{26}^{*}(p)$, podemos inferir se p é simples ou não, de acordo com os sequintes casos:

Caso 1. Se a configuração de 6-vizinhos pertence à Classe 0, p não é simples, visto que as condições *ii* e *iv* da caracterização 3.6 nunca são satisfeitas. Nenhuma computação adicional é necessária.

Caso 2. Se a configuração de 6-vizinhos pertence à Classe 1, todas as condições são satisfeitas e p é simples. Nenhuma computação adicional é necessária.

Caso 3. Se a configuração de 6-vizinhos pertence à Classe 2, p não é simples, visto que a condição iv da caracterização nunca é satisfeita. Nenhuma computação adicional é necessária.

Caso 4. Se a configuração de 6-vizinhos pertence à Classe 3 ou à classe 4, as condições i, ii e iii da caracterização 3.6 são sempre satisfeitas. A condição iv é satisfeita se, e somente se, todos os pontos efetivos forem brancos.

Caso 5. Quando a configuração dos 6-vizinhos pertence à uma das Classes 5-8, as condições i e ii são sempre satisfeitas. A satisfação das condições iii e iv é determinada pela configuração dos pontos efetivos. Seja n_i o número de pontos efetivos da classe i. Assim, usamos uma palavra δ_i, com n_i bits, para especificar uma configuração de pontos efetivos. Cada bit corresponde à cor (branco ou preto) de um ponto efetivo. É gerada uma tabela T_i para cada Classe i onde a j-ésima entrada é 1 se p é simples em $N_{26}^*(p)$ com $base_i$ como configuração de 6-vizinhos e j como configuração de pontos efetivos. Caso contrário, a entrada é 0. A tabela T_i deve ter entradas para todas as 2^{n_i} possíveis configurações de pontos efetivos.

Seja γ uma configuração de 6-vizinhos de p pertencente à Classe i. Através de uma rotação podemos transformar γ em $base_i$. Seja r_γ tal rotação. Através de r_γ podemos mapear os pontos efetivos de γ para os pontos efetivos de $base_i$. Todas as rotações r_γ podem ser calculadas, permitindo que uma mesma tabela T_i seja usada para todas as configurações da Classe i.

Caso 6. Em $N_{26}^*(p)$, uma única configuração de 6-vizinhos pertence à Classe 9. Neste caso, o número de pontos efetivos é 20. Serão necessárias portanto 2^{20} entradas na tabela T_9 (128 Kbytes, assumindo 1 bit por entrada).

A figura 3.14 mostra as configurações $base_i$ e os respectivos pontos efetivos para as Classes 3-9.

3.5.1 Esboço do algoritmo de Saha et. al

Apresentamos, no algoritmo 3.4, um esboço do algoritmo de verificação de pontos simples de Saha et al.

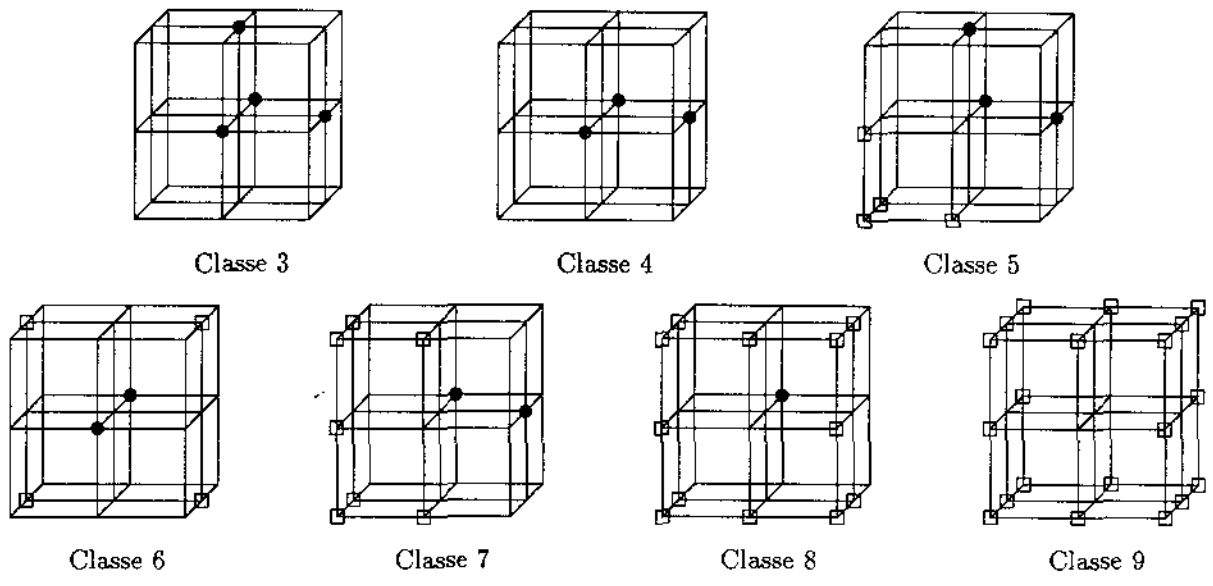


Figura 3.14: *Configurações base$_i$. Os pontos efetivos estão representados por* □. *Os pontos pretos em* $N_6^*(p)$ *estão representados por* •. *O restante dos pontos em* $N_6^*(p)$ *tem cor branca.*

Algoritmo Saha-et-al(p, $\Im$)
Entrada: Imagem $\Im = (\mathbb{Z}^3, \beta, \omega, B)$ e um ponto $p \in B$.
Saída: 1 se p é simples, 0 caso contrário (algoritmo Saha et al.).

$i \leftarrow$Classifica($N_{26}^*(p)$);
$\delta_i \leftarrow r_{N_6^*(p)}$(Efetivos($N_{26}^*(p)$));
retorne T_{i,δ_i};
Fim-Algoritmo

Algoritmo 3.4: *Algoritmo de verificação de pontos simples Saha et al.*

3.6 Algoritmo de Lee et. al

O algoritmo de afinamento para imagens (26, 6), apresentado em [2] por Lee et al., utiliza-se diretamente da caracterização de pontos simples de Morgenthaler, através de uma implementação bastante eficiente, e da estratégia direcional visando um paralelismo na remoção de pontos.

3.6.1 Caracterização de pontos simples

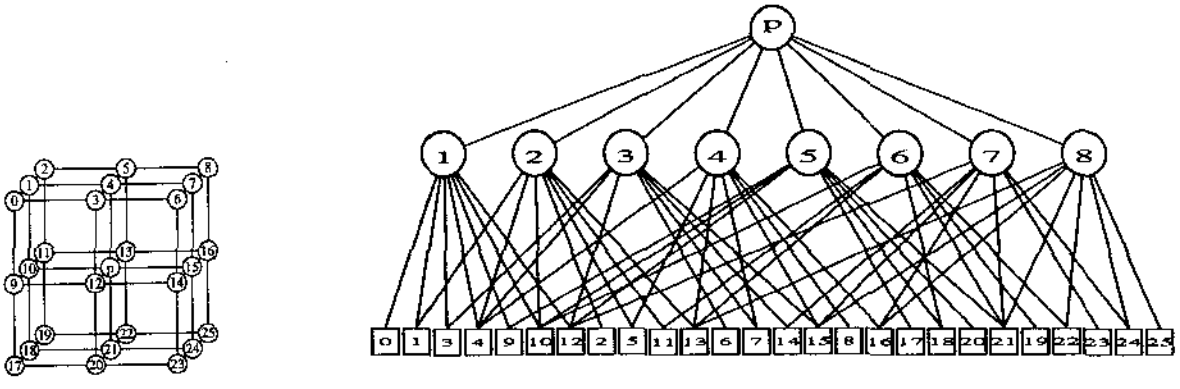
Como visto na seção 3.1, para imagens (26, 6), as condições *i* e *iii* da caracterização 3.2 são suficientes para caracterizar pontos simples. Estas condições são garantidas a partir das consi-

derações sobre a conexidade em $N_{26}^*(p)$ e invariância do número de Euler, discutidas a seguir.

Conexidade em $N_{26}^*(p)$

Sejam $\Im = (\mathbb{Z}^3, 26, 6, B)$ e $p \in B$. A condição i da caracterização 3.2 estabelece a necessidade da igualdade do número de componentes em $B \cap N_{26}^*(p)$ e $B \cap N_{26}(p)$ para a caracterização de p como ponto simples. Como p é adjacente (no sentido de 26-adjacência) a todos os pontos em $N_{26}^*(p)$, temos apenas uma componente em $B \cap N_{26}(p)$. Precisamos portanto, apenas da contagem do número de componentes em $B \cap N_{26}^*(p)$ que deve ser igual a 1.

Lee et al, em [2], utilizam uma estrutura de árvore *octree* para representar as relações de adjacência entre pontos e octantes em $N_{26}(p)$. Um *octante* é um cubo de tamanho $2 \times 2 \times 2$. A estrutura *octree* é uma árvore em que os nós não terminais têm 8 filhos. Para representar a estrutura de $N_{26}(p)$, usamos uma *octree* com três níveis (ver figura 3.15). A raiz representa p, o segundo nível representa os 8 octantes em $N_{26}(p)$ e o terceiro nível representa os pontos em $N_{26}^*(p)$.



(a) (b)

Figura 3.15: *Numeração dos pontos em $N_{26}^*(p)$ e sua representação numa árvore octree (b).*

A contagem do número de componentes em $B \cap N_{26}^*(p)$, utilizando a estrutura *octree*, é feita por um algoritmo de etiquetagem. O algoritmo de etiquetagem consiste de dois procedimentos: $N_{26}(p)$**_Etiquetagem** e **Octree_Etiquetagem** (algoritmos 3.5 e 3.6). O objetivo de $N_{26}(p)$**_Etiquetagem** é etiquetar $N_{26}^*(p)$ visando determinar o número de componentes conexas. Inicialmente, todos os pontos em $N_{26}^*(p) \cap B$ são etiquetados com o valor 0. A seguir, para cada um dos pontos em $N_{26}(p)$, com etiqueta 0, é feita uma chamada ao segundo procedimento **Octree_Etiquetagem**. Este procedimento é usado para etiquetar recursivamente os *octantes* correspondentes a um ponto. A estrutura desta recursão é baseada na árvore *octree* e seus octantes. Para cada octante, há sete pontos que estão sob investigação, p excluído. A ordem de etiquetagem depende do número de *octantes* adjacentes a um ponto em particular. Se o ponto tem etiqueta 0, ele é etiquetado com a etiqueta corrente, esta é incrementada e o octante correspondente é etiquetado pela chamada de **Octree_Etiquetagem** com o índice do octante

e a etiqueta corrente. Ao fim da etiquetagem de todo a 26-vizinhança, o número da etiqueta corrente corresponde ao número de componentes em $N^*_{26}(p) \cap B$.

A figura 3.16 ilustra o funcionamento do algoritmo. A ordem de etiquetagem dos pontos é 0, 9, 21, 11, 2, 16.

Invariante do número de Euler

A estratégia usada para verificar a invariância do número de Euler é semelhante àquela usada por Lobregt [3], diferindo no fato de que o número de consultas à tabela de configurações $2\times2\times2$ é reduzido à metade.

Para o cálculo de $\chi(B \cap N^*_{26}(p))$ (ver seção 2.3, sobre o número de Euler), a chave para cada consulta tem o valor do bit 7 fixo no ponto p (figura 3.17). Isso nos permite saber previamente qual a configuração de cada octante de $N_{26}(p)$ antes e depois da remoção de p. Usando estas configurações é possível calcular qual a contribuição de cada octante para o número de Euler e construir uma nova tabela. Esta tabela (tabela 3.3), assim como a tabela 2.2, possui 256 entradas contendo o valor da diferença entre as contribuições de cada octante antes e depois da remoção de p, para cada configuração de octante possível. Somando-se estas diferenças para cada um dos oito octante em $N_{26}(p)$, teremos o valor de $\chi(B \cap N_{26}(p)) - \chi(B \cap N^*_{26}(p))$ com apenas oito consultas à tabela, metade do número de consultas exigidas pelo algoritmo de Lobregt (secção 3.2.1).

3.6.2 Pontos finais

Pontos finais, no algoritmo de Lee et. al, são definidos em [2] a partir da configuração dos octantes em $N^*_{26}(p)$. Sejam $octante_0, \ldots, octante_7$ os octantes em $N_{26}(p)$. Para esqueletos de superfície, um ponto p é considerado ponto final se $\forall i \in \{0, \ldots, 7\}$ as seguintes condições são válidas:

i. a configuração do $octante_i$ é igual a um dos tipos da figura 3.18. Lembremos que o bit 7 corresponde a p e a ele está associado o valor 1 (ponto de cor preta).

ii. $\#(B \cap octante_i) < 3$

Ponto final para esqueleto de linhas é definido de maneira simples. Todo ponto p tal que $\#(B \cap N^*_{26}(p)) \leq 2$, ou seja, tem no máximo dois 26-vizinhos pretos, é considerado ponto final.

3.6.3 Paralelismo

Em [2], os autores não solucionam completamente o problema de remoção simultânea de pontos simples. O que é feito, em paralelo, é a classificação dos pontos segundo sua topologia, ou seja, os pontos simples são etiquetados. Em seguida, pontos que satisfazem uma condição extra de remoção, testada seqüencialmente para cada ponto, são removidos.

Inicialmente, os pontos simples, em dada direção, são etiquetados. Seja R o conjunto formado por estes pontos. Esta verificação e etiquetagem podem ser feitas em paralelo. A seguir, é realizada uma rechecagem seqüencial da conexidade para os pontos em R. É proposta a seguinte

---

Algoritmo $N_{26}(p)$_Etiquetagem(p, $\Im$)
Entrada: imagem $\Im = (\mathbb{Z}^3, 26, 6, B)$ e $p \in B$.
Saída: número de componentes em $B \cap N^*_{26}(p)$.

ParaTodo $q \in B \cap N^*_{26}(p)$ **faça**
    etiqueta[q] $\leftarrow$0;
Fim-ParaTodo
etiqueta_corrente $\leftarrow$1;
ParaTodo $q \in B \cap N^*_{26}(p)$ **faça**
    Se etiqueta[q] = 0 **então**
        determine octante *oct* de q;
        Octree_Etiquetagem(*oct*, etiqueta_corrente);
        etiqueta_corrente = etiqueta_corrente+1;
    Fim-Se
Fim-ParaTodo
retorne (etiqueta_corrente-1); {*subtrai 1 devido a contagem excessiva*}
Fim-Algoritmo

Algoritmo 3.5: *Algoritmo* $N_{26}(p)$*_Etiquetagem.*

Algoritmo Octree_Etiquetagem(*oct*, etiqueta_corrente)
Entrada: índice do octante e etiqueta.
Saída: cubo etiquetado.

ParaTodo p no octante *oct* **faça**
    Se etiqueta[p] = 1 **então**
        etiqueta[p] $\leftarrow$etiqueta_corrente;
        ParaTodo octante *oct'* adjacente a *oct* **faça**
            Octree_Etiquetagem(*oct'*, etiqueta_corrente);
        Fim-ParaTodo
    Fim-Se
Fim-ParaTodo
Fim-Algoritmo

Algoritmo 3.6: *Algoritmo Octree_Etiquetagem.*

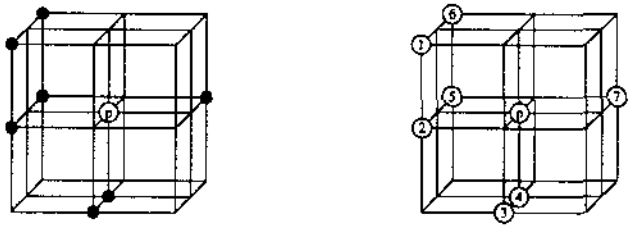


Figura 3.16: *Exemplo para algoritmo de etiquetagem: configuração de $N_{26}^{*}(p)$ (a) e ordem de etiquetagem dos pontos (b). A ordem dos pontos etiquetados é 0, 9, 20, 21, 11, 2 e 16.*

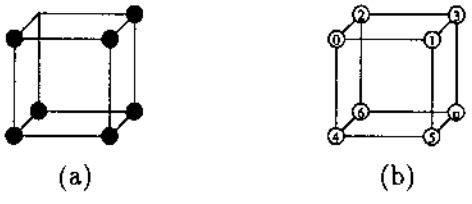


Figura 3.17: *Chave para consulta à tabela de variações das contribuições para o número de Euler: bits associados a cada ponto para exemplo de chave igual a* $(1111\,1011)_2$ *e número do bit associado a cada ponto (b).*

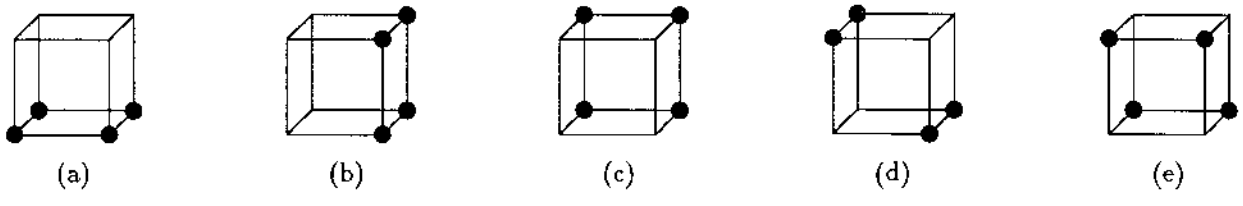


Figura 3.18: *Configurações consideradas para pontos de superfície.*

n	(6, 26)	(26, 6)	n	(6, 26)	(26, 6)	n	(6, 26)	(26, 6)
1	-1	1	87	1	-1	173	1	3
3	1	-1	89	1	3	175	-1	1
5	1	-1	91	3	1	177	1	-1
7	-3	1	93	-1	1	179	-1	1
9	-1	-3	95	-3	-1	181	3	1
11	1	-1	97	-1	1	183	1	-1
13	1	-1	99	1	3	185	1	3
15	-1	1	101	1	3	187	-1	1
17	1	-1	103	3	1	189	3	1
19	3	1	105	-1	5	191	-3	-1
21	3	1	107	1	3	193	-1	-3
23	5	-1	109	1	3	195	1	3
25	1	3	111	-1	1	197	1	-1
27	3	1	113	1	-1	199	3	1
29	3	1	115	-1	1	201	-1	1
31	1	-1	117	-1	1	203	1	3
33	-1	-3	119	-3	-1	205	1	-1
35	1	-1	121	1	3	207	-1	1
37	1	3	123	-1	1	209	1	-1
39	3	1	125	-1	1	211	3	1
41	-1	1	127	-7	-1	213	-1	1
43	1	-1	129	-1	-7	215	1	-1
45	1	3	131	1	-1	217	1	3
47	-1	1	133	1	-1	219	3	1
49	1	-1	135	3	1	221	-1	1
51	-1	1	137	-1	-3	223	-3	-1
53	3	1	139	1	-1	225	-1	1
55	1	-1	141	1	-1	227	1	3
57	1	3	143	-1	1	229	1	3
59	-1	1	145	1	-1	231	3	1
61	3	1	147	3	1	233	-1	5
63	-3	-1	149	3	1	235	1	3
65	-1	-3	151	5	-1	237	1	3
67	1	3	153	1	3	239	-1	1
69	1	-1	155	3	1	241	1	-1
71	3	1	157	3	1	243	-1	1
73	-1	1	159	1	-1	245	-1	1
75	1	3	161	-1	-3	247	-3	-1
77	1	-1	163	1	-1	249	1	3
79	-1	1	165	1	3	251	-1	1
81	1	-1	167	3	1	253	-1	1
83	3	1	169	-1	1	255	1	-1
85	-1	1	171	1	-1			

Tabela 3.3: *Variação das contribuições do número de Euler.*

condição extra para remoção de $p \in R$ na rechecagem: $\#C(\{R \cap N_{26}^*(p)\} \cup \{(B-R) \cap N_{26}^*(p)\}) = 1$. Ora, sabemos da teoria dos conjuntos que $\{R \cap N_{26}^*(p)\} \cup \{(B-R) \cap N_{26}^*(p)\} = B \cap N_{26}^*(p)$ e a condição pode ser reescrita como $\#C(B \cap N_{26}^*(p)) = 1$. Esta checagem é feita seqüencialmente e os pontos em R satisfazendo a condição são removidos. A natureza seqüencia desta checagem que o esqueleto resultante seja diferente, de acordo com a ordem de verificação dos pontos em R e inviabiliza a execução de todo o algoritmo em paralelo.

3.6.4 Esboço do algoritmo de Lee et al.

Apresentamos, a seguir, o esboço do algoritmo. Enfatizamos mais uma vez que apenas parte do algoritmo pode ser executada em paralelo.

Algoritmo Lee-et-al($\Im$)
Entrada: Imagem $\Im = (\mathbb{Z}^3, 26, 6, B)$.
Saída: Esqueleto de $\Im$ (algoritmo Lee et al.).

Repita
pontos_removidos ←0;
ParaTodo $p \in B$ **faça em paralelo**
etiqueta[p] ←"falso";
Fim-ParaTodo
ParaTodo $p \in B$ **faça em paralelo**
Se p é simples **então**
etiqueta[p] ←"verdadeiro";
Fim-Se
Fim-ParaTodo
ParaTodo $p \in B$ **faça**
Se etiqueta[p] **e** $\#\mathcal{C}(N_{26}^*(p)) = 1$ **então**
Remove(p, $\Im$);
pontos_removidos ←pontos_removidos+1;
Fim-Se
Fim-ParaTodo
Até pontos_removidos = 0;
retorne $\Im$;
Fim-Algoritmo

Algoritmo 3.7: *Algoritmo de afinamento Lee-et-al*

3.7 Conclusão

De modo geral, este capítulo apresentou diversos aspectos e problemas inerentes ao afinamento 3-d. Foram descritos os principais algoritmos encontrados na literatura e apresentada, a título de simplificação, sua forma geral de implementação. Esta forma concisa, descrita por nós, visa ajudar o leitor na compreensão do texto original dos respectivos métodos indicados na bibliografia. O capítulo seguinte apresenta uma análise comparativa desses algoritmos.

Capítulo 4

Análise dos Algoritmos

Descrevemos, no capítulo anterior, várias características inerentes ao afinamento 3-d, bem como vários destes algoritmos. No presente capítulo, nós analisamos comparativamente estes algoritmos, estabeleceno equivalências entre eles. Esta análise, fruto de nosso estudo e trabalho prático com os diferentes métodos, é feita, ainda, de maneira muito superficial na literatura [4, 8, 2, 13, 18]. Na seção 4.1, relacionaremos as várias caracterização de ponto simples entre si. Na seção 4.2, são apresentados os testes de todos os algoritmos para imagens (26, 6), descritos no capítulo anterior, e os respectivos resultados obtidos. Discutimos as características geométricas destes algoritmos na seção 4.3. A seção 4.4 discorre sobre os resultados obtidos pelo Esquema de Afinamento para imagens (6, 26) (apresentado na seção 3.4.2).

4.1 Equivalência entre caracterizações de pontos simples

Após o estudo de vários algoritmos de afinamento e caracterização de pontos simples, estabeleceremos, agora, algumas equivalências entre essas caracterizações. Trataremos, nesta seção, apenas das caracterizações para imagens (26, 6).

4.1.1 Equivalência entre Bertrand e Lee-et-al

A argumentação exposta por Lee et al. em [2] ilustra os problemas e discussões existentes em torno da caracterização de pontos simples e, conseqüentemente, da definição de algoritmos de afinamento no espaço 3-d. No referido trabalho, os autores declaram que a caracterização de Bertrand e Malandain em [17], baseada no critério 3.1 (seção 3.1.1) não representa todos os possíveis pontos simples, afirmando que o ponto p na configuração de pontos da figura 4.1 não é classificada como ponto simples. Todavia os autores de [2] interpretaram erroneamente a caracterização proposta por Bertrand. Na realidade, o exemplo apresentado é classificado corretamente pela caracterização 3.3. Em [19], Bertrand e Malandain contra argumentam as

críticas em [2] mostrando que o ponto p do exemplo citado é classificado corretamente por sua caracterização de pontos simples. Porém, não é feita uma discussão profunda sobre o tema.

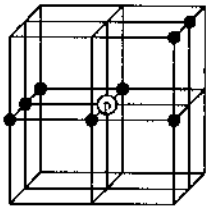


Figura 4.1: *Configuração apresentada por Lee et al. como contra-exemplo para a caracterização de Bertrand e Malandain.*

De fato, mostraremos nesta seção a equivalência entre as caracterizações de pontos simples propostas por Bertrand Malandain em [17] e Lee et al em [2].

Sejam $\Im = (\mathbb{Z}^3, 26, 6, B)$ e $p \in B$. Resumidamente, as caracterizações de Bertrand e Lee podem ser descritas como segue:

Bertrand. p é ponto simples se, e somente se (ver proposição 3.1, seção 3.3)

$$\#C^p(B \cap N_{26}^*(p)) = 1 \tag{4.1}$$
$$\#C^p(\overline{B} \cap N_{18}^*(p)) = 1 \tag{4.2}$$

Lee. Segundo caracterização considerada por Lee et al., p é ponto simples se, e somente se (seção 3.6)

$$\#C(B \cap N_{26}^*(p)) = \#C(B \cap N_{26}(p)) \tag{4.3}$$
$$\chi(B \cap N_{26}^*(p)) = \chi(B \cap N_{26}(p)) \tag{4.4}$$

Teorema 4.1 *(4.1) ⇔ (4.3).*

PROVA: Inicialmente provemos que (4.3) ⇒ (4.1). Seja p tal que a equação (4.3) vale. Em $N_{26}(p)$, p é adjacente a todos os pontos pretos, logo temos apenas uma componente conexa em $B \cap N_{26}(p)$, ou seja, $\#C(B \cap N_{26}(p)) = 1$. Daí, $\#C(B \cap N_{26}^*(p)) = 1$. Portanto, $B \cap N_{26}^*(p)$ possui uma única componente e esta componente é adjacente a p. Concluímos que $\#C^p(B \cap N_{26}^*(p)) = 1$ e (4.1) vale.

Provemos agora que (4.1) ⇒ (4.3). Seja p tal que (4.1) é válida. Usando o raciocínio inverso ao caso anterior, é fácil mostrar que (4.3) vale. ∎

Teorema 4.2 *(4.1) e (4.2) ⇒ (4.4).*

PROVA: Seja p tal que (4.1) e (4.2) são satisfeitas. Devemos mostrar que

$$\begin{aligned}&\#\mathcal{C}(B \cap N_{26}(p)) - \#\mathcal{T}(B \cap N_{26}(p)) + \#\mathcal{H}(B \cap N_{26}(p)) \quad = \\ &\#\mathcal{C}(B \cap N^*_{26}(p)) - \#\mathcal{T}(B \cap N^*_{26}(p)) + \#\mathcal{H}(B \cap N^*_{26}(p))\end{aligned}$$

Pela hipótese e pelo teorema 4.1, temos que $\#\mathcal{C}(B \cap N^*_{26}(p)) = \#\mathcal{C}(B \cap N_{26}(p)) = 1$.

Pela hipótese, sabemos que há uma, e somente uma, componente em $\overline{B} \cap N^*_{26}(p)$ adjacente a p. Isto é possível somente se $\exists q \in \overline{B} \cap N^*_6(p)$, ou seja, existe pelo menos um ponto branco q 6-adjacente a p. Com isso, temos que p é um ponto de borda. Para que houvesse uma cavidade em $B \cap N_{26}(p)$ seria necessário que $N_6 \subset B$, ou seja, todos os 6-vizinhos de p deveriam ser pontos pretos. Como isto não ocorre, $\#\mathcal{H}(B \cap N_{26}(p)) = 0$. Com raciocínio análogo, concluímos que $\#\mathcal{H}(B \cap N^*_{26}(p)) = 0$.

Como p é preto, é fácil notar que não existe túnel em $B \cap N_{26}(p)$, ou seja, $\#\mathcal{T}(B \cap N_{26}(p)) = 0$.

Saha et al. mostraram em [13] que $\#\mathcal{T}(B \cap N_{26}(p)) = 0$ se, e somente se, o conjunto dos 6-vizinhos brancos de p são 6-conectados em $\overline{B} \cap N^*_{18}(p)$. Pela hipótese de (4.2), os 6-vizinhos brancos de p são conectados em $\overline{B} \cap N^*_{18}(p)$. Daí, $\#\mathcal{T}(B \cap N^*_{26}(p)) = 0$.

Com isto, temos que:

$$\begin{aligned}\#\mathcal{C}(B \cap N_{26}(p)) - \#\mathcal{T}(B \cap N_{26}(p)) + \#\mathcal{H}(B \cap N_{26}(p)) &= 1 - 0 + 0 = 1 \\ \#\mathcal{C}(B \cap N^*_{26}(p)) - \#\mathcal{T}(B \cap N^*_{26}(p)) + \#\mathcal{H}(B \cap N^*_{26}(p)) &= 1 - 0 + 0 = 1\end{aligned}$$

∎

Teorema 4.3 *(4.3) e (4.4) ⇒ (4.2).*

PROVA: Seja p tal que (4.3) e (4.4) valem. Pela discussão anterior, sabemos que: $\#\mathcal{T}(B \cap N_{26}(p)) = 0$, $\#\mathcal{H}(B \cap N_{26}(p)) = +\#\mathcal{H}(B \cap N^*_{26}(p)) = 0$. Ademais, pela hipótese de (4.3), $\#\mathcal{C}(B \cap N^*_{26}(p)) = \#\mathcal{C}(B \cap N_{26}(p)) = 0$. Daí, e pela hipótese de (4.4), conclui-se que $\#\mathcal{T}(B \cap N^*_{26}(p)) = 0$.

Saha et al mostraram em [13] que $\#\mathcal{T}(B \cap N_{26}(p)) = 0$ se, e somente se, o conjunto de 6-vizinhos brancos de p é 6-conectado em $\overline{B} \cap N^*_{18}(p))$. Se os pontos em $\overline{B} \cap N^*_{18}(p))$ são 6-conectados em $\overline{B} \cap N^*_{18}(p))$ então temos apenas uma componente branca adjacente a p em $\overline{B} \cap N^*_{18}(p))$, pois qualquer ponto branco em $N_{18}(p)$ que não é 6-conectado a um ponto q qualquer em $\overline{B} \cap N_6(p)$ não é adjacente a p. Daí, concluímos que $\#\mathcal{C}^p(\overline{B} \cap N^*_{18}(p)) = 1$. ∎

Teorema 4.4 *Um ponto é simples segundo Bertrand e Malandain [7] se, e somente se, p é ponto simples segundo Lee et al. [2].*

PROVA: Decorrência imediata dos teoremas 4.1, 4.2 e 4.3. ∎

4.1.2 Equivalência entre Bertrand e Saha et al.

Teorema 4.5 *Um ponto é simples segundo Bertrand e Malandain [7] se, e somente se, p é ponto simples segundo Saha et al. [13].*

Para fins de praticidade, reescreveremos a caracterização 3.6, proposta por Saha et al, da seguinte maneira: p é ponto simples se, e somente se

$$\#(\overline{B} \cap N_6^*(p)) \geq 1 \tag{4.5}$$

$$\#(B \cap N_{26}^*(p)) \neq 0 \tag{4.6}$$

$$\#\mathcal{C}(B \cap N_{26}^*(p)) = 1 \tag{4.7}$$

$$\overline{B} \cap N_6^*(p) \text{ é conexo em } \overline{B} \cap N_{18}^*(p). \tag{4.8}$$

PROVA: As equações (4.6) e (4.7) equivalem à equação (4.1), ao passo que as equações (4.5) e (4.8) são equivalentes à (4.2). ∎

4.1.3 Relação entre Morgenthaler e Tsao-Fu

O algoritmo de afinamento de Tsao e Fu [4], como mostrado no capítulo 3, utiliza-se de restrições baseadas no cálculo do número de Euler e na manutenção da conexidade em duas janelas de checagem para a caracterização de pontos simples. Estas restrições asseguram, ainda, a preservação de topologia para remoção de pontos, em paralelo, quando usada a estratégia direcional. Contudo, estas condições se mostram restritivas à caracterização de algumas configurações de pontos simples.

Podemos sintetizar a caracterização de pontos simples do algoritmo de afinamento proposto por Tsao e Fu da seguinte maneira:

Caracterização 4.1 *Sejam a imagem* $\Im = (\mathbb{Z}^3, 26, 6, B)$ *e o ponto de borda* $p \in D_{norte}(\Im) \cap B$. *p é considerado simples se, e somente se, as seguintes condições forem satisfeitas:*

i. $\chi(N_{26}(p)) = \chi(N_{26}^*(p))$;

ii. é mantida a conexidade nas janelas de checagem P_y *e* P_z *(ver seção 3.2).*

Teorema 4.6 *Se o ponto p é simples segundo Morgenthaler [5], então p é ponto simples segundo Tsao e Fu [4].*

PROVA: Seja $p \in B \cap D_{norte}(\Im)$ tal que a caracterização 4.1 é válida. Comecemos mostrando que a equação (4.8) é válida.

Para uma prova por absurdo, suponhamos que (4.8) não vale para p. Seja $norte(p) = (p_x + 1, p_y, p_z) \in \mathbb{Z}^3$. Então, $\exists q \in \overline{B} \cap N_6^*(p)$ tal que não existe nenhum caminho 6-conexo $\pi = norte(p), \ldots, q$, onde todos os pontos de π estão em $\overline{B} \cap N_{18}^*(p)$. Como cada um dos 6-vizinhos de p difere de p em apenas uma coordenada, é fácil verificar que $N_6^*(p) \in P_y \cup P_z$.

Pelo menos um destes planos contém ambos $norte(p)$ e q. Supondo, sem perda de generalidade, que P_y contém $norte(p)$ e q. Pela hipótese condição (ii) da caracterização 4.1, temos que existe apenas uma componente preta em $P_y - \{p\}$. Na figura 4.2, os pontos $norte(p)$ e q dividem P_y em duas partições S' e S''. Pela hipótese da condição (ii) da caracterização 4.1, há uma e somente uma componente preta em $P_y - \{p\}$, o que indica a existência de um e somente um caminho 6-conexo em $P_y - \{p\}$ que vai de $norte(p)$ a q. Chegamos a uma contradição e, com isso, temos que (4.8) é válida para p.

Os demais itens da caracterização 3.6 são facilmente verificados. ∎

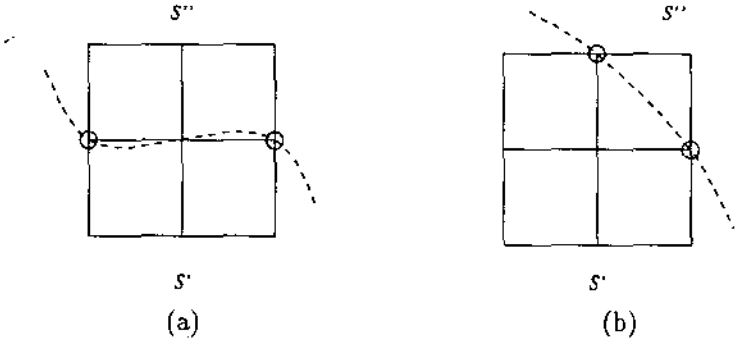


Figura 4.2: *Possíveis configurações para os pontos brancos $norte(p)$ e q em P_y.*

O teorema 4.6 nos garante a preservação da topologia de acordo com os critérios estabelecidos por Morgenthaler no algoritmo Tsao-Fu. Na figura 4.3, vemos um exemplo onde p é ponto simples mas, ainda assim, a janela P_y é desconectada com a remoção de p. Isto significa que a caracterização de Tsao e Fu preserva a topologia mas não consegue identificar todos os possíveis pontos simples de uma imagem 3-d, conduzindo a esqueletos não completamente afinados ou demandando um maior número de iterações ao afinamento.

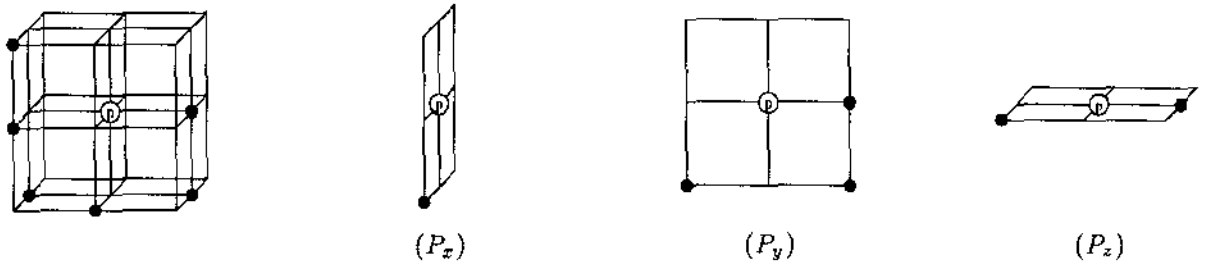


Figura 4.3: *Exemplo onde a remoção do ponto simples p desconecta a janela de checagem.*

As diferentes caracterizações de ponto simples e suas equivalências podem ser vistas, de modo resumido, na tabela 4.1. A coluna 2 da tabela mostra, de modo sucinto, cada uma das caracterizações de pontos simples. A coluna 4 mostra, através dos conjuntos de pontos classificados como simples, por cada caracterização (coluna 3), os relacionamentos entre as mesmas.

Autores	Caracterização usada	Conjunto	Observação
Morgenthaler	$\#\mathcal{C}(B \cap N_{26}(p)) = \#\mathcal{C}(B \cap N^*_{26}(p))$ $\#\mathcal{C}(\overline{B} \cap N_{26}(p)) = \#\mathcal{C}(\overline{B} \cap N^*_{26}(p))$ $\chi(B \cap N_{26}(p)) = \chi(B \cap N^*_{26}(p))$	A	Conjunto completo de pontos simples
Lee et al.	$\#\mathcal{C}(B \cap N_{26}(p)) = \#\mathcal{C}(B \cap N^*_{26}(p))$ $\chi(B \cap N_{26}(p)) = \chi(B \cap N^*_{26}(p))$	B	$A = B = F = C \cap D$
Shirihari et al.	$\#\mathcal{C}(B \cap N_{26}(p)) = \#\mathcal{C}(B \cap N^*_{26}(p))$	C	$A, B, F \subset C$
Lobregt et al.	$\chi(B \cap N_{26}(p)) = \chi(B \cap N^*_{26}(p))$ e 2 planos de checagem	D	$A, B, F \subset D$
Tsao and Fu	$\chi(B \cap N_{26}(p)) = \chi(B \cap N^*_{26}(p))$	E	$E \subset A$
Bertrand	$\#\mathcal{C}(B \cap N^*_{26}(p)) = 1$ $\#\mathcal{C}(\overline{B} \cap N^*_{18}(p)) = 1$	F	$F = A = B$

Tabela 4.1: *Diferentes algoritmos para verificação de ponto simples em imagens (26, 6).*

4.2 Testes e resultados computacionais

Foram realizados testes computacionais para todos os algoritmos descritos no capítulo 3. Os algoritmos foram implementados em linguagem C++ e os testes feitos em *workstation* Sparc 20 sob sistema operacional Solaris. Apesar dos algoritmos de afinamento apresentados poderem ser executados em paralelo, todas as implementações feitas neste trabalho, e discutidas a seguir são seqüenciais, sem que com isso haja alteração nas características dos esqueletos resultantes.

4.2.1 Algoritmos de classificação de pontos simples

Inicialmente, foi feita uma avaliação de performance dos algoritmos de classificação (caracterização) de pontos simples usados pelos diversos algoritmos de afinamento, considerando imagens (26, 6). A avaliação consiste na medição do tempo total gasto por cada algoritmo para classificar um ponto p (como ponto simples ou não simples), para cada uma das 2^{26} possíveis configurações de $N_{26}^{*}(p)$. Foram considerados, para esta avaliação, o algoritmos de Saha et al. e aqueles usados pelos algoritmos de afinamento de Bertrand-Aktouf e Lee-et-al para verificação de ponto simples. O algoritmo de Tsao-Fu não foi incluído nesta avaliação por três razões. Primeiro, a verificação de ponto simples não se distingue claramente do restante do algoritmo, sendo necessárias verificações em planos de checagem as quais determinam também se o ponto é ponto final ou não. Segundo, estas verificações, em planos de checagem, podem ter resultados diferentes (a remoção de um ponto depende da direção sendo verificada pelo algoritmo de afinamento). Terceiro, o conjunto de pontos caracterizados por Tsao-Fu é diferente (como mostramos na seção 4.1.3) daqueles caracterizados pelos algoritmos Bertrand-Aktouf, Lee-et-al e Saha-et-al.

Para os demais algoritmos, os tempos obtidos nos testes são apresentados na tabela 4.2.

Algoritmo	Tempo (em minutos)
Bertrand-Aktouf	473.43
Lee-et-al	174.20
Saha et al.	135.14

Tabela 4.2: *Tempos de execução, em minutos, para os algoritmos de verificação de pontos simples.*

O algoritmo de Saha et al. apresenta melhor resultado. Isto se justifica pelo fato da maior parte da computação necessária para classificação de um ponto ser substituída por consultas a tabelas. Esta substituição reduz grande parte do esforço computacional, visto que as tabelas são previamente calculadas. O preço pago por isso, é o espaço necessário ao armazenamento das tabelas.

O uso de tabelas para cálculo do número de Euler, também é um dos fatores que contribuem para o bom resultado do algoritmo Lee-et-al, embora este resultado seja inferior ao de Saha et

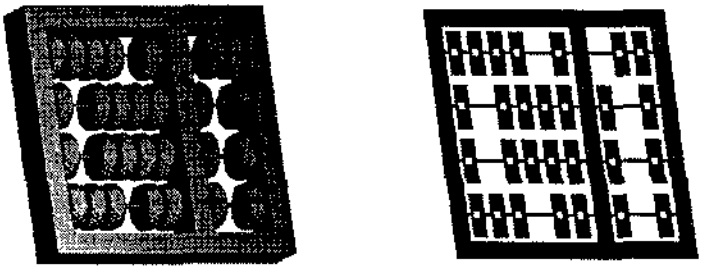
al. Outro fator é a estrutura de dados adequada à 26-vizinhança, usada para cálculo do número de componentes.

Finalmente, o algoritmo Bertrand-Aktouf apresenta o maior tempo de execução quando comparado aos dois anteriores. Neste algoritmo, não são usadas consultas a tabelas e, ao contrário dos demais, todo esforço computacional necessário à classificação de um ponto é realizado durante esta classificação, não havendo pré-cálculo de dados.

4.2.2 Algoritmos de afinamento

Os testes seguintes envolvem os algoritmos de afinamento de Tsao-Fu, Bertrand-Aktouf e Lee-et-al, propriamente ditos. Foram feitos vários testes, com diversas imagens e os resultados são apresentados nas figuras e tabelas a seguir.

Foram utilizados dois grupos de imagens para testes. O primeiro deles corresponde à imagem de um ábaco, apresentado na figura 4.4(a). Esta imagem foi sintetizada por *software*. O plano transversal da figura 4.4(b) mostra que existem cavidades na imagem 3-d, mais precisamente, cada uma das contas do ábaco apresenta uma cavidade. O segundo grupo de imagens é formado por duas imagens de vértebras humanas obtidas através de tomografia computadorizada (figura 4.7). Estas imagens diferem nos ângulos entre as vértebras e os eixos ortogonais, apresentando também diversas cavidades. O efeito do afinamento sobre estas cavidades pode ser visto nos planos dos esqueletos mostrados na figura 4.8.

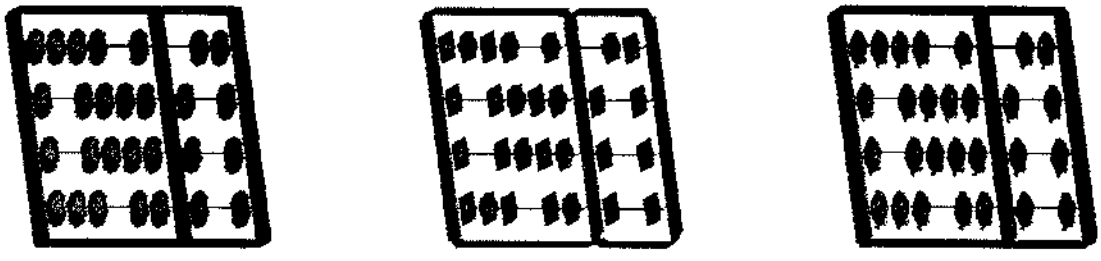


(a) (b)

Figura 4.4: *Imagem 3-d do ábaco (a) e plano transversal central ao mesmo (b).*

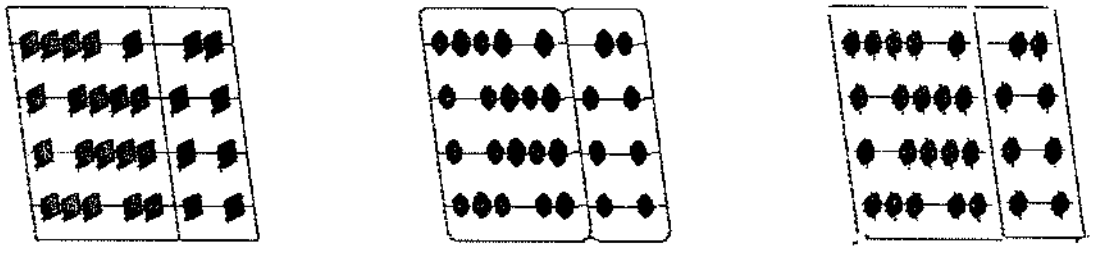
As figuras 4.5, 4.6, 4.9 e 4.10 apresentam os esqueletos de superfícies e de linhas das imagens do ábaco e da vértebra, segundo os algoritmos Tsao-Fu, Bertrand-Aktouf e Lee-et-al. As tabelas 4.3 e 4.4 apresentam os tempos de execução para obtenção de cada um destes esqueletos. No caso das duas vértebras, é apresentada a média dos tempos para o afinamento de cada uma delas.

O algoritmo Tsao-Fu apresenta os maiores tempos de execução, entre os algoritmos testados, para cada uma das imagens. Vários fatores colaboram para isto:



(a) (b) (c)

Figura 4.5: *Esqueletos de superfícies: algoritmos Tsao-Fu (a), Bertrand-Aktouf (b) e Lee-et-al (c).*



(b) (b) (b)

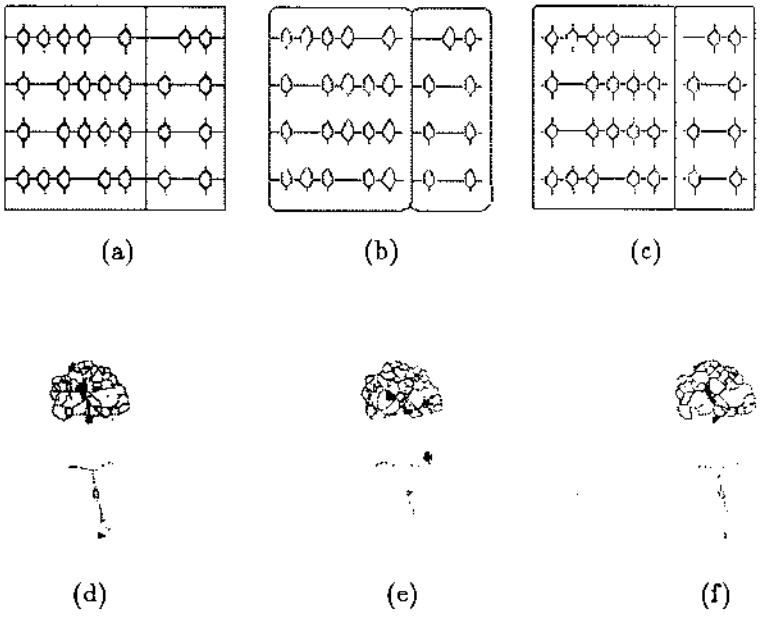
Figura 4.6: *Esqueletos de linhas: algoritmos Tsao-Fu (a), Bertrand-Aktouf (b) e Lee-et-al (c).*

Algoritmo	Esqueleto	
	Superfícies	Linhas
Tsao-Fu	17.55	20.83
Bertrand-Aktouf	10.21	11.58
Lee-et-al	4.56	4.78

Tabela 4.3: *Tempos de execução, em minutos, para afinamento do ábaco.*

(a) (b)

Figura 4.7: *Imagens 3-d de duas vértebras humanas.*



(a) (b) (c)

(d) (e) (f)

Figura 4.8: *Planos transversais aos esqueletos de superfícies do ábaco e da vertebra: Tsao-Fu (a) e (d), Bertrand-Aktouf (b) e (e), Lee-et-al (c) e (f).*

(a) (b) (c)

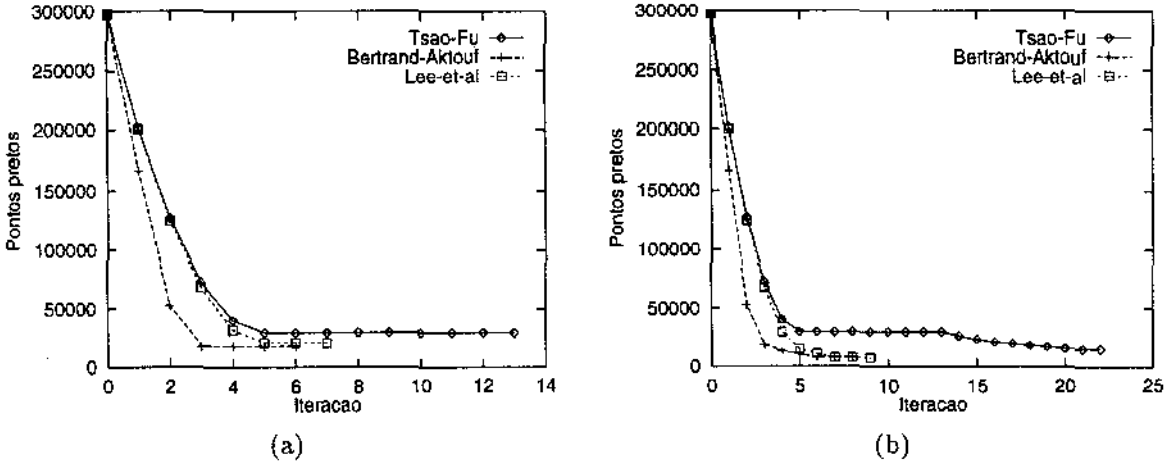
Figura 4.9: *Esqueleto de superfícies: algoritmos Tsao-Fu (a), Bertrand-Aktouf (b) e Lee-et-al (c).*

Algoritmo	Esqueleto	
	Superfícies	Linhas
Tsao-Fu	20.23	27.90
Bertrand-Aktouf	10.68	12.08
Lee-et-al	4.44	4.64

Tabela 4.4: *Tempos de execução, em minutos, para afinamento da vértebra.*

(a) (b) (c)

Figura 4.10: *Esqueletos de linhas: algoritmos Tsao-Fu (a), Bertrand-Aktouf (b) e Lee-et-al (c).*



(a) (b)

Figura 4.11: *Gráficos exemplificando o número de pontos pretos a cada iteração do afinamento do ábaco: esqueleto de superfícies (a) e esqueleto de linhas (b).*

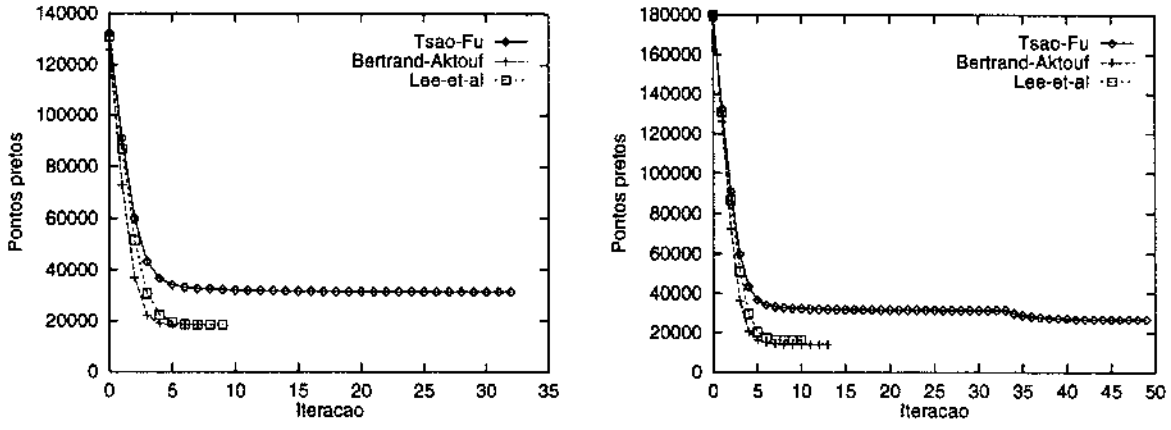


Figura 4.12: *Gráficos exemplificando o número de pontos pretos a cada iteração do afinamento da vértebra: esqueleto de superfícies (a) e esqueleto de linhas (b).*

- cada verificação topológica envolve o cálculo de número de componentes em planos ortogonais ao ponto dado e cálculo do número de Euler, o que demanda um relativo esforço computacional;
- cada iteração é dividida em 6 sub-iterações (estratégia direcional), o que significa que um mesmo ponto pode ter sua topologia verificada várias vezes na mesma iteração (quando o ponto apresenta mais de um 6-vizinho branco);
- o reduzido número de configurações caracterizando pontos simples, que o algoritmo é capaz de reconhecer, obriga a ocorrência de um maior número de iterações para o afinamento.

Ainda de acordo com as tabelas 4.3 e 4.4, o algoritmo de Bertrand-Aktouf é o que apresenta, para as imagens testadas, segundo maior tempo de execução. Neste algoritmo ocorre que:

- há o reconhecimento de todas as configurações de pontos simples, permitindo que o número de iterações necessárias ao afinamento seja menor que no algoritmo de Tsao-Fu;
- a estratégia de sub-campos faz com que cada ponto seja verificado apenas uma vez por iteração;
- cada verificação de ponto simples envolve apenas a contagem do número de pontos em certas vizinhanças do ponto em questão.

Finalmente, o algoritmo de Lee-et-al é o que apresenta os menores tempos de execução, o que atribuímos aos seguintes fatores:

- este algoritmo também é capaz de reconhecer todo o conjunto de configurações de pontos simples;

- apesar de usar estratégia direcional, apresenta um procedimento de verificação topológica que se mostra bastante eficiente quando comparado ao algoritmo de Bertrand-Aktouf (seção 4.2.1).

Número de iterações

É fácil mostrar que o tempo de execução de cada iteração dos algoritmos de Tsao-Fu, Bertrand-Aktouf e Lee-et-al é $O(\#B)$, sendo B o conjunto de pontos pretos da imagem a ser afinada. O tempo de execução total do afinamento é $O(i\#B)$, sendo i o número de iterações total do afinamento (empiricamente, sabemos que i depende de características topológicas e geométricas de difícil determinação). Esta complexidade assintótica de tempo é válida para implementações seqüenciais dos algoritmo.

Através de uma implementação massivamente paralela, com número de processadores proporcional a $\#B$, podemos aproximar o comportamento dos algoritmos pelo modelo de computação paralela PRAM (Parallel Random Acces Memory [20]). Considerando como constante o tempo de verificação topológica e condições de ponto final, para um único ponto, teremos um tempo total para o afinamento de $O(i)$.

Este tempo, proporcional ao número de iterações, motiva uma breve análise do número de passos necessários a cada um dos algoritmos. Nos gráficos apresentados nas figuras 4.12 e 4.11, vemos que o algoritmo Tsao-Fu requer uma quantidade de iterações bem maior que os demais. Esta diferença é mais acentuada ainda para os esqueletos de linhas, devido, principalmente, ao modo como este algoritmo define estes esqueletos: primeiro é obtido um esqueleto de superfícies e, em seguida, nova definição de ponto final é usada na obtenção do esqueleto de linhas (seção 3.2). De um modo geral, o algoritmo de Bertrand-Aktouf requer um número menor de passos na obtenção dos respectivos esqueletos.

4.3 Características Geométricas

Características geométricas são de difícil avaliação, no caso de algoritmos de afinamento 3-d. Estas características devem ter maior ou menor importância conforme a aplicação. Discutiremos as definições de pontos finais e a simetria do esqueleto com relação à imagem original para cada um dos algoritmos estudados, como exemplo de avaliação destas características.

4.3.1 Pontos finais

Nesta seção, nós analisamos a eficácia das definições de ponto final utilizadas pelos algoritmos de afinamento descritos no capítulo anterior.

Esqueletos de linhas

Para o caso de esqueletos de linhas, é desejável que a definição de ponto final preserve as linhas e curvas geradas no afinamento. De fato, os únicos pontos simples que podem ser removidos

destas linhas, durante o afinamento, são os pontos extremos. O objetivo aqui é evitar que estes pontos sejam removidos.

A condição de verificação nos planos de checagem para esqueletos de linhas (seção 3.2.2) torna o algoritmo Tsao-Fu capaz de classificar corretamente os pontos extremos das linhas. Se p é ponto extremo de uma linha afinada, então $\#N^*_{26}(p) = 1$. Com isso, pelo menos um 6-vizinho de p é branco e o número de pontos pretos, em cada plano de checagem, é no máximo 2, garantindo que p será considerado ponto final em pelo menos uma direção.

Todo istmo 1-d (seção 3.3) é considerado ponto final pelo algoritmo Bertrand-Aktouf. Com isso, ocorre a remoção de pontos extremos das curvas, o que, por sua vez, torna necessária a etiquetagem dos istmos 1-d para não remoção nas iterações subseqüentes. Deste modo, uma imagem que possua apenas linhas já afinadas terá os extremos destas linhas removidos e os demais pontos etiquetados como istmos 1-d. Na iteração seguinte, nenhum ponto será removido pois os únicos pontos simples são os novos extremos das linhas etiquetados na iteração anterior (seção 3.3). Concluímos que o algoritmo Bertrand-Aktouf preserva esqueletos de linhas exceto pelos extremos iniciais das linhas.

O algoritmo de Lee-et-al considera um ponto p como ponto final para esqueleto de linhas se $\#N^*_{26}(p) \cap B = 1$ (seção 3.6). Essa definição garante a permanência dos pontos extremos das linhas durante o afinamento.

Esqueletos de superfícies

Para o caso do esqueleto de superfícies, a definição de ponto final não é tão evidente quanto para o caso do esqueleto de linhas. A figura 4.13 mostra alguns exemplos. Em cada um dos exemplos, $p \in B$ é claramente um ponto de superfície se considerarmos imagens (26, 6).

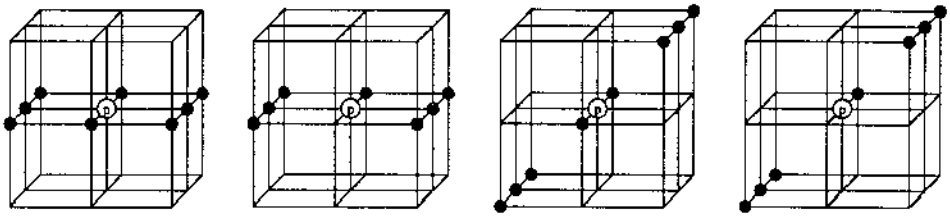


Figura 4.13: *Alguns pontos de superfície para imagens (26, 6).*

O algoritmo de Bertrand-Aktouf considera um istmo 2-d (seção 3.3) como ponto final para esqueleto de superfícies. Uma das condições para que um ponto seja considerado istmo é a de que este não seja ponto simples (seção 3.3.3). Portanto, os pontos de superfície mostrados nas figuras 4.13(b) e 4.13(d) não são considerados pontos finais e são removidos durante o afinamento. Este comportamento de remoção de pontos da borda das superfícies é semelhante ao caso do esqueleto de linhas, onde pontos extremos de curvas são removidos. Mais uma vez, os pontos finais encontrados são etiquetados para evitar remoção nas próximas iterações (seção 3.3).

O algoritmo de Lee-et-al utiliza a classificação dos octantes em $N_{26}(p)$ (seção 3.6.2) para determinar pontos de superfície. Com isto, o algoritmo consegue classificar corretamente o ponto p nas figuras 4.13(a)-(d).

O algoritmo de Tsao-Fu considera também os exemplos 4.13(b) e 4.13(d) como pontos finais, pois todos os planos ortogonais a p são desconectados com a sua remoção, segundo indicado na seção 3.2. E isto impede a remoção de p, segundo mostramos na seção 3.2.

Comportamentos diferenciados dos algoritmos de afinamento contribuem para que os esqueletos gerados por Tsao-Fu e Lee-et-al apresentem número maior de pontos do que aqueles obtidos por Bertrand-Aktouf. Isto pode ser observado nas figuras 4.4 a 4.10 e pelos gráficos da figura 4.12.

4.3.2 Simetria de Esqueletos

Uma maneira de se avaliar a simetria do esqueleto original é verificar quão próximo este se encontra do *eixo medial* da imagem. O eixo medial pode ser compreendido intuitivamente com a partir da figura 4.14 (ver, por exemplo, [21]). Suponha que o interior do retângulo e do círculo estejam repletos de grama e que o restante do terreno não possua qualquer vegetação. Simultaneamente, a borda da grama começa a queimar e o fogo avança de modo uniforme (numa linha de propagação) para o interior da grama. No caso do círculo, o fogo avançará até o centro. No caso do retângulo, o fogo avançará pelos lados até uma linha central do interior. O centro do círculo e a linha central do retângulo constituem o eixo medial destas figuras. A mesma analogia pode ser usada para obter o eixo medial de uma esfera, paralelepípedo ou qualquer sólido no espaço 3-d. De modo mais formal, o eixo medial é o conjunto de pontos eqüidistantes aos dois pontos de borda mais próximos.

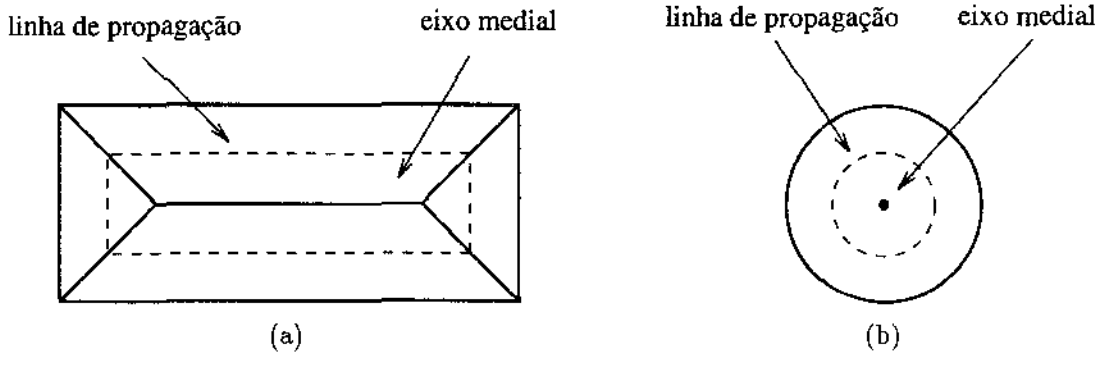
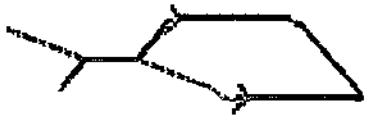


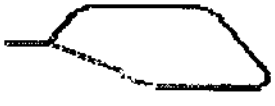
Figura 4.14: *Exemplos de eixo medial.*

A estratégia direcional, onde todos os pontos simples são removidos em direções alternadas, se assemelha à analogia da linha de propagação do fogo em todas as direções.

A figura 4.15 mostra exemplo de esqueletos de linhas obtidos pelos algoritmos de Tsao-Fu, Lee-et-al, Bertrand-Aktouf. Percebemos claramente que o esqueleto da figura 4.15(c), correspondente ao algoritmo de Lee-et-al, apresenta maior número de pontos e uma forma mais próxima

(a)





(b) (c) (d)

Figura 4.15: *Exemplo de esqueletos de linhas de um objeto (a) pelos algoritmos de Tsao-Fu (b), Lee-et-al (c) e Bertrand-Aktouf (d).*

do eixo medial da figura 4.15(a). O esqueleto da figura 4.15(b), algoritmo de Tsao-Fu, é o segundo mais próximo na escala de proximidade com o eixo medial, seguido pelo esqueleto da figura 4.15(d), algoritmo de Bertrand-Aktouf, o de forma mais distante do eixo medial.

Como esperado, os algoritmos Tsao-Fu e Lee-et-al, aqueles que usam estratégia direcional, obtêm esqueletos mais próximos do eixo medial.

(a) (b) (c)

Figura 4.16: *Exemplo de esqueletos de superfícies de um objeto (a) pelos algoritmos de Tsao-Fu (b), Lee-et-al (c) e Bertrand-Aktouf (d).*

Para análise dos esqueletos de superfície, verificamos se as superfícies obtidas passam pelo eixo medial. Seguindo este critério, percebemos que o esqueleto de superfícies obtido pelo algoritmo Tsao-Fu é tão simétrico quanto seu correspondente de linhas. Na verdade, o algoritmo Tsao-Fu constrói o esqueleto de linhas pela remoção de pontos do esqueleto de superfície, como vimos na seção 3.2, o que garante que o esqueleto de superfícies passe por todos os pontos do esqueleto de linhas.

Vemos, pela figura, 4.16 que os esqueletos obtidos por Tsao-Fu e Lee-et-al são bastante semelhantes. Eles apresentam uma superfície afinada de forma bastante próxima do objeto original, enquanto aquele obtido por Bertrand-Aktouf difere pouco dos dois primeiros: apresenta menor número de pontos e superfícies de menor área, o que pode ser uma característica interessante no que concerne o compromisso entre forma de representação e quantidade de dados armazenados.

Estratégia direcional versus subcampos

É importante ressaltar que todos os algoritmos comparados aqui, são algoritmos de remoção de pontos em paralelo. A remoção simultânea de pontos faz com que um grande número de pontos de borda sejam removidos sem que se leve em consideração outras características geométricas da figura e vários pontos sejam considerados indistintamente para remoção em dado instante. Sem dúvida, se a remoção fosse seqüencial, teríamos assimetria no esqueleto devido à ordem

natural de remoção dos pontos. Podemos ver na figura 4.9 que há diferenças entre os esqueletos de superfícies no tocante à suavidade ou regularidade das mesmas. Os esqueletos obtidos pelos algoritmos que se utilizam da estratégia direcional (Tsao-Fu e Lee-et-al) são compostos de superfícies suaves e regulares. Associamos este resultado, mais uma vez, à semelhança entre a estratégia direcional e a linha de propagação da definição de eixo medial, que preserva, no esqueleto, formas da imagem original. Por outro lado, o algoritmo Bertrand-Aktouf, que usa subcampos para remoção simultânea de pontos, fornece esqueletos formados por superfícies mais irregulares e "acidentadas".

4.4 Afinamento para imagens (6, 26)

A seção 3.4 introduz o conceito de ponto P-simples tal como estabelecido por Bertrand em [18]. Naquela seção, é descrito um algoritmo de afinamento para imagens (6, 26) que fornece esqueletos de superfícies. Este algoritmo exemplifica um esquema de afinamento em paralelo, também apresentado naquela seção, e utiliza a caracterização de ponto P-simples para garantir a preservação de topologia. Por se tratar de um algoritmos (6, 26), não o analisamos nas seções anteriores deste capítulo - que tratam de algoritmos para imagens (26, 6) - mas na seção corrente. Apesar da análise em separado, algumas comparações ainda são pertinentes, como veremos a seguir.

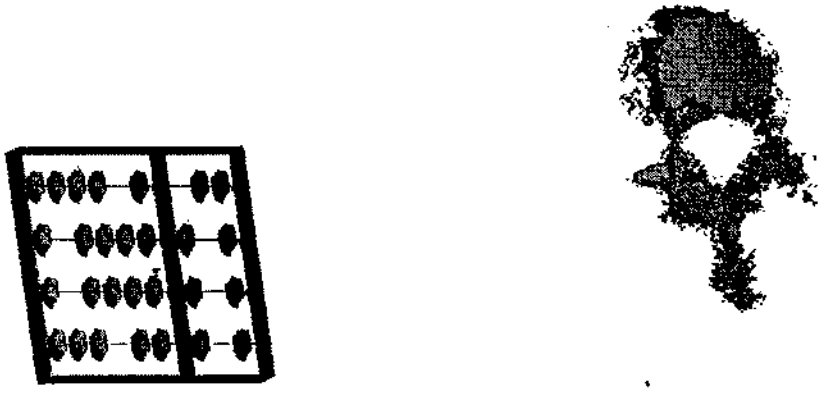
4.4.1 Testes e resultados computacionais

O algoritmo foi testado com as mesmas imagens usadas para os testes dos demais algoritmos: ábaco (figura 4.4) e vértebras (figura 4.7). Os esqueletos de superfícies obtidos são mostrados na figura 4.17. Os tempos de execução são mostrados na tabela 4.5.

Imagem	Tempo (em minutos)
Ábaco (figura 4.4)	16.25
Vértebra (figura 4.7(a))	304.03

Tabela 4.5: *Tempos de execução em minutos.*

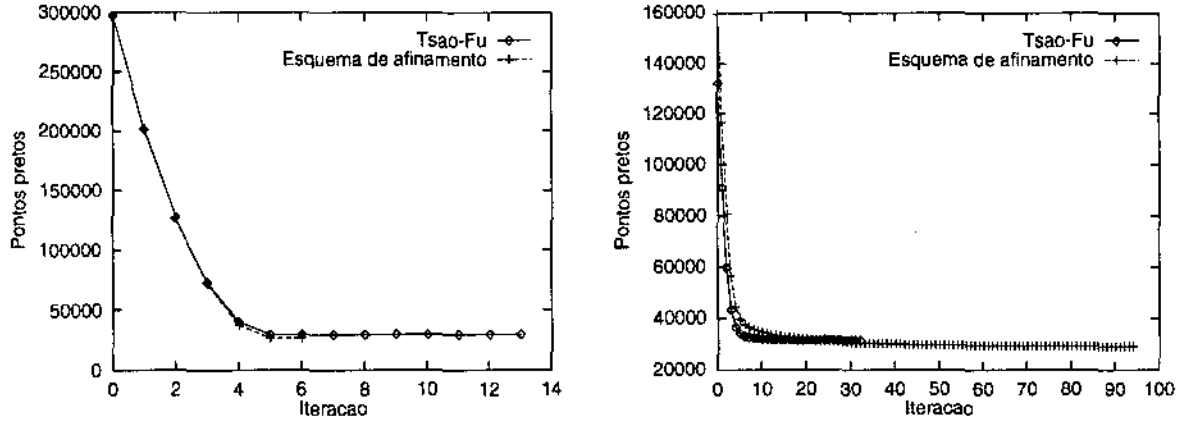
Comparados os tempos dos algoritmos para imagens (26, 6) com os obtidos pelo Esquema de Afinamento sugerido por Bertrand, temos dois resultados bem distintos. O primeiro deles é o resultado do afinamento da imagem do ábaco. O tempo obtido pelo Esquema de Afinamento para esta imagem é comparável ao do algoritmo Tsao-Fu. O número de iterações, porém, é bastante inferior ao daquele algoritmo (figura 4.18(a)). De fato, este resultado se repetiu para outras imagens de formas bem regulares (basicamente blocos sólidos e poucas cavidades). Os tempos de execução se mostram sempre próximos daqueles obtidos pelo algoritmo de Tsao-Fu e o número de iterações é sempre próximo da metade.



(a) (b)

Figura 4.17: *Afinamento do ábaco (a) e da vértebra segundo Esquema de Afinamento para imagens (6, 26) (b).*

O resultado obtido para as imagens das duas vértebras foi bastante diverso deste. Tanto o tempo de execução quanto o número de iterações para o Esquema de Afinamento foram bastante superiores àqueles do algoritmo Tsao-Fu. Atribuímos este comportamento ao tipo de imagem (formas bastante irregulares). Note que mesmo para os algoritmos de afinamento de imagens (26, 6) este é um fator que influencia o tempo de execução e o número de iterações (ver a relação número de pontos pretos por tempo de afinamento para imagens do ábaco e da vértebra). Isto é natural se considerarmos ainda que o algoritmo de verificação de pontos P-simples envolve vários cálculos de números topológicos, além do fato deste ser um método de verificação topológica bastante genérico (pode ser usado tanto para imagens (26, 6) como para (6, 26) e para remoção de pontos em paralelo de qualquer tipo), enquanto os demais algoritmos são mais específicos aos problemas de afinamento de imagens (26, 6).



(a) (b)

Figura 4.18: *Gráfico mostrando o número de pontos removidos, a cada iteração, pelos algoritmos Tsao-Fu e Esquema de Afinamento.*

4.5 Conclusão

Apresentamos, neste capítulo, resultados estabelecendo equivalências entre diferentes caracterizações de pontos simples. Discutimos, respectivamente, nas seções 4.2 e 4.3, resultados de performance e características geométricas dos esqueletos definidos sobre imagens de conexidade (26, 6), descritos a partir dos algoritmos no capítulo anterior. Na seção 4.4, discutimos brevemente os resultados referentes aos testes do algoritmo para imagens (6, 26), exemplificando o uso de pontos P-simples e o Esquema de Afinamento discutido na seção 3.4.

Capítulo 5

Processamento de imagens e técnicas de otimização

Neste capítulo, discutiremos, entre outros, alguns aspectos relativos ao afinamento 2-d que podem ser facilmente estendidos ao caso 3-d. A seção 5.1 apresenta técnicas para reduzir a interferência causada por ruídos sobre os esqueletos. A seção 5.2 introduz e ilustra o afinamento para imagens em níveis de cinza. Por fim, na seção 5.3 são apresentadas técnicas de otimização aplicáveis aos diversos algoritmos vistos no capítulo 3.

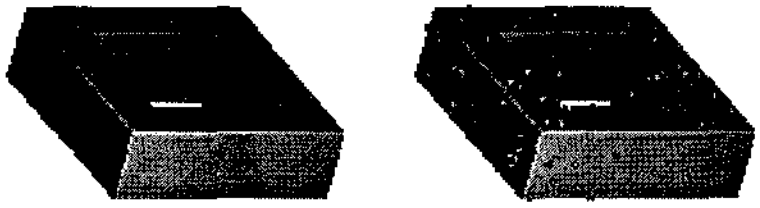
5.1 Conseqüência da presença de ruídos

Analisaremos, nesta seção, o impacto causado nos esqueletos definidos anteriormente, pela presença de ruídos na imagem original, e algumas técnicas para a sua redução.

5.1.1 Sensibilidade a ruídos

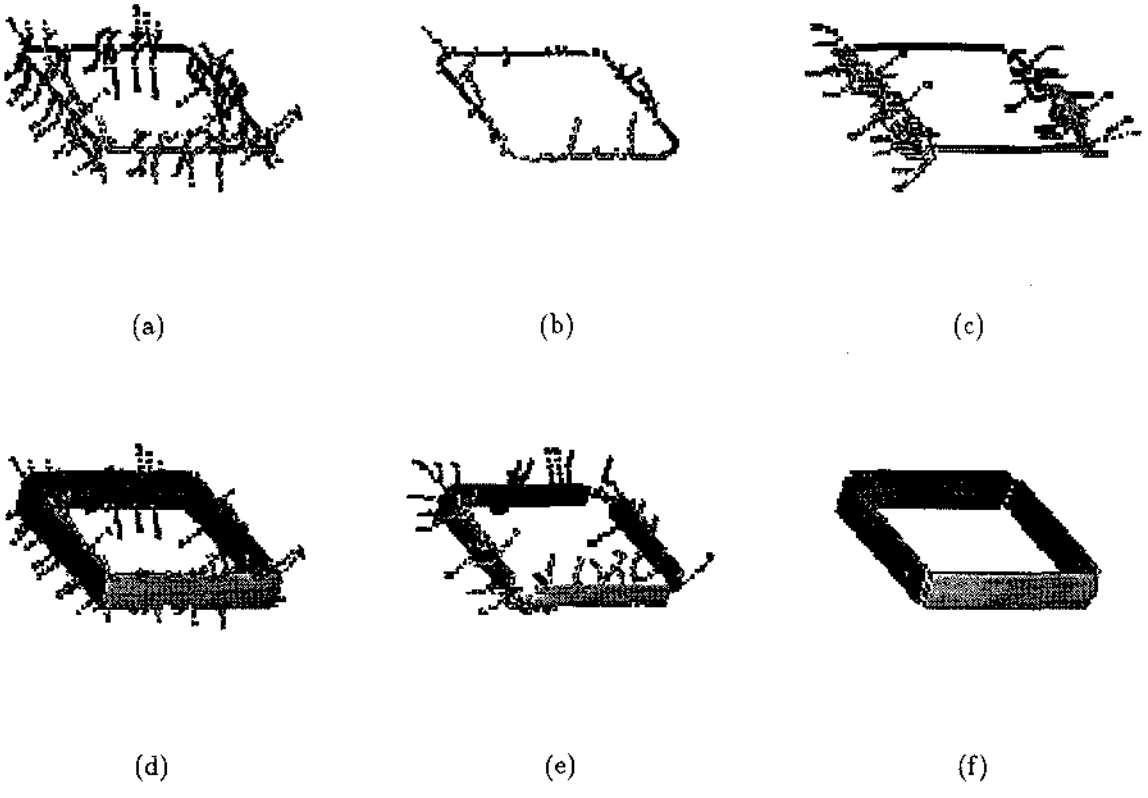
Foram realizados testes para verificar a sensibilidade dos algoritmos a ruídos na imagem original. O objetivo é observar o quanto estes ruídos podem alterar os esqueletos obtidos. Para isto, introduzimos ruídos aleatoriamente em uma imagem de testes (figuras 5.1(a) e 5.1(b)).

O resultado geral é a ocorrência de ramos extras presentes nos esqueletos (figura 5.2). O algoritmo Tsao-Fu é o que apresenta maior sensibilidade e o algoritmo Bertrand-Aktouf o que mostra menor sensibilidade aos ruídos. O algoritmo Lee-et-al apresenta, mais uma vez, resultados que dependem da ordem de verificação dos pontos em sua etapa seqüencial (ver seção 3.6). Esta ordem determina os ramos extras presentes no esqueleto. Diferentes ordens de verificação determinam a permanência de diferentes ramos.



(a) (b)

Figura 5.1: *Objeto original (a) e objeto após inserção de ruído (b).*



(a) (b) (c)

(d) (e) (f)

Figura 5.2: *Esqueletos obtidos a partir de imagem com ruído: algoritmo Tsao-Fu (a)(d), algoritmo Bertrand-Aktouf (b)(e), algoritmo Lee-et-al (c)(f).*

5.1.2 Eliminação de ruídos

Visando a obtenção de esqueletos menos deformados pela presença de ruído, apresentamos algumas técnicas de filtragem baseadas em extensão do caso 2-d [21, 11]. Estas técnicas, discutidas a seguir, permitem a remoção de ruídos das imagens e podem constituir-se de:

i. Pré-processamento

   (a) pré-filtragem homotópica

   (b) pré-filtragem não homotópica

ii. Pós-processamento

   (a) remoção de ramos parasitas

As técnicas de pré-processamento são aplicadas sobre as imagens originais para remoção do ruído, ao passo que as técnicas de pós-processamento são aplicadas sobre os esqueletos a fim de remover ruídos.

Pré-filtragem homotópica

Neste tipo de filtragem, é garantida a preservação da topologia da imagem original na imagem resultante. Deste modo, a filtragem pode remover apenas pontos simples.

Para tanto, selecionamos algumas famílias de máscaras para a 26-vizinhança de um ponto. A figura 5.3 mostra máscaras de cada uma das famílias. As demais máscaras são obtidas por rotação e simetria destas. Podemos perceber que todas estas máscaras correspondem a pontos simples. Assim, o processo de filtragem consiste da verificação, para todo ponto p da imagem, se $N^*_{26}(p)$ corresponde a uma das máscaras. Caso isto ocorra, p é removido da imagem. Este processo pode ser repetido um número n de vezes, representando a *ordem* da filtragem.

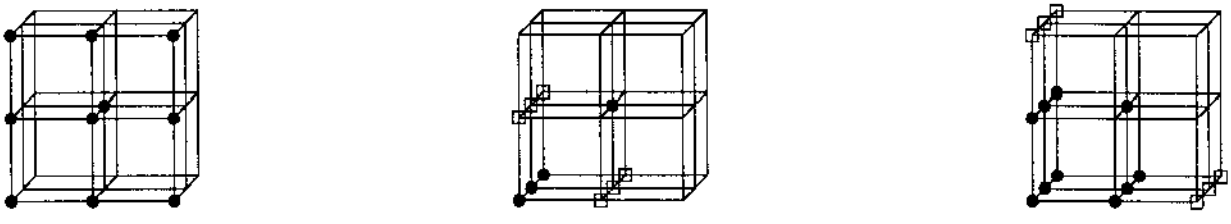


Figura 5.3: *Máscara usadas para filtragem homotópica. Os pontos pretos são representados por* • *e pontos representados por* □ *podem ser brancos ou pretos. Os demais pontos são brancos.*

Como podemos ver, através da figura 5.4, a aplicação destes filtros remove grande parte do ruído mas não completamente, resultando na presença de ramos e superfícies indesejados nos esqueletos. Os esqueletos obtidos pelo afinamento da imagem filtrada são mostrados na figura

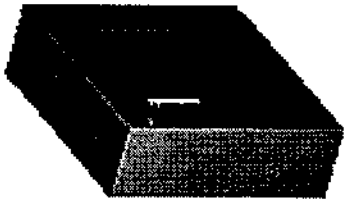


Figura 5.4: *Objeto após filtragem homotópica.*

5.5. Alternativas à permanência de ruído após a filtragem homotópica são a remoção destes ramos e/ou a aplicação de filtros não homotópicos.

5.1.3 Pré-filtragem não homotópica

Existem muitos filtros digitais desenvolvidos com o propósito de suavizar imagens com ruído. Um dos filtros locais mais simples combina operações de máximo e mínimo (veja [11], para exemplos no caso 2-d).

Para imagens 3-d, os operadores de máximo e mínimo são definidos como: $MAX(B) = \{p | p \in B$ ou $p \in N_{26}^{*}(q)$ para algum $q \in B\}$ e $MIN(B) = \{p | p \notin B$ ou $p \in N_{26}^{*}(q)$ para algum $q \notin B\}$. A ordem de aplicação destes operadores resulta em dois tipos de filtros definidos como $FU1^n(B) = MIN^n(MAX^n(B))$ e $FU2^n(B) = MAX^n(MIN^n(B))$, sendo n o número de vezes que cada operador é executado (a *ordem* de $FU1$ e $FU2$). Estes filtros podem ser definidos a partir dos operadores morfológicos de fechamento e abertura, respectivamente [22].

O filtro $FU1^n$ provoca uma expansão dos objetos da imagem, de modo que cavidades e concavidades com espessura menor que $2n$ são eliminadas. O operador $FU2^n$, por sua vez, acarreta um encolhimento dos objetos fazendo com que convexidades menores que $2n$ sejam eliminadas. Estes filtros são similares, ainda, aos operadores de expansão e encolhimento (*expanding* e *shrinking*) discutidos em [11]. Note que, dependendo da ordem do filtro e da imagem a ser processada, podemos ter alteração da topologia, o que demanda um conhecimento prévio do tipo de ruído presente na imagem para uma adequada aplicação.

O filtro $FU1$ não é capaz de remover qualquer ruído na imagem 5.1(b). O filtro $FU2^1$, por sua vez, consegue remover todo o ruído, visto que este é composto por pontos adicionados à borda do objeto e constitui-se apenas de convexidades. Aplicando $FU2^{12}$, por exemplo, o objeto será totalmente removido.

(d) (e) (f)

(d) (e) (f)

Figura 5.5: *Esqueletos obtidos a partir de imagem filtrada: algoritmo Tsao-Fu (a)(d), algoritmo Bertrand-Aktouf (b)(e), algoritmo Lee-et-al (c)(f).*

5.1.4 Remoção de ramos

Como vimos anteriormente, as técnicas acima nem sempre conseguem remover completamente o ruído da imagem, resultando em ramos parasitas no esqueleto. Estes ramos podem ser facilmente removidos a partir de determinados critérios.

A remoção é feita para ramos de tamanho máximo n. Após o afinamento, remove-se todo ponto p que possua apenas um vizinho preto em $N_{26}^*(p)$, isto é, $\#(B \cap N_{26}^*(p)) = 1$. Repetimos esta operação n vezes.

A figura 5.6 exemplifica a remoção de ramos (ordem 10) nos esqueletos mostrados na figura 5.5(a)-(d), obtidos com o algoritmo de Tsao-Fu.

(a) (b)

Figura 5.6: *Exemplo de remoção de ramos dos esqueletos obtidos pelo algoritmo Tsao-Fu.*

5.2 Afinamento de imagens em níveis de cinza

5.2.1 Imagens em níveis de cinza

Nas imagens binárias, é fácil reconhecer os objetos ou forma devido à diferença de sua cor com o fundo ou *background* da imagem. Em imagens em níveis de cinza, o par fundo-objeto não é tão bem determinado e, portanto, não podemos estabelecer convenientemente as relações de adjacência entre pontos da imagem, como na imagem binária. No entanto, observando imagens em níveis de cinza é possível reconhecer os seus objetos que pode, ser mais claros ou mais escuros do que o fundo da imagem.

A informação da estrutura dos objetos não é dada pelos diferentes níveis de cinza, mas pelas relações de vizinhança entre os diferentes elementos da imagem, como no caso binário.

Representação

Uma *imagem em níveis de cinza* é uma função $f : \mathbb{Z}^n \mapsto \mathbb{N}$ que a cada ponto $p \in \mathbb{Z}^n$ associa um valor em $\mathbb{N}$ denominado *nível de cinza* ou simplesmente *nível* de p. Mais uma vez, nos

referiremos apenas a casos onde $n \in \{2, 3\}$.

Mínimo regional

Considerando a função f de nosso interesse contínua, podemos definir um *mínimo* em f da seguinte maneira. Suponha que caminhemos pelo gráfico de f (f no caso bidimensional, ou seja, $n = 2$), considerado-o como uma superfície topográfica. Partindo de um ponto e seguindo um caminho não ascendente, buscando um ponto mais baixo na superfície, dizemos que este ponto é mínimo se não existe caminho descendente possível partindo deste.

Mais formalmente, podemos definir mínimo usando várias *binarizações* ou *tresholds* de f. Chamamos $B_\lambda = \{p \in \mathbb{Z}^n \mid f(p) \leq \lambda\}$ uma *binarização de f no nível λ*. Uma componente B_λ^i de B_λ é dita um mínimo no nível λ se, e somente se, $B_\lambda^i \cap B_\mu = \emptyset$, $\forall \mu < \lambda$.

Esta definição representa um modo de computar os mínimos usando binarizações sucessivas de uma função.

Divisores de água de uma função

Uma outra noção importante em segmentação de imagens e definida a partir de conceitos de Morfologia Matemática, é a de *Linha de Divisores de Água* ou *LDA*, definida para o caso 2-d). A LDA de uma função 2-d pode ser vista como as *zonas de influência* dos mínimos desta função. Vejamos uma abordagem intuitiva para o caso bidimensional (para maiores detalhes, consultar [22, 23]). A todo mínimo é associado uma bacia e uma maneira de determinarmos a extensão desta bacia é inundá-la. Suponha a superfície topográfica correspondente a f com um furo em cada ponto de mínimo. Esta superfície é mergulhada na água. Sempre que uma bacia está na iminência de transbordar, um dique é construído, evitando que isto ocorra. Quando o dique apresentar um caminho fechado, teremos o limite da bacia associada ao mínimo. O conjunto de todos os limites para todos os mínimos é chamado linha de divisores de água da função. A LDA de f é, portanto, uma imagem binária na qual os pontos correspondentes aos limites dos mínimos de f (os diques) são pretos e os demais pontos são brancos.

O caso 3-d é análogo, sendo que cada mínimo tem uma região de influência formada por um volume 3-d em lugar de uma área plana como no espaço 2-d. Os divisores de água são formados por superfícies (3-d) e não mais por linhas (2-d). Ao conjunto de todos os limites de todos os mínimos de uma função $f : \mathbb{Z}^3 \mapsto \mathbb{N}$ denominamos *superfície de divisores de água ou SDA de f* que corresponde, como no caso 2-d, a uma imagem binária. Nesta imagem, cada cavidade está associada a uma região de mínimo em f. Como veremos, a seguir, é possível utilizar o afinamento na extração dos divisores de água de uma função.

5.2.2 Divisores de água e algoritmos de afinamento

O caso bidimensional

O afinamento de imagens em níveis de cinza, para o caso bidimensional, pode ser descrito da seguinte forma [22]. Seja $T = (T_1, T_2)$ um elemento estruturante com duas fases, isto é, uma máscara composta apenas por valores 1 (representado por T_1) e 0 (representado por T_2). O afinamento de f por T é a transformação cujo resultado é uma nova função g definida por:

$$g(p) = \begin{cases} MAX_{q \in T_2}[f(q)] & \text{se, e somente se, } MAX_{q \in T_2}[f(q)] < f(p) \leq MIN_{r \in T_1}[f(r)] \\ f(p), & \text{caso contrário} \end{cases}$$

Serra mostra em [22] que se for usada a classe de elementos estruturantes correta, podemos usar o afinamento como uma maneira simples de se obter os divisores de água de uma função. Esta classe de elementos estruturantes deve preservar a topologia. A figura 5.7 mostra um exemplo de representação de uma família de elementos estruturantes desta classe. Os elementos da família são obtidos através de rotações da referida máscara. Observe que estes elementos estruturantes constituem casos particulares de configurações de pontos simples de uma imagem. De fato, os divisores de água de uma função podem ser obtidos através de um algoritmo de afinamento modificado: são removidos todos os pontos simples até a idempotência, sem qualquer preservação de pontos finais.

$$\begin{bmatrix} 0 & 0 & 0 \\ * & 1 & * \\ 1 & 1 & 1 \end{bmatrix}$$

Figura 5.7: *Exemplo de elemento estruturante que garante preservação de topologia no afinamento 2-d em grade 4-conexa. Os pontos correspondentes a * podem assumir valor 0 ou 1 indiferentemente.*

O caso tridimensional

O algoritmo anterior pode ser facilmente estendido ao caso 3-d a partir de classes de elementos estruturantes que preservam a topologia no caso binário. Como vimos nos últimos capítulos, existem 2^{26} combinações possíveis se considerarmos a vizinhança $3 \times 3 \times 3$, necessária para garantir a informação topológica de uma imagem 3-d. Ao contrário do caso 2-d, onde o número de configurações homotópicas associadas ao afinamento é bastante limitado, o caso 3-d, com suas 2^{26} configurações possíveis, torna difícil a definição completa de classes de elementos estruturantes desse tipo. A estratégia seguinte transforma o problema a nível de verificação topológica de um voxel.

Seja $\lambda = f(p)$ e B_λ a binarização de f no nível λ tal que $B_\lambda = \{q \in \mathbb{Z}^3 \mid f(q) \geq \lambda\}$. Fazendo $T = B_\lambda \cap N_{26}(p)$, podemos reescrever o afinamento em níveis de cinza descrito acima como sendo

a função g, da seguinte forma:

$$g(p) = \begin{cases} MAX_{q \in \overline{T} \cap N_{26}(p)}[f(q)] & \text{se, e somente se, } p \text{ é simples em } T \\ f(p), & \text{caso contrário} \end{cases}$$

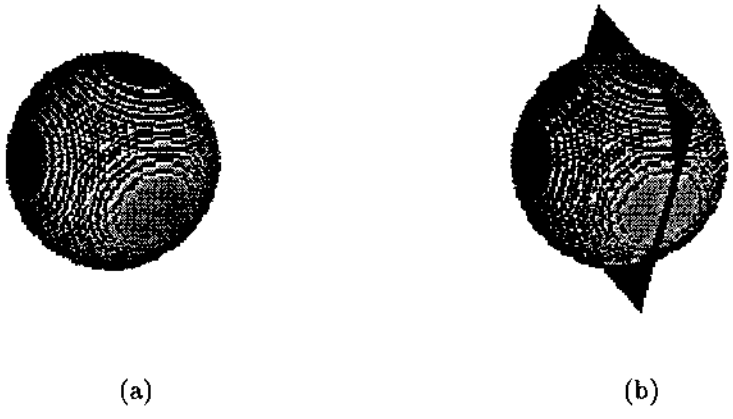
É importante notar que p deve ser simples de acordo com uma das caracterizações para imagens binárias 3-d e assim, podemos obter esqueletos 6-conectados ou 26-conectados.

A utilização do algoritmo acima na definição das SDA de uma imagem 3-d se dá da mesma forma que no caso 2-d: remove-se todos os pontos simples até a idempotência, sem qualquer preservação de pontos finais (no nosso caso, desconsidera-se as caracterizações de pontos finais). Como passo final na obtenção das SDA, é feita a binarização da seguinte forma. Se p caracteriza um ponto de superfície (de acordo com caracterizações vistas anteriormente) em $B_\lambda \cap N_{26}(p)$ (sendo $\lambda = f(p)$), p será um ponto preto na SDA, caso contrário, p será um ponto branco nesta imagem.

5.2.3 Testes e resultados computacionais

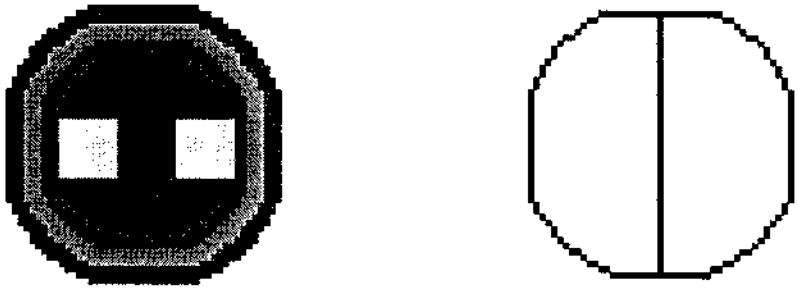
Os testes foram feitos utilizando-se a caracterização de ponto simples de Bertrand e Malandain em [17] e a estratégia direcional, visando obter simetria como característica geométrica dos esqueletos (discutido no capítulo 4). Um ponto p é considerado de direção *norte* se o nível do ponto $(p_x + 1, p_y, p_z)$ é menor do que o nível de p.

As figuras 5.8 e 5.9 mostram exemplo de divisores de água de uma esfera. Através de um plano central à esfera mostrado na figura 5.9(a), podemos perceber que esta possui duas regiões de mínimo em seu interior. O mesmo plano é mostrado, após o afinamento, na figura 5.9(b).



(a) (b)

Figura 5.8: *Esfera (a) e um plano central (b).*



(a) (b)

Figura 5.9: *Planos passando pelo centro da esfera antes (a) e definição da SDA (b). Regiões mais escuras na imagem em níveis de cinza (a) indicam maior nível.*

Um exemplo de afinamento 3-d em níveis de cinza é mostrado nas figuras 5.10 e 5.11. Nestes exemplos, não há remoção de pontos isolados e pontos finais. Um ponto p é considerado ponto final se $\exists q \in N_{26}^{*}(p)$ tal que $f(p) \leq f(q)$, ou seja, existe apenas um ponto em $N_{26}^{*}(p)$ com nível maior ou igual ao nível de p. Esta definição de pontos finais nos leva a esqueletos de linhas. As figuras 5.10(a) e 5.10(b) mostram um objeto e seu esqueleto, respectivamente.

(a) (b)

Figura 5.10: *Objeto original (a) e objeto afinado (b).*

5.3 Técnicas de otimização

Como vimos nos algoritmos apresentados no capítulo 3, para efetuar a remoção de um ponto é necessário realizar verificações de topologia (ponto simples) e de condições de ponto final. Além disto, outras condições, dependentes da aplicação, podem ser necessárias para garantir algumas

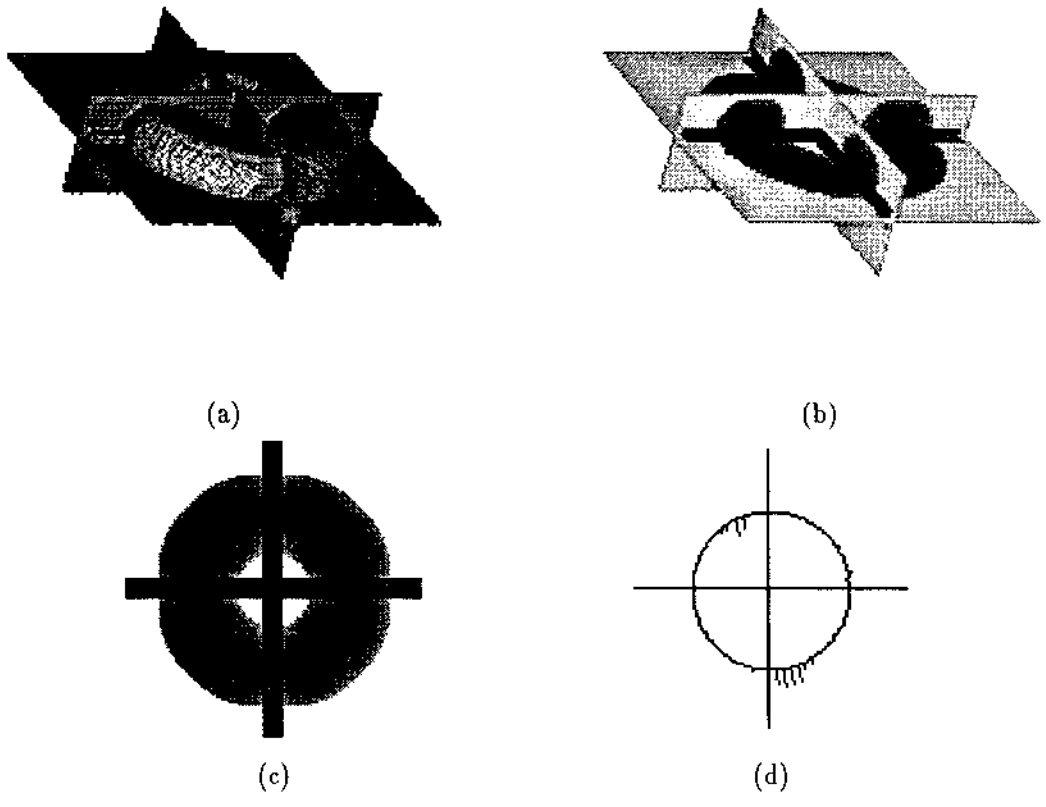


Figura 5.11: *Objeto original com planos ortogonais (a) e cortes correspondentes aos planos ortogonais (b), cortes do objeto antes (c) e após o afinamento (d).*

propriedades desejadas no esqueleto. À verificação de todas estas condições, chamamos *teste de remoção* de um ponto. Quando todas as condições do teste de remoção são satisfeitas, o ponto é dito *removível*, caso contrário, é dito *não removível*. Ainda de acordo com o capítulo 3, testes de remoção são complexos e demandam um tempo considerável. Os algoritmos de afinamento para imagens binárias realizam testes de remoção intensamente. Cada um dos pontos pretos é testado a cada iteração até que seja removido ou até a idempotência. No afinamento em níveis de cinza, o teste continua iterativamente mesmo para pontos que tiveram seus valores alterados em iterações anteriores. As técnicas de otimização apresentadas a seguir visam reduzir o número de testes de remoção.

Evitar testes de remoção desnecessários é uma maneira de otimizar o processo de afinamento. Isto será feito aqui explorando-se as características das condições de ponto final e da caracterização de Morgenthaler (da qual derivam os algoritmos estudados).

Pela caracterização 4.1 (Morgenthaler), a informação necessária à verificação topológica de um ponto p é restrita à sua 26-vizinhança $N_{26}^*(p)$. De acordo com os algoritmos estudados no capítulo 3, a informação necessária para a verificação das condições de ponto final também estão restritas a esta vizinhança. Portanto, toda a informação necessária aos testes de remoção de um ponto p está contida em $N_{26}^*(p)$. As técnicas de otimização apresentadas a seguir baseiam-se nesta característica.

5.3.1 Etiquetagem de pontos viáveis

A idéia desta técnica é simples: em uma dada iteração, evitar testes de remoção para pontos não removíveis, baseada no conjunto de pontos removidos nas iterações anteriores.

Teorema 5.1 *Sejam $\Psi : \mathcal{E} \to \mathcal{E}$ um operador de afinamento e $\Im \in \mathcal{E}$ uma imagem. Sejam $\Psi_i(\Im)$ o conjunto de pontos de $\Im$ após i iterações do operador Ψ sobre $\Im$ e $p \in \Psi_i(\Im)$, para algum $i \geq 1$. Temos que*

$$(\Psi_{i-1}(\Im) - \Psi_i(\Im)) \cap N_{26}^*(p) = \emptyset \Rightarrow p \in \Psi_{i+1}(\Im). \tag{5.1}$$

Ou seja, se p não é removível e não são removíveis os pontos em $N_{26}^*(p)$ durante a iteração i, então p não é removível na iteração $i+1$.

PROVA: Pela discussão anterior, sabemos que a informação necessária ao teste de remoção de p reside localmente em $N_{26}^*(p)$. Até a iteração i, temos os caso em que p é não removível. Visto que esta informação não se altera durante a iteração i, devido a não remoção de pontos em $N_{26}^*(p)$, p continuará não removível durante a iteração $i+1$. ∎

Chamamos de *ponto viável*, após iteração i, todo ponto p tal que p não satisfaz a equação 5.1. O conjunto de todos os pontos viáveis após a iteração i é denotado $V_i(\Im)$. Por definição, $V_0(\Im) = B$.

A cada iteração i, é feito o teste de remoção somente para os pontos viáveis V_{i-1}. Pelo teorema 5.1, temos que os pontos não viáveis em $\Psi_{i-1}(\Im) - V_{i-1}(\Im)$ são não removíveis na iteração i. Portanto, os testes de remoção destes pontos podem ser evitados.

A implementação desta técnica é feita pela etiquetagem dos pontos viáveis. Quando um ponto p é removido, todos os pontos pretos em $N_{26}^{*}(p)$ tornam-se viáveis para a próxima iteração, e são etiquetados como tal. A cada iteração, apenas os pontos etiquetados como viáveis são verificados pelo teste de remoção. Na primeira iteração, todos os pontos pretos são considerados viáveis.

A etiquetagem dos pontos viáveis leva tempo $O(1)$ para cada ponto removido. O mesmo ocorre com a verificação da etiqueta de um ponto, o que significa que a complexidade de tempo de execução dos algoritmos de afinamento não se altera com o uso desta técnica. Cada etiqueta ocupa 1 bit e há duas etiquetas para cada ponto: uma para a iteração corrente e uma outra para a próxima iteração. O espaço necessário para a aplicação da técnica é, portanto, $O(\mathfrak{I})$.

Esboço do algoritmo

O algoritmo 5.8 apresenta um esboço desta técnica que pode ser aplicada a qualquer método de afinamento visto nos capítulos anteriores.

Algoritmo Etiquetagem
Entrada: Imagem $\Im = (\mathbb{Z}^3, \beta, \omega, B)$.
Saída: Esqueleto de $\Im$.

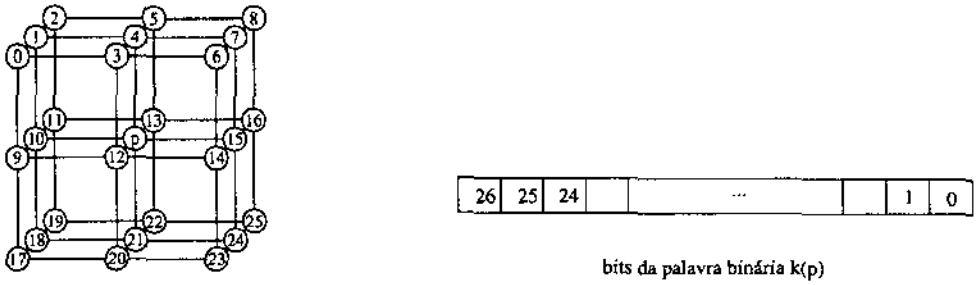
ParaTodo $p \in B$ **faça**
    viável[p] ←"verdadeiro";
Fim-ParaTodo
Repita
    pontos_removidos ←0;
    ParaTodo $p \in B$ **faça**
        viável_prox_iter ←"falso";
    Fim-ParaTodo
    ParaTodo $p \in B$ **faça**
        Se viável[p] = "verdadeiro" **então**
            teste ←Teste_Remoção(p, $\Im$);
            Se teste = "removível" **então**
                Remove(p, $\Im$);
                pontos_removidos ←pontos_removidos+1;
                viável_prox_iter[p] ←"verdadeiro";
            Fim-Se
        Fim-Se
    Fim-ParaTodo
    viável ←viável_prox_iter;
Até pontos_removidos = 0;
retorne $\Im$;
retorne $\Im$;
Fim-Algoritmo

Algoritmo 5.8: *Algoritmo geral de afinamento com etiquetagem de pontos viáveis.*

5.3.2 Dicionário de configurações

Uma maneira simples e eficiente de se evitar o teste de remoção é a que segue. Para cada possível configuração de $N_{26}^*(p)$, definimos um dicionário no qual é armazenado a configuração em questão e o valor 0 se p é removível ou o valor 1 se p é não removível. A chave para acesso ao dicionário $k(p)$ é composta pelos 26 bits correspondentes às cores dos pontos em $N_{26}(p)$, como mostrado na figura 5.12. Obviamente, serão necessários 2^{26} bits para o armazenamento do dicionário, o que inviabiliza seu uso prático na maioria dos computadores disponíveis.

Em uma operação de afinamento típica, uma mesma configuração para 26-vizinhança pode ocorrer para diferentes pontos durante diferentes iterações do algoritmo. Assim, podemos usar



(a) (b)

Figura 5.12: *Chave obtida a partir da configuração de $N^*_{26}(p)$. Associação entre os pontos em $N^*_{26}(p)$ (a) e os bits da chave $k(p)$ (b).*

um dicionário onde são armazenados os resultados dos testes de remoção já realizados. Para cada ponto p, o dicionário é consultado com a chave $k(p)$. Se a chave for encontrada no dicionário, significa que o teste já foi realizado para uma vizinhança de configuração idêntica a $N^*_{26}(p)$. O resultado deste teste está armazenado e, portanto, disponível. Por outro lado, se $k(p)$ não é encontrado no dicionário, é feito o teste de remoção para p e seu resultado é armazenado no dicionário com a chave $k(p)$.

É importante notar que, numa utilização prática, o dicionário deve ter número de entradas limitado por uma constante. Caso o dicionário cresça de modo indeterminado, teremos maior tempo de busca e corremos o risco de nos aproximarmos das 2^{26} entradas possíveis, o que é indesejável.

A implementação eficiente do dicionário é de suma importância para a aplicação desta técnica. Foram implementadas e testadas duas estruturas de dados para o dicionário. Bons resultados foram alcançados utilizando-se uma estrutura de árvore AVL. Árvores AVL são árvores binárias de busca, balanceadas e que permitem realizar operações de inserção e consulta em tempo $O(\log n)$, sendo n o número de nós da árvore (para maiores detalhes sobre árvores binárias de busca e árvores AVL, consultar [24, 25], por exemplo).

A segunda estrutura implementada foi uma variação de tabela de *hashing* (para maiores detalhes sobre hashing, ver [24, 25]) com árvores AVL para tratamento de colisões. A tabela de hashing é indexada por uma função de hashing, que a cada chave $k(p)$, associa um endereço $k'(p)$ na tabela. Neste tipo de estrutura, o número de endereços da tabela é menor que o universo de chaves, ou seja, podemos ter duas chaves $k(p)$ e $k(q)$ tais que $k(p) \neq k(q)$ e $k'(p) \neq k'(q)$. Quando isto ocorre, dizemos que há uma *colisão*, sendo necessário o uso de uma estrutura para armazenar todas as chaves que provocam colisão em cada endereço da tabela. Usamos uma árvore AVL para cada endereço na tabela. Adotamos como função de hashing a função que associa ao ponto p o endereço $k'(p)$ formado pelos 12 bits correspondentes à configuração dos pontos em $N^*_{18}(p) - N^*_6(p)$. Com isso, teremos uma tabela com $2^{12} = 4096$ posições e, conseqüentemente, 4096 árvores AVL para armazenamento de colisões. A palavra binária $k''(p)$, formada pela configuração dos pontos em $N^*_{26}(p) \cap \overline{N^*_{18}(p) - N^*_6(p)}$ (14 bits) é usada como chave secundária para indexar as várias configurações em uma mesma árvore. Note que todas as

configurações em uma árvore possuem chave k' idêntica (figura 5.13).

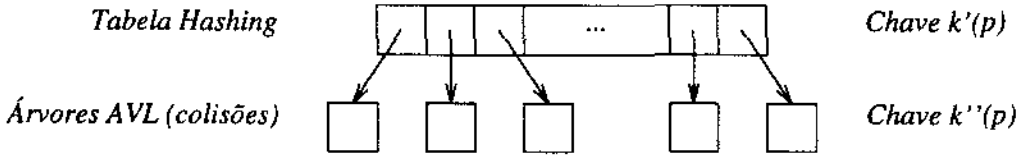


Figura 5.13: *Dicionário implementado por tabela de hashing onde as colisões são armazenadas em árvores AVL.*

Quando usamos somente árvores AVL para implementação do dicionário, cada operação de inserção ou consulta gasta tempo $O(\log n)$, sendo n o número de entradas no dicionário. Como o tamanho do dicionário deve ser limitado por uma constante, consultas e inserções gastarão tempo constante. Portanto, esta técnica não altera a complexidade assintótica dos respectivos algoritmos, visto que o número de pontos verificados em cada iteração também não é alterado. No uso da tabela hashing, acrescenta-se um passo inicial para determinação da árvore onde a chave $k''(p)$ deve ser procurada. Este passo leva tempo $O(1)$, o que leva mais uma vez à não alteração da complexidade assintótica do tempo de execução dos algoritmos.

Esboço do algoritmo

A aplicação do dicionário de configurações em um método geral de afinamento é apresentado no algoritmo 5.9.

Algoritmo Dicionário
Entrada: Imagem $\Im = (\mathbb{Z}^3, \beta, \omega, B)$.
Saída: Esqueleto de $\Im$.

```
Repita
    pontos_removidos ←0;
    ParaTodo p ∈ B faça
        posição ←Consulta(dicionário, k(p));
        Se posição = nil então
            teste ←Teste_Remoção(p, ℑ);
            Se teste = "removível" então
                Remove(p, ℑ);
                pontos_removidos ←pontos_removidos+1;
            Fim-Se
            Insere(dicionário, k(p), teste);
        Fim-Se
        Senão
            Se dicionário[posição] = "removível" então
                Remove(p, ℑ);
                pontos_removidos ←pontos_removidos+1;
            Fim-Se
        Fim-Senão
    Fim-ParaTodo
Até pontos_removidos = 0;
retorne ℑ;
Fim-Algoritmo
```

Algoritmo 5.9: *Algoritmo de afinamento com uso de dicionário de configurações.*

5.3.3 Testes

As técnicas de otimização, etiquetagem de pontos efetivos e dicionário de configurações, foram implementadas sobre os algoritmos de afinamento de imagens binárias (26, 6) e sobre o algoritmo de divisores de águas para imagens em níveis de cinza.

Afinamento de imagens binárias

Foram realizados testes para avaliação do tempo de execução dos algoritmos com cada uma das técnicas de otimização apresentadas. Além da etiquetagem de pontos viáveis e do dicionário de configurações (com duas implementações diferentes: árvore AVL e hashing), implementamos

uma técnica combinada, que é simplesmente o uso simultâneo das duas anteriores (também com duas implementações, cada uma correspondendo a uma implementação do dicionário). As imagens do ábaco e das vértebras (figuras 4.4 e 4.7) foram usadas para os testes.

As tabelas 5.1, 5.2 e 5.3 mostram, respectivamente, os tempos de execução dos algoritmos de Tsao-Fu, Bertrand-Aktouf e Lee-et-al. Nestas tabelas, Dicionário 1 e Combinada 1 correspondem aos tempos de execução das técnicas usando dicionário implementado na forma de árvore AVL enquanto Dicionário 2 e Combinada 2 correspondem aos tempos referentes à implementação do dicionário empregando tabelas hashing.

Como era de se esperar, o algoritmo de Tsao-Fu apresentou as maiores reduções de tempo, com a aplicação das técnicas de otimização, devido a estratégia complexa empregada nas verificações topológicas e de ponto final (seção 3.2). Neste caso, cada teste de remoção evitado pelas técnicas de otimização representa, naturalmente, maior redução no tempo de execução do método.

Técnica	Ábaco				Vértebra			
	Superfícies		Linhas		Superfícies		Linhas	
sem otimização	20.22	(100%)	28.12	(100%)	74.79	(100%)	109.37	(100%)
Etiquetagem	13.89	(68.3%)	14.71	(52.3%)	53.93	(72.1%)	38.52	(35.2%)
Dicionário 1	4.11	(20.3%)	4.24	(15.0%)	50.53	(67.5%)	34.07	(31.1%)
Dicionário 2	4.78	(23.6%)	5.03	(17.8%)	49.05	(65.5%)	35.34	(32.3%)
Combinada 1	4.43	(21.9%)	4.39	(15.6%)	49.50	(66.1%)	33.82	(30.9%)
Combinada 2	4.57	(22.6%)	5.06	(17.9%)	49.31	(65.9%)	33.75	(30.8%)

Tabela 5.1: *Tempos de execução, medidos em minutos, para o algoritmo de Tsao-Fu.*

A técnica de etiquetagem de pontos viáveis, de modo geral, não apresentou melhora significativa no tempo de execução dos algoritmos de Bertrand-Aktouf (tabela 5.2) e Lee-et-al (tabela 5.3). Este resultado é atribuído à estratégia de subcampos, no caso de Bertrand-Aktouf, que ao contrário da estratégia direcional de Tsao-Fu, que remove pontos em diferentes lados dos objetos, subcampos removem pontos geometricamente distribuídos por toda a imagem, levando muitas vezes à etiquetagem de pontos já etiquetados na mesma iteração. O tempo gasto com estas etiquetagens desnecessárias diminui a eficiência da técnica. A técnica do dicionário de configurações com árvores AVL apresentou os melhores resultados para o algoritmo de Bertrand-Aktouf.

O algoritmo de Lee-et-al apresentou as menores reduções de tempo com o uso das técnicas de otimização (tabela 5.3). Isto ocorre devido à maior eficiência deste algoritmo no que concerne os testes de remoção. Novamente, a técnica do dicionário de configurações implementada com árvores AVL apresentou, em geral, melhores resultados que as demais.

De modo geral, a técnica de dicionário de configurações mostrou resultados superiores quando implementada através de árvores AVL, exceto no algoritmo de Tsao-Fu (esqueletos das vértebras). Como conseqüência, a técnica Combinada 1 apresentou resultados melhores

Técnica	Ábaco				Vértebra			
	Superfícies		Linhas		Superfícies		Linhas	
sem otimização	10.21	(100%)	11.54	(100%)	15.62	(100%)	22.78	(100%)
Etiquetagem	10.66	(104%)	11.32	(98.0%)	15.50	(99.2%)	20.61	(90.4%)
Dicionário 1	6.42	(62.8%)	8.02	(69.4%)	13.67	(87.5%)	21.13	(92.7%)
Dicionário 2	6.86	(66.7%)	8.38	(72.6%)	13.93	(89.1%)	21.44	(94.1%)
Combinada 1	7.04	(68.9%)	8.00	(69.3%)	13.50	(86.4%)	18.80	(82.5%)
Combinada 2	7.88	(77.1%)	8.69	(75.3%)	13.77	(88.1%)	19.07	(83.7%)

Tabela 5.2: *Tempos de execução, medidos em minutos, para o algoritmo de Bertrand-Aktouf.*

Técnica	Ábaco				Vértebra			
	Superfícies		Linhas		Superfícies		Linhas	
sem otimização	4.66	(100%)	4.85	(100%)	16.39	(100%)	16.08	(100%)
Etiquetagem	4.51	(96.7%)	5.13	(105%)	16.05	(97.9%)	15.89	(98.8%)
Dicionário 1	3.50	(75.1%)	3.73	(76.9%)	15.88	(96.8%)	15.61	(97.0%)
Dicionário 2	4.02	(86.2%)	3.96	(81.6%)	16.31	(99.5%)	16.02	(99.6%)
Combinada 1	3.93	(84.3%)	3.98	(82.0%)	15.85	(96.7%)	15.72	(97.7%)
Combinada 2	4.66	(100%)	4.42	(91.1%)	17.01	(104%)	16.55	(103%)

Tabela 5.3: *Tempos de execução, medidos em minutos, para o algoritmo de Lee-et-al.*

que Combinada 2, também com exceção do algoritmo de Tsao-Fu. O afinamento da imagem do ábaco apresentou ganhos maiores com esta técnica que o afinamento das vértebras. Isto se explica pelo menor número de configurações armazenadas no dicionário, para o caso do ábaco, provocado pelas características geométricas desta imagem: formas bastante regulares. O afinamento das imagens das vértebras, por sua vez, armazena um grande número de configurações no dicionário, provocando um grande número de testes de remoção (realizados a cada inserção).

Divisores de água de imagens em níveis de cinza

As técnicas de otimização foram implementadas também para o algoritmo de divisores de água em níveis de cinza. Foram feitos testes com as imagens em níveis de cinza de uma vértebra e com a imagem de um crânio obtida através de ressonância magnética (figura 5.14). As figuras 5.14(b) e 5.14(d) mostram exemplos de planos da SDA. Os tempos de execução são apresentados na tabela 5.4.

Técnica	Vértebra		Crânio	
sem otimização	1.94	(100%)	74.30	(100%)
Etiquetagem	1.14	(58.7%)	22.34	(30.0%)
Dicionário 1	1.15	(59.2%)	33.95	(45.6%)
Dicionário 2	1.19	(61.5%)	35.88	(48.3%)
Combinada 1	0.89	(45.8%)	32.76	(44.0%)
Combinada 2	0.91	(47.1%)	34.91	(46.9%)

Tabela 5.4: *Tempos de execução, medidos em horas, do algoritmo de divisores de água de imagens em níveis de cinza.*

5.4 Conclusão

Inicialmente, foram apresentadas, neste capítulo, algumas técnicas de redução de ruídos em esqueletos 3-d definidas a partir de extensões diretas do caso 2-d. A extensão de um algoritmo de afinamento de imagens 2-d em níveis de cinza para o caso 3-d também foi considerada.

Finalmente, este capítulo apresentou técnicas de otimização de fácil implementação para os algoritmos de afinamento de imagens binárias e para os algoritmos de afinamento e de divisores de água de imagens em níveis de cinza. Estas técnicas apresentaram bons resultados, sobretudo no algoritmo de Tsao-Fu e no algoritmo de afinamento de imagens em níveis de cinza.

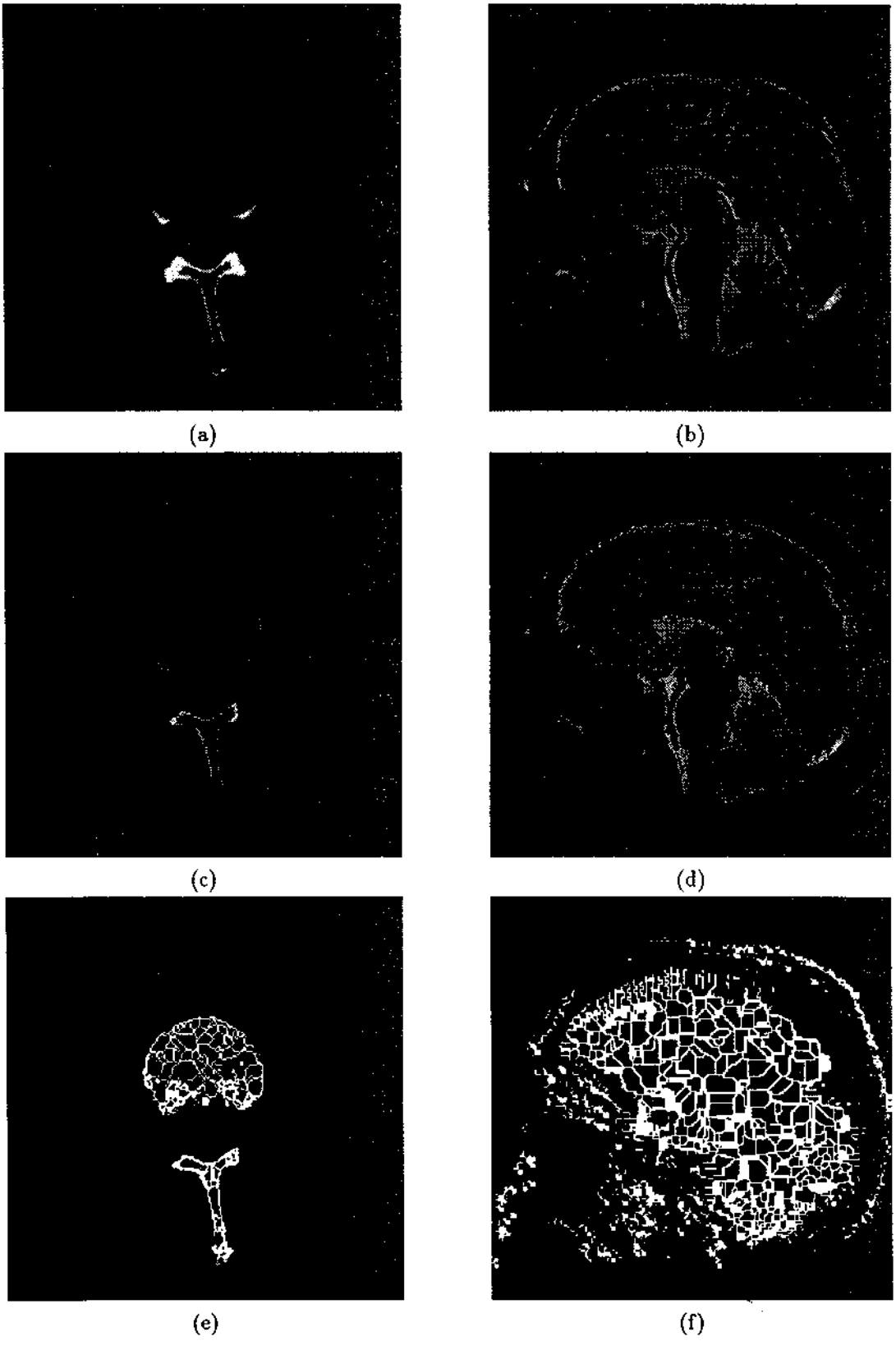


Figura 5.14: *Corte das imagens em níveis de cinza: da vértebra (a), do crânio (b), seus respectivos divisores de água sem binarização (c-d) e após a binarização (e-f).*

Capítulo 6

Conclusões

Este trabalho apresentou um estudo comparativo sobre algoritmos recentes de afinamento de imagens binárias 3-d. Foram apresentados alguns problemas encontrados no desenvolvimento deste tipo de algoritmo, com ênfase na preservação de topologia, problemas estes de solução mais complexa que no caso 2-d, já bastante estudado e conhecido. Em seguida, descrevemos os principais algoritmos de afinamento 3-d encontrados na literatura. Tais algoritmos foram escolhidos pela solução apresentada aos principais problemas envolvidos no afinamento e pela clareza e robustez de suas descrições. Os algoritmos estudados foram implementados e analisados com base em testes referentes a tempo de execução, número de iterações, características geométricas, equivalência entre os métodos, etc.

Na etapa seguinte do trabalho foram descritos alguns tópicos de processamento digital de imagens 2-d aplicáveis ao afinamento 3-d tais como técnicas para redução de ruído nas imagens afinadas e o afinamento de imagens 3-d em níveis de cinza. Finalmente, foram apresentadas técnicas de otimização para os algoritmos estudados, visando redução do tempo de execução. Estas técnicas são de fácil implementação e apresentam bons resultados, principalmente no algoritmo de obtenção dos divisores de água de uma imagem em níveis de cinza.

6.1 Trabalhos futuros

Sugerimos, como continuação natural deste trabalho, o estudo de técnicas de otimização mais eficientes para o afinamento 3-d e a implementação e avaliação dos algoritmos de afinamento em diferentes modelos de arquiteturas paralelas.

Apêndice A

Notação

β, ω	relações de adjacência entre pontos. Geralmente 4-, 6-, 8-, 18- ou 26-adjacência.
$N^*_\beta(p)$	conjunto de pontos β-adjacentes ao ponto p.
$N_\beta(p)$	conjunto formado pelo ponto p e pelos pontos β-adjacentes a p.
$\Im$	imagem binária digital.
$\mathcal{E}$	espaço digital de imagens.
$C(A)$	número de componentes de A.
$C^p(A)$	conjunto de componentes de A adjacentes ao ponto p.
$C^p_\beta(A)$	conjunto de componentes de A adjacentes ao ponto p, usando β-conexidade para os pontos de A.
$T(A)$	número de túneis de A.
$\mathcal{H}(A)$	número de cavidades de A.
A^p_β	conjunto de pontos em $A \cap N^*_\beta(p)$.
$\#A$	cardinalidade do conjunto A.
$\overline{A}$	complemento de A.
$\|a\|$	valor absoluto de a.

Bibliografia

[1] L. Lam, Seong-Whan Lee, and C. Y. Suen. Thinning methodologies - a compreensive survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(9):183–197, 1992.

[2] T. C. Lee, R. L. Kashyap, and G. N. Chu. Building skeleton models via 3-d medial suface/axis thinning algorithms. *CVGIP: Graphical Models and Image Processing*, 56(6):462–478, 1994.

[3] S. Lobregt, W. Verbeek, and F. C. A. Groen. Three-dimensional skeletonization: Principle and algorithm. *IEEE Transactions on Pattern Analyses and Machine Intelligence*, 2(1):75–77, 1980.

[4] Y. F. Tsao and K. S. Fu. A parallel thinning algorithm for 3-d pictures. *Computer Graphics and Image Processing*, 17:315–331, 1981.

[5] T. Y. Kong, A. W. Roscoe, and A. Rosenfeld. Concepts of digital topology: Introduction and survey. *Computer Vision Graphics and Image Processing*, 48:357–393, 1989.

[6] T. Y. Kong, A. Roscoe, and A. Rosenfeld. Concepts of digital topology. *Topology and its Applications*, 46:219–262, 1992.

[7] G. Bertrand and G. Malandain. A new topological classification of points in 3d images. *2nd European Conference in Computer Vision*, pages 710–714, 1992.

[8] G. Bertrand and Z. Aktouf. A three-dimensional thinninlg algorithm using subfields. *Proceedings of the SPIE Conference on Vision Geometry*, 1994.

[9] Solomon Lefschetz. *Introduction to topology.* Princeton University Press, 1949.

[10] Cherng Min Ma. On topology preservation in 3d thinning. *CVGIP: Image Understanding*, 59(3):328–339, 1994.

[11] A. Rosenfeld and Avinash C. Kak. *Digital Picture Procesing*, volume 2. Academic Press, second edition, 1982.

[12] G. Malandain and G. Bertrand. Fast characterization of 3d simple points. *Proceedings of the 11th International Conference on Pattern Recognition*, C:232–235, 1992.

[13] P. K. Saha, B. B. Chauduri, B. Chanda, and D. Dutta Majumber. Topology preservation in 3d digital space. *Pattern Recognition*, 27(2):295–300, 1994.

[14] S. N. Srihari, J. K. Udupa, and M. M. Yau. Understanding the bin of parts. *Proceedings of the IEEE Conference on Decision Contr.*, Dec 1979.

[15] Y. F. Tsao and K. S. Fu. A 3d parallel skeletonwise thinning algorithm. *Proceedings of the IEEE Pattern Recognition Image Processing Conference*, pages 378–683, 1982.

[16] Z. Guo and W. Hall. Parallel thinning with two-subiteration algoriths. *Image Processing and Computer Vision*, 32(3):259–373, 1989.

[17] G. Bertrand. Simple points, topological numbers and geodesic neghborhoods in cubic grids. *Pattern Recognition Letters*, 15:1003–1011, 1994.

[18] G. Bertrand. P-simple points: a solution for parallel thinning. *5th Conference on Discrete Geometry for Computer Imagery*, 1995.

[19] G. Bertrand and G. Malandain. A note on 'Buiding skeleton models via 3-d medial surface/axis thinning'. *CVGIP: Graphical models and image processing*, 57(6):537–538, 1995.

[20] Joseph Jájá. *An introduction do parallel algorithms.* Addison-Wesley Publishing Company, Inc, 1992.

[21] William K. Pratt. *Digital Image Processing*, chapter 15. John Wiley & Sons, Inc, second edition, 1991.

[22] J. P. Serra. *Image Analysis and Mathematical Morphology.* 1982.

[23] S. Beucher and F. Meyer. *Mathematical Morphology in Image Processing*, chapter 12. Marcel Dekker, Inc., 1993.

[24] Thomas H. Cormem, Charles E. Leiserson, and Ronald L. Rivest. *Introduction to Algorithms.* MIT Press, 1990.

[25] Michael Folk and Bill Zoellick. *File Structures.* Addison-Wesley Publishing Company, Inc, second edition, 1992.