Algoritmos de afinamento tridimensional: exemplos de técnicas de otimização

Francisco N. Bezerra *Neucimar J. Leite*

Relatório Técnico DCC–96-13

Outubro de 1996

Algoritmos de afinamento tridimensional: exemplos de técnicas de otimização *

Francisco N. Bezerra Neucimar J. Leite

Sumário

Some algorithms have been proposed to the problem of thinning 3D binary images. These algorithms use homotopic removal of points to achieve an skeleton whose structure preserves the topology of the original image. This work deals with technics for the optimization of such algorithms. Briefly, we present two methods that avoid unnecessary topology checking of the image points, leading to less time consuming sequential thinning.

Key words: 3D binary images, thinning algorithms, digital topology, optimization

1 Introdução

Nas últimas décadas, diversas tecnologias geradoras de imagens tridimensionais têm sido aperfeiçoadas. Como exemplo, temos a tomografia auxiliada por computador, ressonância magnética e ecocardiografia. Com a evolução destas tecnologias e o crescimento natural do volume de informações das imagens, torna-se necessário o desenvolvimento e aperfeiçoamento de técnicas de análise e representação das mesmas. Algumas técnicas de análise de imagens podem conduzir a uma etapa de reconhecimento de padrões tridimensionais (por exemplo, órgãos humanos e estruturas de DNA).

Afinamento é uma operação de erosão homotópica sobre uma imagem gerando um esqueleto como resultado ([Kong et al. (1989)], [Rosenfeld-Kak (1982)], [Lam et al. (1992)]). O esqueleto é uma nova imagem que preserva as características topológicas da imagem original. A obtenção do esqueleto é, portanto, uma forma de reduzir a quantidade de informação da imagem sendo, assim, útil a uma representação compacta da mesma e, conseqüentemente, ao reconhecimento de padrões.

Recentemente, alguns algoritmos têm sido propostos para o afinamento 3D ([Lobregt et al. (1980)], [Tsao-Fu (1981)], [Bertrand-Aktouf (1994)], [Saha et al. (1994)]). Como as imagens tratadas pelo afinamento são geralmente de tamanho muito grande, torna-se necessário o desenvolvimento de técnicas de otimização destes algoritmos para que seu uso seja viável diante de grandes massas de dados.

O presente trabalho apresenta duas técnicas para otimização de algorimos de afinamento 3D que podem ser usadas em separado ou conjuntamente. Essas técnicas foram testadas em algoritmos de afinamento 3D existentes.

*Este trabalho está sendo desenvolvido com o apoio da FAPESP e CNPq.

Na seção seguinte, definiremos alguns conceitos básicos sobre relações de adjacência e representação formal de imagens binárias 3D. A seguir, na seção 3, abordamos algumas considerações sobre afinamento pertinentes ao nosso trabalho e apresentaremos, na seção 4, alguns algoritmos de afinamento 3D. A seção 5 aborda, especificamente, algumas técnicas de otimização associadas aos algoritmos considerados na seção 4. Finalmente, testes e conclusões serão apresentados nas seções 6 e 7.

2 Definições básicas

Apresentamos, a seguir, algumas relações básicas de adjacência no espaço tridimensional. Para maiores detalhes, consultar, por exemplo, [Kong et al. (1989)], [Kong-Rosenfeld (1992)] e [Ma (1994)].

Dizemos que dois pontos $p = (p_x, p_y, p_z)$ e $q = (q_x, q_y, q_z) \in \mathbb{Z}^3$, são:

26-adjacentes: quando $|p_x - q_x| \leq 1$, $|p_y - q_y| \leq 1$, $|p_z - q_z| \leq 1$;

18-adjacentes: quando são 26-adjacentes e $|p_x - q_x| + |p_y - q_y| + |p_z - q_z| \leq 2$;

6-adjacentes: quando $|p_x - q_x| + |p_y - q_y| + |p_z - q_z| \leq 1$.

Os pontos α-adjacentes a um ponto p são chamados α-vizinhos de p. Denotamos $N^*_\alpha(p)$ o conjunto de pontos formado pelos α-vizinhos de p e por $N_\alpha(p)$ o conjunto $N^*_\alpha(p) \cup \{p\}$, $\alpha \in \{6, 18, 26\}$.

Uma *imagem binária digital* (ou *imagem digital*, ou simplesmente *imagem*) $\Im$ é uma quádrupla $(\mathbb{Z}^n, \beta, \omega, B)$ onde cada elemento em $\mathbb{Z}^n$ é chamado de um ponto de $\Im$ e a ele está associado o valor 1 (um *ponto preto*) ou o valor 0 (um *ponto branco*); dois pontos pretos são *adjacentes* se são β-adjacentes; 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 $\Im$. $\Im$ é também chamada uma *imagem n-dimensional* (β, ω). Neste trabalho, consideraremos apenas imagens tridimensionais, ou seja, $n = 3$.

Seja P um conjunto de pontos de uma imagem $\Im$. Um *caminho* de P é uma seqüência $< p_1, ..., p_k >$ tal que cada $p_i \in P$ e p_j é adjacente a p_{j+1}, $1 \leq j < k$. Um *caminho preto* de P é um caminho constituído de pontos pretos de P. Dois pontos p e q estão *conectados* se existe um caminho preto $< p = p_1, ..., p_k = q >$ de P. Uma *componente preta* de P é um subconjunto conectado maximal do conjunto de pontos pretos de P; uma *componente branca* de P é definida analogamente. Uma *cavidade* em uma imagem 3D é uma componente branca finita. Uma esfera "oca", por exemplo, possui apenas uma cavidade.

Uma outra estrutura específica que ocorre em imagens 3D é o *túnel*. Não há uma definição precisa para túnel na literatura, embora uma idéia intuitiva possa ser dada ([Kong et al. (1989)]). Por exemplo, um toro sólido possui um túnel enquanto uma esfera não possui túneis. Mesmo não havendo definição conhecida de túnel, o número de túneis em uma imagem pode ser calculado e, juntamente com o número de componentes e o número de cavidades da imagem, pode ser utilizado para verificar a topologia da mesma.

Denotamos o número de componentes pretas, o número de cavidades e o número de túneis de um conjunto de pontos $P \subseteq \mathbb{Z}^3$ respectivamente por $\#_b(P)$, $\#_c(P)$ e $\#_t(P)$.

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. A *remoção* de um ponto p é a transformação que muda o valor de p de 1 (preto) para 0 (branco).

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 paradoxos de conexidade, ver [Kong et al. (1989)]. 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 26), ou vice-versa.

3 Considerações sobre afinamento 3D

Um algoritmo de afinamento atua sobre a imagem removendo iterativamente os pontos de borda. A cada iteração, são considerados apenas os pontos cuja remoção preserva a topologia da imagem.

Consideraremos, inicialmente, imagens bidimensionais na análise de preservação de topologia. Durante transformações com remoção de pontos de uma imagem 2D, uma componente preta pode ser dividida em duas ou mais componentes, ou pode até ser totalmente removida. Do mesmo modo, componentes brancas podem ser criadas ou destruídas, alterando a topologia da imagem.

Propriedade 1 *Seja* $\Im = (\mathbb{Z}^2, \beta, \omega, B)$ *uma imagem e* $D \subset B$ *um conjunto de pontos pretos de* $\Im$. *A remoção de D preserva topologia de* $\Im$ *se as seguintes condições forem satisfeitas ([Ma (1994)]):*

- *nenhuma componente preta de* $\Im$ *é dividida em 2 componentes pretas de* $\Im - D$;
- *nenhuma componente preta de* $\Im$ *é completamente removida;*
- *não há componentes brancas de* $\Im$ *unidas em uma só componente branca de* $\Im - D$;
- *não são criadas componentes brancas em* $\Im - D$.

Estas condições não são suficientes para mostrar que a remoção de um conjunto de pontos de uma imagem 3D preserva a topologia. Considere as imagens da figura 1. Seja a imagem 1(b), obtida a partir da imagem 1(a) pela remoção do ponto p_1. Esta transformação não preserva topologia, embora todas as condições acima sejam satisfeitas. Neste caso, um túnel é criado. Na transformação da imagem 1(b) na imagem 1(c), pela remoção do ponto p_2, também não há preservação de topologia pois um túnel é destruído e, novamente, todas as condições acima são satisfeitas.

O número de Euler de uma imagem $\Im$ é definido por $\mathcal{X}(\Im) = \#_b(\Im) + \#_c(\Im) - \#_t(\Im)$.

Propriedade 2 *Dizemos que a remoção de um ponto p preserva a topologia de uma imagem 3D* $\Im = (\mathbb{Z}^3, \beta, \omega, B)$ *se, e somente se:*

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

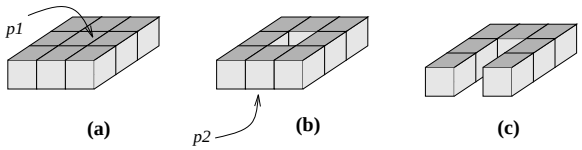

Figura 1: *Ocorrência de túneis. Cada ponto preto é representado por um cubo.*

ii. p *é adjacente a apenas uma componente branca em* $N^*_{26}(p)$*;*

iii. $\mathcal{X}(\Im \cap N_{26}(p)) = \mathcal{X}(\Im \cap (N^*_{26}(p))$*.*

Neste caso, p é chamado *ponto simples*, ver [Ma (1994)]. Existem outras caracterizações para pontos simples ([Bertrand-Aktouf (1994)], [Lobregt et al. (1980)] e [Saha et al. (1994)].

O afinamento pode levar a dois tipos de esqueletos: esqueleto de superfícies e esqueleto de linhas (ver figura 2). Para obtermos tais esqueletos é necessário evitar a remoção de *pontos finais*. No caso do esqueleto de linha, os pontos adjacentes a apenas um ponto preto serão considerados pontos finais. Para esqueletos de superfície, a caracterização de ponto final é mais complexa e depende do algoritmo proposto, veja [Tsao-Fu (1981)] e [Bertrand-Aktouf (1994)], por exemplo.

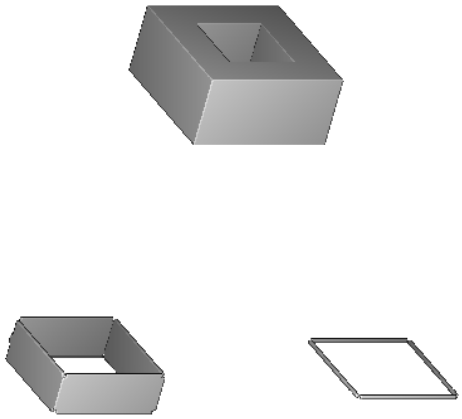

Figura 2: *Objeto (acima) seguido de seus esqueletos de superfícies (esquerda) e de linhas (direita).*

4 Trabalhos anteriores

Descrevemos, a seguir, dois algoritmos de afinamento considerados, inicialmente, neste trabalho. A escolha destes algortimos se deve, entre outros, à melhor adequação das caracte-

rizações de ponto simples à implementação das mesmas, e à robustez na sua formalização. Como veremos, estes métodos podem ser divididos em duas versões: uma seqüencial representada pela implementação direta das caracterizações de ponto simples e ponto final, e uma paralela associada ao modelo de computação SIMD (Single Instruction - Multiple Data streams) representado, neste caso, por arquiteturas maciçamente paralelas tridimensionais ([Jackson]). Naturalmente, neste trabalho, são consideradas apenas versões seqüenciais destes algoritmos.

4.1 Tsao e Fu

Em [Tsao-Fu (1981)], o algoritmo proposto por Lobregt et. al [Lobregt (1980)] para verificação da topologia de um ponto é tomado como base para um algoritmo de afinamento de imagens (6, 26) ou (26, 6) com remoção de pontos em paralelo.

A verificação da preservação de topologia é feita considerando-se que um ponto pode ser removido se sua remoção não altera a conexidade da imagem. A conexidade de uma imagem $\Im$ é dada pelo número de Euler. O número de Euler é calculado para a vizinhança N_{26} do ponto, de maneira eficiente, antes e depois da remoção do ponto. Se o seu valor permanece inalterado, então o ponto é removível (ponto simpes).

Porém, essa estratégia de verificação não garante preservação de topologia quando pontos são analisados e removidos em paralelo. Foi então proposta uma verificação adicional.

Seja $p = (p_x, p_y, p_z)$ um ponto preto. Dizemos que p é um ponto na direção Norte se $q = (p_x + 1, p_y, p_z)$ é um ponto branco. Analogamente, dizemos que p é um ponto na direção Sul, Leste, Oeste, Acima ou Abaixo se, respectivamente, o ponto $(p_x - 1, p_y, p_z)$, $(p_x, p_y + 1, p_z)$, $(p_x, p_y - 1, p_z)$, $(p_x, p_y, p_z + 1)$ ou $(p_x, p_y, p_z - 1)$ for branco.

Cada iteração do algoritmo é dividida em seis sub-iterações, e em cada uma destas, são removidos pontos em apenas uma das direções Norte, Sul, Leste, Oeste, Acima ou Abaixo. Para garantir a remoção simultânea de pontos em uma mesma direção com preservação da topologia são estabelecidas condições adicionais.

Se a remoção de p não altera a conexidade dos pontos em um dado plano, dizemos que p é removível neste plano.

O ponto $p = (p_x, p_y, p_z)$ poderá ser removido em paralelo se:

i. p é um ponto simples;

ii. se p é um ponto nas direções Norte ou Sul então p é removível nos planos $y = p_y$ e $z = p_z$;

iii. se p é um ponto nas direções Leste ou Oeste então p é removível nos planos $x = p_x$ e $z = p_z$;

iv. se p é um ponto nas direções Acima ou Abaixo então p é removível nos planos $x = p_x$ e $y = p_y$.

4.2 Bertrand e Aktouf

Em [Bertrand-Aktouf (1994)], os autores propõem um algoritmo considerando imagens (26, 6) e (6, 26). O algoritmo baseia-se no cálculo de dois números topológicos T_n e $T_{\overline{n}}$, onde $(n, \overline{n}) \in \{(26,6),(6,26)\}$, para determinar pontos simples e pontos finais.

Sejam $\Im$ uma imagem binária tridimensional e p um ponto de $\Im$. Definimos $T_6(p,\Im) = \#_b(N_{18}(p) \cap \Im)$ e $T_{26}(p,\Im) = \#_b(N_{26}(p) \cap \Im)$. Os números topológicos levam a uma caracterização concisa de ponto simples: p é um ponto simples se, e somente se, $T_n(p,\Im) = 1$ e $T_{\overline{n}}(p,\overline{\Im}) = 1$.

A estratégia usada para remoção de pontos em paralelo, neste caso, é baseada em subconjuntos da imagem chamados *subcampos*. Em uma dada iteração, apenas pontos de um mesmo subcampo são considerados para remoção.

Seja $p = (p_x, p_y, p_y) \in \mathbb{Z}^3$. Seja $\rho_i = p_i \, mod \, 2$ para $i \in \{x, y, z\}$, isto é, $\rho_i = 0$ se p_i é par e $\rho_i = 1$ se p_i é ímpar. Sejam os subcampos $S_i = \{p \in \mathbb{Z}^3 : i = 4\rho_x + 2\rho_y + \rho_z\}$, para $i \in [0,7]$. Os 8 subcampos S_i constituem uma partição de $\mathbb{Z}^3$ (ver figura 3).

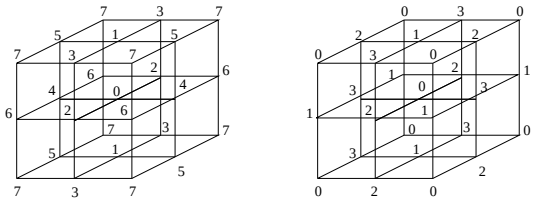

Figura 3: *Divisão em 8 subcampos (esquerda) e 4 subcampos (direita). O ponto central de cada uma das grades tem coordenadas (0, 0, 0).*

Como visto anteriormente, a caracterização de um ponto simples envolve apenas a informação dos pontos em uma vizinhança N_{26}. Podemos mostrar ainda que:

$$\forall p, q (p, q \in S_i) \Rightarrow p \notin N_{26}(q).$$

O que significa que, a remoção de um ponto não altera as características topológicas (simples ou não) de outros pontos no mesmo subcampo.

Também é apresentada, em [Bertrand-Aktouf (1994)], uma estratégia usando 4 subcampos. Neste caso, porém, a conexidade é restrita ao caso (26, 6) e não é garantida a preservação de topologia das componentes formadas por dois pontos 26-vizinhos (para detalhes, consultar [Bertand-Aktouf (1994)]). Para garantir tal preservação, outras condições devem ser atendidas.

5 Técnicas de otimização em afinamento 3D

No afinamento de imagens, a topologia e algumas características geométricas dependentes da aplicação (definição do eixo medial, por exemplo) podem ser verificadas várias vezes para um mesmo ponto durante o processo de afinamento, no entanto, apenas os pontos de borda serão removidos a cada etapa. Evitando-se tais verificações desnecessárias no processo iterativo de transformação da imagem, conseguiremos, à priori, uma redução no tempo de

execução do algoritmo. A figura 4 mostra o número de pontos pretos a cada iteração do processo de afinamento de uma imagem. Nos algoritmos descritos na seção anterior, todos os pontos pretos existentes são verificados a cada iteração. Como podemos ver, o número de pontos removidos é bem menor que o de pontos pretos. Nosso objetivo é evitar a verificação de pontos pretos que seguramente não serão removidos na iteração corrente. Algoritmos mais complexos para a verificação de ponto simples e das condições geométricas terão, assim, maior ganho de tempo considerando as técnicas abordadas a seguir.

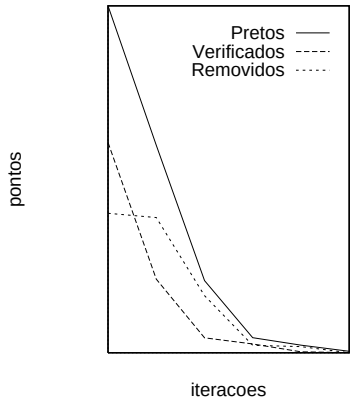

Figura 4: *Média de pontos pretos, pontos verificados e pontos removidos a cada iteração de um algoritmo de afinamento.*

Apresentamos, nas seções seguintes, duas técnicas para evitar redundâncias na verificação de pontos simples de uma imagem.

5.1 Etiquetagem de pontos viáveis

Seja $\Im = (\mathbb{Z}^3, \alpha, \beta, B)$ uma imagem sendo afinada por um algoritmo a. Dizemos que um ponto p é removível se p é simples e p atende às condições geométricas para remoção.

Lema 1 *Seja p um ponto não removível em uma dada iteração i da execução de a. Se q não é removível na iteração i, $\forall q \in N_{26}(p)$, então p não é removível na iteração $i+1$ da execução de a.*

Prova: A prova do lema acima é trivial, visto que a topologia e características geométricas do ponto p são verificadas a partir das cores (preto ou branco) de cada ponto em $N_{26}(p)$ (ver [Kong et al. (1989)], [Bertrand-Aktouf (1994)], [Tsao-Fu (1981)]). Como $N_{26}(p)$ não se altera após a iteração i, as características topológicas e geométricas locais de p permanecem inalteradas na iteração $i+1$, e, de acordo com essas características, p é não removível. □

Chamamos de conjunto de *pontos viáveis* na iteração i, o conjunto dos pontos que tiveram sua 26-vizinhança alterada na iteração i do algoritmo de afinamento a e denotaremos por V_i^a.

Seja R_i o conjunto de pontos removidos na iteração i. Assim, com base no lema 1, $V^a_{i+1} = \{p : p \in N_{26}(q), q \in R_i\}$, $\forall i > 1$. Na primeira iteração, todos os pontos serão considerados viáveis, ou seja, $V^a_1 = B$.

A técnica de etiquetagem de pontos viáveis consiste em etiquetar, como viáveis, os pontos em $N_{26}(p)$, quando da remoção de um ponto p. Na iteração seguinte, somente os pontos etiquetados serão verificados para remoção.

É importante notar que a complexidade assintótica do tempo de execução dos algoritmos sobre os quais a técnica foi usada não muda. Isto porque foi introduzido apenas um passo antes da checagem topológica que gasta tempo $O(1)$ (verificar se o ponto é efetivo) e um passo após a remoção (etiquetar no máximo 26 pontos efetivos) que também gasta tempo $O(1)$.

O algoritmo de afinamento otimizado pode ser descrito da seguinte forma:

Algoritmo

```
# Etiqueta todos os pontos como viáveis
para todo ponto preto (x)
  verificar[x] = Verdade;
faça
{
  algumPontoRemovido = Falso;
  para todo ponto preto (x)
    se( verificar[x] )
      se(Simples(x) e nãoPontoFinal(x))
      {
        # Etiqueta x para remoção
        remove[x] = Verdade;
        algumPontoRemovido = Verdade;
      }
  # Remove etiqueta para prox. iteração
  para todo ponto preto (x)
    verificar[x] = Falso;

  para todo ponto preto (x)
  {
    se( remove[x] )
    {
      delete(x);
      # Etiqueta pontos viáveis
      para todo y em N26(x)
        verificar[y] = Verdade;
    }
  }
} enquanto (algumPontoRemovido);
```

5.2 Dicionário de configurações

Como discutimos anteriormente, num algoritmo de afinamento típico, a cada iteração, todos os pontos pretos são verificados, com relação à sua topologia. Essa verificação pode ser feita várias vezes para um mesmo ponto durante diversas iterações e geralmente não é um processo simples, ver seções 4.1 e 4.2. Uma maneira rápida de verificarmos se um ponto p é simples, seria armazenar para cada configuração possível de $N_{26}(p)$ um bit contendo o valor 1 se p é simples ou o valor 0 se p não é simples. Como seriam necessários 2^{26} bits na representação das configurações, esta estratégia torna-se inviável. Teoricamente, este método representa uma generalização do afinamento a partir de transformações morfológicas ([Serra (1992)]), bastante empregadas no caso bidimensional, onde o conjunto de configurações possíveis é significativamente reduzido. No caso de imagens 3D, adotamos uma estratégia intermediária: armazenar configurações já calculadas em um dicionário. Este dicionário conterá o conjunto de configurções que aparecerão no decorrer do afinamento.

Seja um ponto p sendo verificado, utilizamos a palavra δ_p formada pelos bits correspondentes às cores dos pontos em $N_{26}(p)$, segundo alguma ordem fixa, como chave para consulta ao dicionário. Inicialmente, o dicionário pode estar vazio e a cada novo ponto p sendo verificado, é feita uma busca pela chave δ_p no dicionário. Se a chave não for encontrada, a topologia de p é verificada de acordo com um determinado método de verificação topológica. Daí, decide-se se p é removível ou não e essa informação é adicionada ao dicionário com a chave δ_p. Nas iterações seguintes, pontos com mesma configuração de suas vizinhanças evitarão a etapa complexa de teste da topologia, realizando, simplesmente, uma consulta ao dicionário.

Esta técnica apresenta um melhor resultado quando utilizada para afinamento de imagens regulares. Havendo um pequeno número de diferentes configurações para a vizinhança dos pontos, serão feitas mais consultas com sucesso e poucas inserções no dicionário. Ao contrário, quando existem muitas configurações diferentes, um grande número de consultas e inserções são feitas, comprometendo o resultado final.

É importante notar que, numa utilização prática, o dicionário deve ter número de entradas limitado por uma constante. Caso o dicionário cresça indeterminadamente, teremos maior tempo de busca e corremos o risco de nos aproximarmos das 2^{26} entradas possíveis. Quando usamos árvores AVL para implementação do dicionário, cada operação de inserção ou consulta gasta tempo $O(\log n)$, onde n é o número de ítens no dicionário. Como o tamanho do dicionário deve ser limitado por uma constante, consultas e insersões gastarão tempo constante. Portanto, esta técnica não altera a complexidade assintótica dos algoritmos onde é aplicada, visto que o número de pontos verificados em cada iteração também não é alterado.

O algoritmo otimizado, neste caso, pode ser descrito por:

Algoritmo

```
faça
{
  algumPontoRemovido = Falso;
  para todo ponto preto (x)
```

```
  {
    # busca configuração no dicionário
    posição = busca(dicionário, N26(x))
    se( posição==nil ) # não encontrou
      se( Simples(x) e nãoPontoFinal(x) )
      {
        delete(x);
        inserir(dicionário,N26(x),Verdade);
        algumPontoRemovido = Verdade;
      }
      senão
      {
        inserir(dicionário, N26(x), Falso);
      }
    senão               #encontrou
      se( dicionário[position] )
      {
        delete(x)
        algumPontoRemovido = Verdade;
      }
  }
} enquanto (algumPontoRemovido);
```

6 Testes

As técnicas mencionadas anteriormente foram associadas aos algoritmos descritos na seção 4. Além disto, a etiquetagem de pontos viáveis e o dicionário de configurações foram combinados, resultando nas otimizações apresentadas a seguir.

Para a implementação do dicionário de configurações, foi usado o dicionário implementado numa estrutura de árvores AVL da biblioteca LEDA ([Nürer-Uhrig]).

Os testes foram realizados sobre imagens binarizadas de um crânio obtidas pela técnica de ressonância magnética (figuras 5 e 6) numa estação SPARC 1000. A dimensão das imagens é de $256 \times 256 \times 60$ pontos. A figura 7 apresenta o resultado do esqueleto de superfície para o plano correspondente à figura 6. Os tempos de execução são mostrados nas tabelas 1 e 2.

7 Conclusão

Apresentamos duas técnicas de fácil implementação para otimização de algoritmos de afinamento tridimensional.

A etiquetagem de pontos viáveis funciona melhor para imagens que apresentem componentes densas, onde existe grande número de pontos interiores e pequeno número de pontos de borda. Neste caso, um número maior de verificações topológicas é evitado.

Técnica	Esqueleto	
	Superfície	Linha
Algo. original	279.37 (100%)	691.05 (100%)
Etiquetagem	251.36 (90%)	683.02 (99%)
Dicionário	56.93 (20%)	61.25 (9%)
Combinada	55.78 (20%)	44.62 (6%)

Tabela 1: *Tempos de execução medidos em minutos para o algoritmo de Tsao e Fu.*

Técnica	Esqueleto	
	Superfície	Linha
Algo. original	157.03 (100%)	278.70 (100%)
Etiquetagem	133.85 (85%)	169.17 (61%)
Dicionário	176.93 (113%)	250.17 (90%)
Combinada	121.27 (77%)	157.45 (56%)

Tabela 2: *Tempos de execução medidos em minutos para o algoritmo de Bertrand e Aktouf.*

A técnica de baseada no dicionário apresenta maior ganho de tempo para imagens que com semelhança nas vizinhanças dos pontos sendo verificados, gerando assim maior número de buscas com sucesso e dicionários de menor tamanho.

A técnica do dicionário de configurações apresentou melhores resultados que a etiquetagem de pontos viáveis, no caso do algoritmo de Tsao e Fu. Isso porque, neste algoritmo, um ponto pode ser verificado várias vezes, na mesma iteração. Assim, a busca no dicionário obtém sucesso um número maior de vezes e, como a verificação topológica neste algoritmo consume um tempo consideravelmente maior que uma busca ao dicionário, a diminuição do tempo de execução é realmente significativa (ver tabela 1). A etiquetagem, neste caso, pode levar muito tempo atribuindo etiquetas, desnecessariamente, produzindo resultados menos expressivos.

Para o algoritmo de Bertrand e Aktouf, a técnica de etiquetagem, sobretudo para o esqueleto de linha, apresentou melhor resultado que a do dicionário de configurações. Isso se deve ao fato deste algorimo verificar cada ponto uma única vez, por iteração, e desta verificação ser bem mais simples (ver seção 4.2). Enfim, como podemos ver nas tabelas 1 e 2, o uso das duas técnicas, conjuntamente consegue otimizar os aspectos vulneráveis de cada um dos algoritmos relacionados ao tempo necessário à verificação topológica local.

Como extensão direta deste trabalho, podemos citar a análise do impacto destas e de outras técnicas de otimização a outros algoritmos de afinamento em abordagens seqüenciais e paralelas.

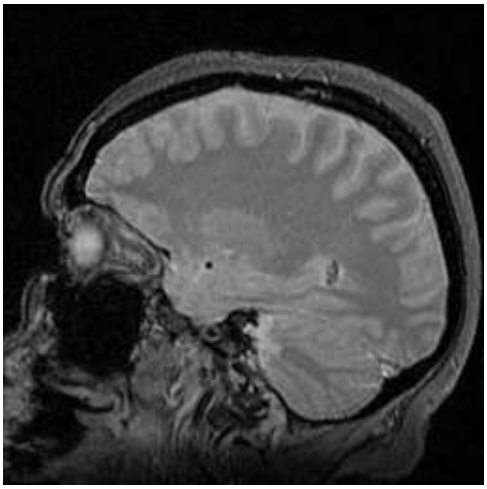

Figura 5: *Corte transversal de uma das imagens originais usadas nos testes.*

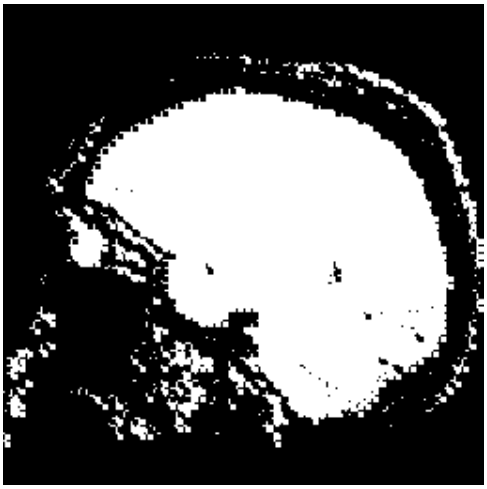

Figura 6: *Corte transversal da imagem binarizada.*

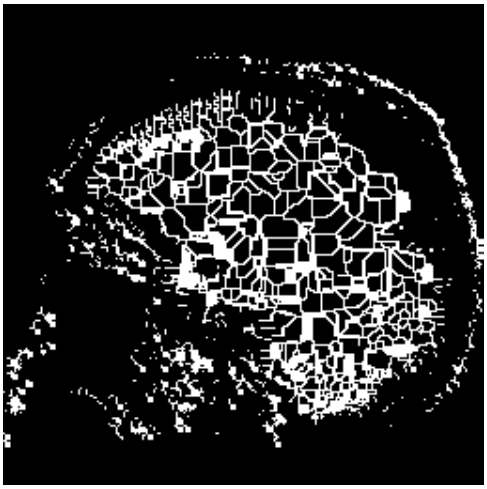

Figura 7: *Corte transversal da imagem afinada.*

Referências

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

[Kong et al. (1989)] 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.

[Kong-Rosenfeld (1992)] T. Y. Kong and A. Rosenfeld. Concepts of digital topology. *Topology and its applications*, 46:219–262, 1992.

[Lam et al. (1992)] 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.

[Lobregt et. al (1980)] 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.

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

[Nürer-Uhrig] S. Nüher and C. Uhrig. *The LEDA User Manual Version R 3.3.c.*

[Rosenfeld-Kak (1982)] A. Rosenfeld and A. C. Kak. *Digital Picture Procesing*, volume 2, pages 231–250. Academic Press, second edition, 1982.

[Saha et al. (1994)] 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.

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

[Serra (1992)] J. Serra. *Image Analysis Using Mathematical Morphology*, Academic Press, London 1992.

[Jackson] J. H. Jackson. *The Data Transport Computer: A 3-Dimensional Massively Parallel SIMD Computer*, Wavetracer Inc., Acton MA 01720.

Relatórios Técnicos – 1995

95-01 **Paradigmas de algoritmos na solução de problemas de busca multidimensional,** *Pedro J. de Rezende, Renato Fileto*

95-02 **Adaptive enumeration of implicit surfaces with affine arithmetic,** *Luiz Henrique de Figueiredo, Jorge Stolfi*

95-03 **W3 no Ensino de Graduação?,** *Hans Liesenberg*

95-04 **A greedy method for edge-colouring odd maximum degree doubly chordal graphs,** *Celina M. H. de Figueiredo, João Meidanis, Célia Picinin de Mello*

95-05 **Protocols for Maintaining Consistency of Replicated Data,** *Ricardo Anido, N. C. Mendonça*

95-06 **Guaranteeing Full Fault Coverage for UIO-Based Methods,** *Ricardo Anido and Ana Cavalli*

95-07 **Xchart-Based Complex Dialogue Development,** *Fábio Nogueira de Lucena, Hans K.E. Liesenberg*

95-08 **A Direct Manipulation User Interface for Querying Geographic Databases,** *Juliano Lopes de Oliveira, Claudia Bauzer Medeiros*

95-09 **Bases for the Matching Lattice of Matching Covered Graphs,** *Cláudio L. Lucchesi, Marcelo H. Carvalho*

95-10 **A Highly Reconfigurable Neighborhood Image Processor based on Functional Programming,** *Neucimar J. Leite, Marcelo A. de Barros*

95-11 **Processador de Vizinhança para Filtragem Morfológica,** *Ilka Marinho Barros, Roberto de Alencar Lotufo, Neucimar Jerônimo Leite*

95-12 **Modelos Computacionais para Processamento Digital de Imagens em Arquiteturas Paralelas,** *Neucimar Jerônimo Leite*

95-13 **Modelos de Computação Paralela e Projeto de Algoritmos,** *Ronaldo Parente de Menezes e João Carlos Setubal*

95-14 **Vertex Splitting and Tension-Free Layout,** *P. Eades, C. F. X. de Mendonça N.*

95-15 **NP-Hardness Results for Tension-Free Layout,** *C. F. X. de Mendonça N., P. Eades, C. L. Lucchesi, J. Meidanis*

95-16 **Agentes Replicantes e Algoritmos de Eco,** *Marcos J. C. Euzébio*

95-17 **Anais da II Oficina Nacional em Problemas Combinatórios: Teoria, Algoritmos e Aplicações,** *Editores: Marcus Vinicius S. Poggi de Aragão, Cid Carvalho de Souza [Not available.]*

95-18 **Asynchronous Teams: A Multi-Algorithm Approach for Solving Combinatorial Multiobjective Optimization Problems,** *Rosiane de Freitas Rodrigues, Pedro Sérgio de Souza*

95-19 **wxWindows: Uma Introdução,** *Carlos Neves Júnior, Tallys Hoover Yunes, Fábio Nogueira de Lucena, Hans Kurt E. Liesenberg*

95-20 **John von Neumann: Suas Contribuições à Computação,** *Tomasz Kowaltowski*

95-21 **A Linear Time Algorithm for Binary Phylogeny using PQ-Trees,** *J. Meidanis and E. G. Munuera*

95-22 **Text Structure Aiming at Machine Translation,** *Horacio Saggion and Ariadne Carvalho*

95-23 **Cálculo de la Estructura de un Texto en un Sistema de Procesamiento de Lenguaje Natural,** *Horacio Saggion and Ariadne Carvalho*

95-24 **ATIFS: Um Ambiente de Testes baseado em Inje,c ao de Falhas por Software,** *Eliane Martins*

95-25 **Multiware Plataform: Some Issues About the Middleware Layer,** *Edmundo Roberto Mauro Madeira*

95-26 **WorkFlow Systems: a few definitions and a few suggestions,** *Paulo Barthelmess and Jacques Wainer*

95-27 **Workflow Modeling,** *Paulo Barthelmess and Jacques Wainer*

Relatórios Técnicos – 1996

96-01 **Construção de Interfaces Homem-Computador: Uma Proposta Revisada de Disciplina de Graduação,** *F'abio Nogueira Lucena and Hans K.E. Liesenberg*

96Abs **DCC-IMECC-UNICAMP Technical Reports 1992–1996 Abstracts,** *C. L. Lucchesi and P. J. de Rezende and J.Stolfi*

96-02 **Automatic visualization of two-dimensional cellular complexes,** *Rober Marcone Rosi and Jorge Stolfi*

Instituto de Computação
Universidade Estadual de Campinas
13081-970 – Campinas – SP
BRASIL

reltec@dcc.unicamp.br