Esqueletos de Objetos Volumétricos

Adelailson Peixoto
peixoto@inf.puc-rio.br

Paulo Cézar Pinto Carvalho
pcezar@visgraf.impa.br

PUC-Rio.Inf.MCC 34/00 Agosto, 2000

Resumo

Esqueletos de objetos gráficos são poderosas ferramentas para se ter uma representação simples e eficiente do objeto. Objetos volumétricos, em particular, são representados por uma grande quantidade de informações. A extração de seus esqueletos permite reduzir estas informações, oferecendo uma alternativa para sua compactação. Neste trabalho é feito um levantamento das principais técnicas para calcular esqueletos de dados volumétricos. São abordados os principais problemas e discutidas as principais soluções adotadas.

Palavras Chave. esqueleto, eixo medial, thinning, dados volumétricos, funções distância, codificação de voxels, contornos.

Abstract

Skeletons of graphical objects are powerful tools that represent the object in a simple and efficient way. Volumetric objects are represented by a large quantity of information. Skeleton extraction, among other applications, can be used to calculate a compact representation of volumetric objects. This work presents a general framework of the main techniques used to compute skeletons of volumetric data. We will discuss the main problems and the different solution approaches.

Keywords. skeleton, medial axis, thinning, volumetric data, distance functions voxel-coding, contours.

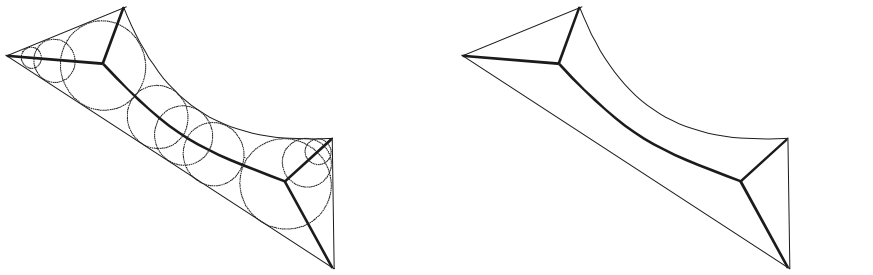
1-Introdução

Objetos do mundo real (incluindo os de interesse da Computação Gráfica, chamados de *objetos gráficos* [8]) são composto por um conjunto infinito de informações. Para modelar objetos no computador, estes devem ser representados por um conjunto finito de informações, uma vez que os recursos computacionais de memória, processamento e etc. possuem limitações físicas [7]. Estas informações finitas que representam o objeto gráfico são chamadas de *amostras*. O ideal é que as amostras que representam o objeto contenham o mínimo de informações, porém tais informações devem ser suficientes para que as características relevantes do objeto não sejam perdidas.

Uma maneira de se representar objetos gráficos é através de *esqueletos*, que possuem informações centrais relevantes da topologia e da geometria do objeto. No caso de objetos volumétricos, que possuem um grande número de informações, o uso de esqueletos tem-se mostrado cada vez mais importante, uma vez que esta grande quantidade de informações passa ser representada de forma bastante reduzida.

1.1-Eixo Medial e Esqueletos

O *eixo medial* de um objeto é definido como a união dos centros das circunferências de raio maximal, contidas no objeto. Tais circunferências tocam o interior do objeto em pelo menos dois pontos (figura 1).



a) Circunferências com raios maximais b) União dos centros das circunferências

Figura 1: Definição do eixo medial.

A definição acima é aplicada a objetos bidimensionais. No caso de objetos 3D, o eixo medial é definido como a união dos centros de esferas. Neste caso o resultado pode ser ou uma superfície (superfície medial) ou uma curva (eixo. medial). O eixo medial é também chamado de *esqueleto*, justamente por representar uma estrutura interna simplificada do objeto. Porém, nem todo esqueleto é o eixo medial. Muitas vezes, por ser difícil calcular um esqueleto de acordo com a definição acima (eixo medial), muitos métodos calculam esqueletos que procurem representar os objetos de forma parecida com esta definição, porém estes esqueletos diferem do eixo medial, que é considerado um dos esqueletos principais do objeto.

Esqueletos são operações que transformam objetos *n*-dimensionais em objetos com dimensões, no máximo, *n*-1 dimensões. Uma breve definição de esqueleto no universo contínuo pode ser encontrada em [3]. Neste trabalho, Blum postulou um paradigma onde uma frente em chamas avança ao queimar uma região coberta de grama. O eixo medial pode ser visto como o local onde a frente que está se propagando se encontra. Por exemplo, considerando que o objeto da figura 1 está coberto de grama, se for ateado fogo na borda (frente inicial) do objeto, a chama (frente evoluindo) vai invadindo o interior do objeto até se encontrar justamente no eixo medial.

Este paradigma da evolução de uma frente em chamas é utilizado para inspirar diversos métodos para o cálculo de esqueletos de objetos. Dentre este métodos destacam-se:

Métodos Usando Transformada de Distância. Estes métodos tentam extrair o esqueleto diretamente, testando a distância euclidiana mínima entre os pontos internos e a borda do objeto (figura 2). Os pontos eqüidistantes da borda compõem o esqueleto do objeto. Isto se deve ao fato de estes pontos estarem no centro de circunferências (esferas) inscritas ao objeto e, portanto, sua distância à borda é igual ao raio da circunferência.

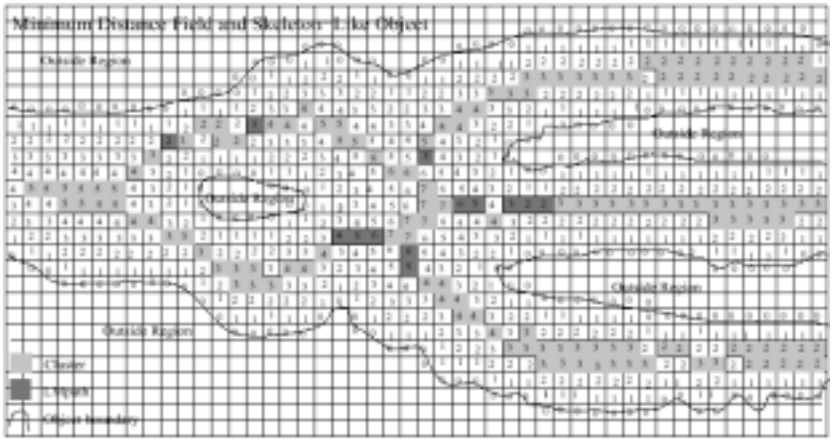


Figura 2: Esqueleto a partir de uma função distância discreta.

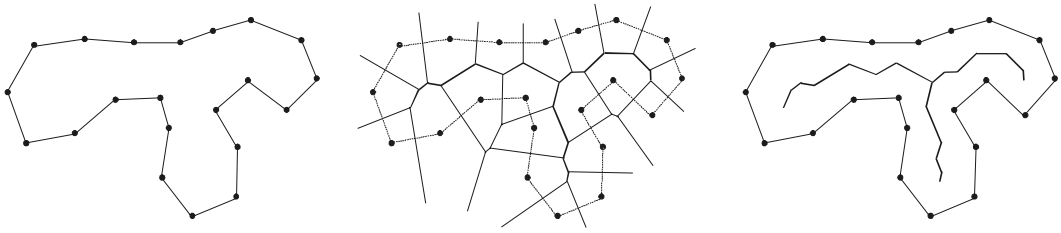
Métodos de Erosão ou *Thinning*. Estes métodos tentam, iterativamente, retirar camadas da borda do objeto até que reste apenas o esqueleto. A cada iteração são retirados pontos da borda (camadas), de modo que a topologia do objeto não seja alterada. Ao final deste processo restará apenas o esqueleto do objeto (figura 3).

Métodos de Voronoi. Estes métodos são baseados na construção do Diagrama de Voronoi dos pontos do objeto [17][24]. Após a construção do diagrama, são mantidas apenas as arestas internas ao objeto. Estas arestas definem o esqueleto. Estes métodos são mais utilizados para objetos poligonais ou poliedrais (representados por um conjuntos de vértices, arestas e faces) (figura 4).

a) b) c)

Figura 3: Thinning.

Para um melhor entendimento sobre a definição e classificação de esqueletos, o leitor pode consultar as referências [16], que traz várias discussões discretas sobre estes objetos, e [21], onde são discutidos vários conceitos matemáticos envolvidos com a definição de esqueletos.



a) Objeto Poligonal b) Diagrama de Voronoi c) Esqueleto de Voronoi

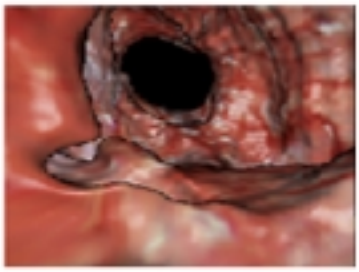
Figura 4: Método de Voronoi.

1.3- Aplicações

O cálculo de esqueletos de objetos gráficos tem diversas aplicações. Em particular, para objetos gráficos volumétricos, há uma série de aplicações onde o uso de esqueletos tem-se mostrado fundamental.

Na área médica, em particular, esqueletos podem ser usados como ferramentas para auxiliar na navegação virtual dentro de órgãos humanos. O esqueleto define um percurso que deve ser seguido, de forma que tais órgãos sejam explorados, sem a necessidade de técnicas invasivas [10][18]. A figura 5a mostra o esqueleto calculado sobre um intestino grosso e a figura 5b mostra a visão interna (endoscopia virtual) deste órgão, que é gerada a partir da uma câmera virtual navegando sobre o esqueleto.

a) Esqueleto



b) Endoscopia virtual

Figura 5: Endoscopia virtual a partir do esqueleto.

Outra aplicação importante é em compactação de dados volumétricos. Como o esqueleto mantém informações topológicas e geométricas do objeto, ele pode ser visto como uma forma de representação do volume, sendo que o objeto volumétrico, em geral, é formado por uma grande quantidade de informações, enquanto o esqueleto contém informações bem compactas. Nestes casos é essencial que o volume possa ser reconstruído a partir do esqueleto, ou seja, o esqueleto deve conter informações suficientes para que os dados volumétricos possam ser recuperados de forma exata ou aproximada (neste último caso, o objeto é reconstruído com um erro).

O cálculo de esqueletos pode ser aplicado ainda nas áreas de Visão Computacional e Reconhecimento de Padrões [4][22]. Em Modelagem de Sólidos, esqueletos podem ser usados como ferramentas para representações intermediárias de um grande número de operações geométricas [26][27][29].

2- Esqueletos de Objetos Volumétricos

2.1-Objetos Volumétricos

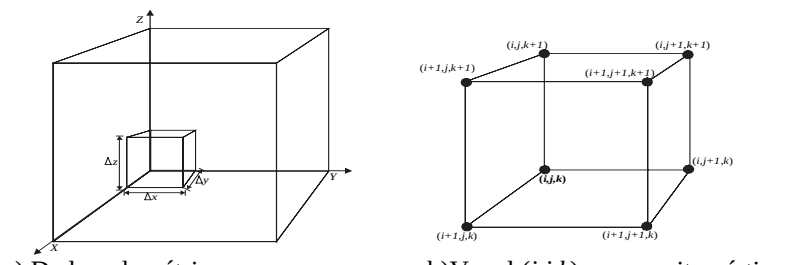
Um objeto volumétrico pode ser matematicamente definido como um par (U,f), onde $U \subset \mathbf{R}^3$ é o *suporte geométrico* e $f: U \subset \mathbf{R}^3 \rightarrow \mathbf{R}^m$ é a *função de atributos* do objeto [8]. No geral o suporte geométrico é definido como um bloco $U = [0,X]\times[0,Y]\times[0,Z]$. A dimensão do conjunto de atributos ($\mathbf{R}^m$) depende da aplicação. Por exemplo, se os atributos considerados são densidade e opacidade (dois atributos, logo m=2), então a cada ponto $(x,y,z) \subset U$ são associados um valor de densidade e um valor de opacidade,

$$f: \; U \subset \mathbf{R}^3 \rightarrow \mathbf{R}^2$$
$$f(x,y,z) = (f_d(x,y,z), f_o(x,y,z)) \, ,$$

onde $f_d(x, y, z) \in \mathbf{R}$ é a função de densidade e $f_o(x, y, z) \in \mathbf{R}$ é a função de opacidade do elemento (x, y, z). A função de densidade é um dos principais atributos destes objetos. É esta função quem decide quais pontos (x, y, z) fazem parte do objeto volumétrico e quais pontos fazem parte do fundo ou *background* do volume.

No caso de objetos volumétricos binários, a função de densidade assume apenas os valores 0 ou 1. Assim $f_d(x, y, z)=1$ se o ponto (x, y, z) pertence ao objeto e $f_d(x, y, z)=0$ se o ponto (x, y, z) não faz parte do objeto definido dentro do volume (neste caso, o ponto (x, y, z) faz parte do *background* do volume).

No universo discreto, cada um dos três intervalos $[0,X]$, $[0,Y]$ e $[0,Z]$ é uniformemente amostrado em NX, NY e NZ amostras, respectivamente. Assim, o espaçamento entre cada amostra, em cada direção, será $\Delta x = X/NX$, $\Delta y = Y/NY$ e Δz Z/NZ, respectivamente. Cada elemento ou *voxel* do objeto, referenciado como voxel (i, j, k), é tratado como um pequeno bloco retangular, com dimensões Δx, Δy e Δz e coordenadas $(i\Delta x, j\Delta y , k\Delta z) \in U$ (i = 0, 1 ,2,…, NX-1; j = 0 ,1, 2,…, NY-1; k = 0, 1, 2,…, NZ-1). Este tipo de representação de objetos é conhecido como *representação matricial*. A figura 6a mostra um volume com um voxel. A figura 6b mostra o voxel (i,j,k) ampliado e seus 8 vértices. Para uma melhor compreensão de objetos volumétricos, o leitor pode consultar as referências [11] [25].



a) Dado volumétrico b)Voxel (i,j,k) e seus oito vértices

Figura 6: Definição discreta de um volume.

2.2-Esqueletos de Volumes

Assim como os objetos volumétricos, os esqueletos extraídos destes objetos são compostos por um conjunto de voxels. Essencialmente, o esqueleto retém as características relevantes dos objetos volumétricos binários. Portanto a extração do esqueleto deve ser aplicada apenas aos voxels do volume que fazem parte da composição do objeto (onde a função de densidade f_d vale 1) e não aos voxels que compõem o background (onde a função de densidade f_d vale 0).

Quando o objeto não é binário, informações sobre os voxels que compõem o objeto são obtidas a partir de uma densidade de busca d e da função de densidade f_d. Assim, dada a densidade de busca d, o volume é percorrido e, dependendo do valor $f_d(i,j,k)$ de cada voxel (i,j,k), este pode fazer parte do objeto ou fazer parte do background.

Como o conjunto de voxels que compõe o esqueleto tem uma estrutura que pode ser descrita por um conjunto de cadeias ou sequências, algumas áreas de pesquisa, como reconhecimento de padrões, comumente tratam o esqueleto de objetos volumétricos como sendo uma operação que converte objetos rasterizados (matriciais) para objetos vetoriais.

É importante observar que a implementação de métodos para calcular esqueletos no universo discreto tem sido um desafio. Isto ocorre por causa da dificuldade de se tratar propriedades importantes, como conectividade ou medidas euclidianas, no universo discreto. No universo contínuo esqueletos são invariantes por orientação dos objetos e por escalas. No universo discreto estas invariâncias não ocorrem. Portanto uma série de problemas devem ser tratados exclusivamente para objetos discretos .

Os principais problemas envolvidos no cálculo destes esqueletos estão voltados à manutenção das propriedades intrínsecas que todo esqueleto deve ter. Dentre estas propriedades, destacam-se: conectividade dos voxels que compõem o esqueleto, posição centralizada em relação ao objeto, mesma topologia do objeto. Além destas propriedades, é importante também considerar a sensibilidade a ruídos, pois isto interfere no resultado final, e a viabilidade do cálculo computacional do esqueleto.

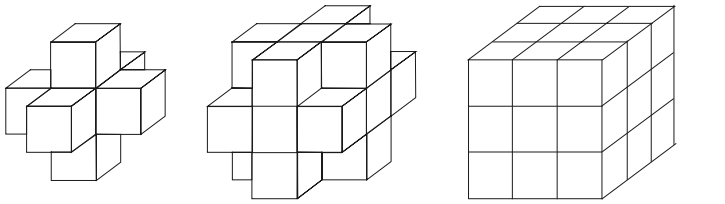
2.3- Propriedades

Conectividade. Qualquer método para calcular esqueletos de dados volumétricos deve garantir esqueletos conexos ou contínuos. Como estes objetos são compostos por uniões de seqüências de voxels, sua conectividade deve ser definida de acordo com as três possibilidades: *6-connected, 18-connected* e *26-connected*.

Uma seqüência de voxels é chamada *6-connected* quando cada dois voxels p e q só são considerados vizinhos ou adjacentes se compartilharem uma face. Assim, cada voxel contém, no máximo, seis vértices adjacentes (figura 7a).

Uma seqüência de voxels é chamada *18-connected* quando cada dois vértices *p* e *q* só são considerados vizinhos ou adjacentes se compartilharem uma face ou uma aresta. Cada voxel contém, no máximo, 18 vizinhos, sendo 6 pelas faces e 12 vizinhos pelas arestas (figura 7b).

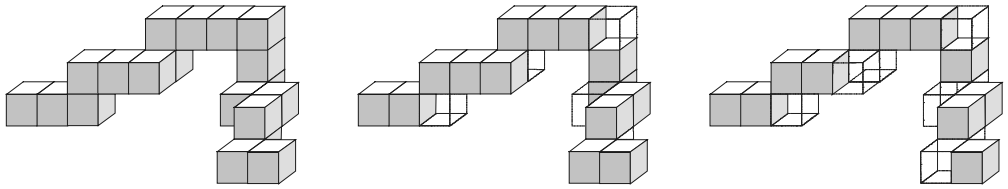
Uma seqüência de voxels é chamada *26-connected* quando cada dois vértices *p* e *q* só são considerados vizinhos ou adjacentes se compartilharem uma face, uma aresta ou um vértice. Cada voxel contém, no máximo, 26 vizinhos, sendo 6 pelas faces, 12 pelas arestas e 8 vizinhos pelos vértices (figura 7c).



a) 6 vizinhos b) 18 vizinhos c) 26 vizinhos

Figura 7: Adjacência entre voxels.

Esqueletos 6-connected tendem a ser mais grossos, uma vez que os voxels vizinhos estão conectados uns aos outros através de suas faces, o que necessita de um número maior de voxels. Em compensação, esqueletos 26-connected tendem a ser mais finos, pois muitos de seus voxels vizinhos estão ligados apenas por um vértice, o que reduz o número de voxels da seqüência. A figura 9 mostra uma mesma sequência de voxels considerando conectividade 6-connected (a), 18-connected (b) e 26-connected (c).



a) Seqüência 6-connected b) Seqüência 18-connected c) Seqüência 26-connected

Figura 8: Sequências de voxels conectados.

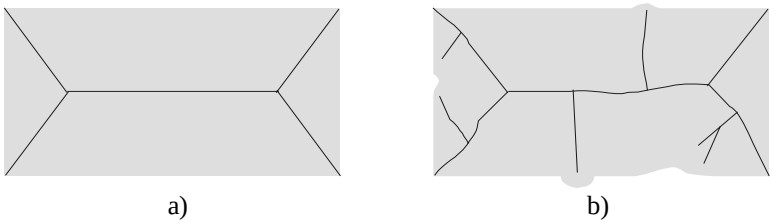
A maioria dos métodos para cálculo de esqueletos dedicam atenção considerável à demonstração de que eles realmente mantêm a conectividade destes objetos. Como será visto na seção 3.1, a definição das relações de adjacências entre voxels está relacionada com a escolha de uma métrica regular, utilizada para o cálculo de distâncias entre os voxels.

Posição Centralizada. O cálculo de esqueletos deve sempre seguir a idéia das circunferências de raio maximal, contidas no objeto. Esta definição garante que o esqueleto está geometricamente centralizado em relação ao objeto. Assim as seqüências conexas dos voxels que compõem o esqueleto do volume devem representar a união dos

centros de esferas inscritas ao objeto volumétrico. É claro que, como o objeto é discreto, esta centralização, em geral, é aproximada.

Preservação da Topologia. É de fundamental importância que o esqueleto mantenha a mesma topologia do objeto que ele representa. Por exemplo, se o objeto tem o formato de um 'b', seu esqueleto não pode ter a forma de um 'o' ou de um 'l'. Caso o esqueleto não consiga representar as informações topológicas adequadamente, fica inviável reconstruí-lo, mesmo que seja com uma margem de erro. Uma das causas que pode levar a não preservação da topologia é o cálculo incorreto da conectividade do esqueleto.É essencial que todas as informações topológicas do objeto, como buracos, bifurcações e etc. sejam, de alguma forma, passadas ao esqueleto.

Sensibilidade a Ruídos. Pequenas perturbações aplicadas à borda de um objeto podem interferir, de forma significante, no resultado final do seu esqueleto. Em geral, pequenas perturbações são consideradas ruído (informação irrelevante) e não é desejável que elas influenciem na extração do esqueleto. A figura 9 mostra como pequenas alterações na borda de um retângulo resultam em esqueletos completamente diferentes.



a) b)

Figura 9: Esqueleto sensível à alteração da borda.

Na prática é comum que os objetos contenham algum tipo de ruído. Os métodos utilizados para extrair os esqueletos devem selecionar as informações importantes para compor o resultado final. Assim as bifurcações indesejadas, que resultam em esqueletos mais complexos (como o da figura 9b), devem ser eliminadas, de alguma forma.

Cálculo Computacional. Para conjuntos de dados muito grandes e complicados, o tempo de processamento de esqueletos pode se tornar um fator crítico. Como já foi dito antes, objetos volumétricos costumam conter uma grande quantidade de informações. Para calcular o esqueleto, o volume de voxels deve ser percorrido e são aplicadas várias operações, em cada voxel, até que a seqüência final, que irá compor o esqueleto, seja obtida. Para se ter uma idéia, é muito comum processar volumes de 512 x 512 x 512 (134.217.728) voxels. Dependendo da forma como este volume é varrido e das operações aplicadas sobre seus elementos, pode se tornar praticamente inviável obter o esqueleto deste objeto, em um intervalo de tempo considerado razoável. Portanto, o cálculo computacional é um fator importante a ser considerado na extração de esqueletos.

Como mostrado na seção 1.1, há vários métodos utilizados para cálculo de esqueletos. Para dados volumétricos, os mais usados são o método com transformadas de distância e o de *thinning* ou erosão. A seguir estes dois métodos são detalhadamente estudados.

3-Métodos Usando Transformada de Distância

Estes métodos calculam o esqueleto diretamente, através de uma *transformada de distância* aplicada aos pontos do objeto volumétrico. A transformada escolhida deve fornecer valores que representam distância mínima, ou seja, a distância entre um ponto interno e a borda do objeto deve ser a menor possível. Após aplicar esta transformada a todos os pontos do objeto, o esqueleto será composto pelos pontos para os quais a distância é um máximo local. Estes pontos são eqüidistantes da borda e, conseqüentemente, estão situados nos centros das esferas inscritas ao objeto, conforme a definição de esqueletos.

3.1-A Transformada Distância

A transformada distância T_O, aplicada aos pontos de um objeto O, pode ser definida da seguinte maneira: para cada ponto $p \in O$,

$$T_O(p) = \min_{p_i \in \bar{O}} dist(p, p_i)$$

onde $\{p_i\}$ é o conjunto de pontos complementar de O, ou seja cada $p_i \in \bar{O}$ ($\bar{O}$ é também chamado *background*), e *dist* representa uma função distância ou métrica utilizada. Assim, para cada ponto p do objeto, a transformada calcula a distância de p ao ponto p_i (externo ao objeto) que está mais próximo de p. Na prática p representa um ponto interno ao objeto O e p_i é um ponto da borda do objeto (figura 10). É claro que com esta definição, para os pontos p situados na borda do objeto, tem-se T_O=0.

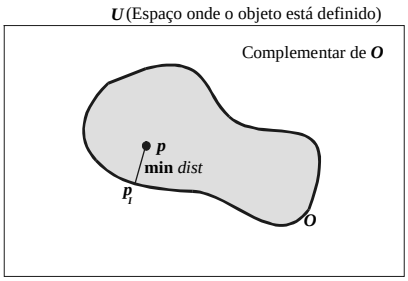


Figura 10: Distância mínima entre dois pontos.

O resultado da tranformada T_O depende da métrica que define a função *dist*. Algumas das métricas consideradas são: a euclidiana, a *city block* e a *chessblock*.

Métrica Euclidiana. Se a métrica utilizada representa a distância euclidiana, então

$$dist(p,q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + ... + (p_n - q_n)^2},$$

onde $p = (p_1, p_2, \ldots, p_n)$ e $q = (q_1, q_2, \ldots, q_n)$ são pontos do espaço *n*-dimensional.

Esta definição para a transformada T_O pode ser aplicada tanto a objetos do espaço contínuo quanto a objetos do espaço discreto (como é o caso dos objetos volumétricos). O problema de utilizar a transformada distância com a métrica euclidiana para objetos discretizados é que nem é algoritmicamente fácil implementá-la, nem seu cálculo computacional é eficiente, já que envolve o cálculo de quadrados e raízes.

Por esta razão, muitos métodos de cálculo de esqueletos substituem a métrica euclidiana por métrica regulares, como a *city block* e a *chessboard*, que possuem um cálculo computacional mais eficiente e são de fácil implementação [15]. A seguir estas métricas são definidas.

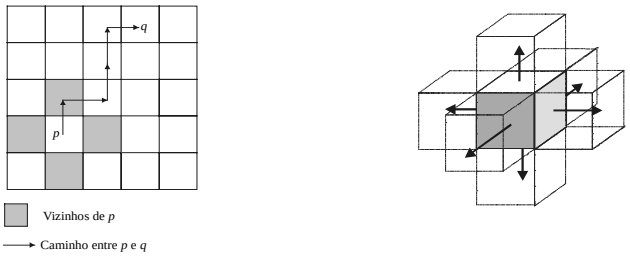
Métrica City Block. Nesta métrica, também conhecida como *métrica de Manhattan*, a função *dist* é definida como

$$dist(p,q) = |p_1 - q_1| + |p_2 - q_2| + \ldots + |p_n - q_n|.$$

Quando aplicada a objetos do espaço discreto, esta métrica assume que, para ir de um ponto *p* a um ponto *q*, só é possível andar nas direções dos eixos principais, ou seja, não é possível andar em diagonal. Os dois pontos *p e q* só são adjacentes se *dist*(*p*,*q*) =1.

No caso 2D discreto, esta métrica determina que, para ir de um ponto (chamado pixel) a outro, só há 4 possibilidades (2 movimentos horizontais e 2 movimentos verticais) que são justamente nas direções dos 4 pixels vizinhos por aresta. Isto se deve ao fato de não ser permitido andar de um ponto para outro na direção diagonal. Por isto diz-se que os pontos são *4-connected*. (figura 11a). No caso 2D a city block também é referenciada como métrica "1-2", no sentido de que, dado um ponto *p*, seus vizinhos por aresta têm distância 1 (pixels adjacentes) e seus vizinhos por vértices têm distância 2.

No caso 3D discreto, para ir de um ponto (neste caso chamado de voxel) a outro, só há 6 possibilidades, que são os 6 vizinhos por face. Diz que os voxels são *6-connected* (figuras 7a e 11b). No caso 3D esta métrica é também conhecida como métrica "1-2-3", no sentido de que os vizinhos por face de um ponto *p* têm distância 1 (voxels adjacentes), os vizinhos por aresta têm distância 2 e os vizinhos por vértices têm distância 3.



a) 2D: 4 pontos adjacentes (4-connected) b) 3D: 6 pontos adjacentes (6-connected)

Figura 11: Vizinhos definidos pela métrica city block.

Métrica Chessboard. Nesta métrica a função *dist* é definida como

$$dist(p,q) = \max(|p_1 - q_1|, |p_2 - q_2|, ..., |p_n - q_n|).$$

Quando aplicada a objetos do espaço discreto, esta métrica assume que, para ir de um ponto *p* a um ponto *q*, é permitido se deslocar em todas as direções. Os dois pontos *p e q* só são adjacentes se *dist*(*p*,*q*) =1.

A principal motivação para o nome dessa métrica vem das aplicações discretas 2D, onde, saindo de um ponto (pixel) *p* é possível fazer os movimentos que um rei faz em um tabuleiro de xadrez. Portanto é permitido andar tanto nas direções dos eixos principais (horizontal e vertical) quanto nas direções diagonais (figura 12a). Devido a isto, um pixel *p* possui 8 pixels adjacentes: os 4 vizinhos por aresta (direções horizontal e vertical) e os 4 vizinhos por vértices (direções diagonal). Diz, então que os pixels são *8-connected*. No caso 2D a métrica chessblock é também chamada de métrica "1-1", no sentido de que, dado um ponto *p*, tanto seus vizinhos por aresta quanto seus vizinhos por vértice possuem distância 1.

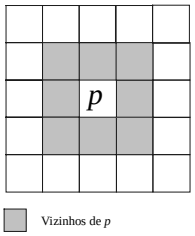


Figura 12: Vizinhos definidos pela métrica chessblock.

No caso 3D discreto, partindo de um ponto (voxel) *p* é permitido seguir em qualquer direção, ou seja, na direção dos 6 vizinhos por faces (figuras 7a e 11b), na direção dos 12 vizinhos por arestas e na direção dos 8 vizinhos por vértices , totalizando 26 direções possíveis. Portanto, qualquer vizinho (por face, aresta ou vértice) é adjacente a *p*. Diz, então que, utilizando a métrica chessblock, os voxel são 26-connected (figura 7c). No caso 3D esta métrica é também chamada de métrica "1-1-1", pois os vizinhos de um ponto *p* possuem distância *1* em relação aos vizinhos por faces, por arestas e por vértices.

Métricas "n_f - n_a - n_v". A métrica "1-2-3" (city block 3D discreta) e a métrica "1,1,1" (chessblock 3D discreta) podem ser vistas como casos particulares da métrica discreta "n_f-n_a-n_v", onde, dado um voxel *p*, n_f representa a distância entre *p* e seus vizinhos por face, n_a representa a distância entre *p* e seus vizinhos por aresta e n_v é a distância entre *p* e seus vizinhos por vértices.

Dois exemplos de métricas "n_f-n_a-n_v" bastante utilizadas são a métrica "2-3-4" [6] e a métrica "3-4-5" [5].

3.2-Codificação de Voxels

Definida a métrica a ser utilizada, ela deve ser aplicada aos voxels do volume, de modo que a transformada de distância discreta seja determinada. Este processo é feito através de uma *codificação dos voxels* do volume.

Codificação dos voxels de um objeto volumétrico O é uma operação, definida a partir de uma métrica, que se propaga recursivamente voxel a voxel do volume. Esta operação começa a ser aplicada em um conjunto inicial de voxels V_0 ($V_0 \subset O$) e se espalha pelos demais voxels de O, até que uma condição de parada seja atingida.

A partir da codificação de voxels de um volume é possível obter ou detectar a conectividade do objeto e extrair informações sobre sua geometria. Estas informações geométricas e de conectividade podem ser utilizadas para extração de linhas centralizadas do objeto e, conseqüentemente, para o cálculo de esqueletos [30]. Outras aplicações da codificação de voxels estão ligadas à geração de contornos do objeto e à reconstrução de superfícies a partir destes contornos.

A codificação se dá por um processo de propagação, semelhante à evolução de uma frente em chamas. Uma vez que um voxel é visitado, um valor (código do voxel) é associado a ele, indicando sua distância ao conjunto inicial V_0 (frente inicial).

A operação de propagação de voxels usando a métrica "n_f-n_a-n_v" pode ser descrita como se segue: primeiro, todos os voxels do objeto O são codificados com o valor infinito, em seguida todos os voxels do conjunto V_0 são codificados com o valor zero (início da propagação). A todos os vizinhos dos voxels de V_0 por faces é associado o valor n_f, a todos os vizinhos por arestas é associado o valor n_a e a todos os vizinhos por vértices é associado o valor n_v. Durante a propagação, todos os voxels com um determinado código n são processados ao mesmo tempo. Assim, se voxels com valor n são processados, aos seus vizinhos por face, por aresta e por vértice são associados os valores $n+n_f$, $n+n_a$ e $n+n_v$, respectivamente, caso estes valores sejam menores do que os valores correntes dos voxels vizinhos. Este processo de codificação continua até que sejam atingidas as condições de parada.

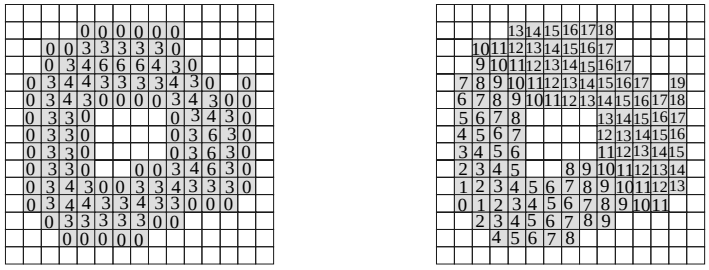
A escolha do conjunto de voxels iniciais V_0 depende das características que se deseja extrair da codificação. A seguir são escolhidos dois conjuntos distintos para V_0, que resultam em campos escalares com características diferentes: *Boundary Field* e *Single Field*.

Boundary Field. O conjunto V_0 é composto pelos voxels que formam a borda do objeto. A codificação gerada nos voxels forma um campo escalar distância tradicional, chamado *Bounday Field*. O código gerado para cada voxel interno indica sua distância à borda do objeto e será chamado *boundary-code*. Os voxels centralizados, em relação ao objeto, possuem código máximo local. Estas informações são fundamentais para a extração do esqueleto. A figura 2 mostra o campo distância gerado sobre um objeto 2D, onde o conjunto V_0 é formado pelos pixels da borda do objeto. Os pixels centralizados (com

maior código gerado) estão em destaque. A figura 13a mostra outro exemplo de boundary field, com a métrica “3-4”.

É importante observar que a métrica selecionada, ou seja a escolha dos valores de n_f, n_a e n_v, influencia na precisão final do esqueleto. Alguns trabalhos que utilizam este campo são mostrados em [31] e [30].

Single Field. O conjunto V_0 é formado por um único voxel inicial v_0. A codificação gerada nos voxels forma um campo escalar distância chamado *Single Field*. O código gerado para cada voxel interno indica sua distância ao voxel inicial v_0 e será chamado *single-code*. A figura 13b mostra um exemplo de single field, com a métrica “1-2”. Se o objeto é formado por partes desconexas, é necessária a escolha de um ponto inicial v_0 para cada parte desconexa.



a) Boudary Field 2D b) Single Field 2D

Figura 13: Dois campos escalares distância.

O campo single field pode ser utilizado, por exemplo, para extração do menor caminho entre dois voxels (ou pixels) v_1 e v_2, como mostrado em [30]. Este problema envolve duas etapas: a primeira etapa é a geração do single field, utilizando v_2 como ponto inicial (ou seja $v_2 = v_0$). A segunda etapa extrai o caminho mais curto: v_1 é escolhido como o primeiro voxel do caminho e o próximo voxel escolhido é o vizinho de v_1 que contém o menor código. Recursivamente, o próximo voxel é escolhido de maneira semelhante, como sendo o vizinho, com menor código, do voxel escolhido anteriormente.

3.3-Exemplo de Método que Utiliza Transformada de Distância

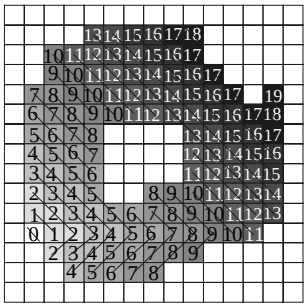
Para uma melhor compreensão da extração de esqueletos de objetos volumétricos a partir da transformada de distância, esta seção discute um método que foi desenvolvido por Zhou et al. [31].

Este método calcula o esqueleto a partir dos campos escalares single field, usando a métrica “1-2-3”, e boundary field, usando a métrica “3-4-5”. Os esqueletos gerados são formados por sequências de voxels que representam curvas e não superfícies do objeto.

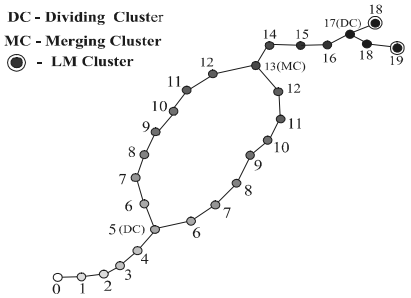
O campo single field fornece informações sobre a conectividade do objeto, pois define o objeto como sendo composto por um conjunto de *clusters*. Um cluster é definido como um conjunto de voxels conexos que possuem o mesmo single-code (código calculado a

partir do single field) *n*. Com esta definição, o cluster passa a ser considerado a unidade do objeto, e não o voxel. O objeto pode ser tratado como um grafo direcionado, onde cada nó do grafo é tratado como um cluster (figura 14b). Um cluster com single-code *n* adjacente a pelo menos dois outros clusters com single-code *n*-1 é chamado *merging cluster*. Um cluster com single-code *n* adjacente a pelo menos dois outros clusters com single-code *n*+1 é chamado *dividing cluster*. Dividing clusters indicam o início de uma cavidade ou buraco no objeto, enquanto merging clusters indicam o final de uma cavidade ou buraco. Outra definição importante é a de *local maximum cluster* ou *LMcluster* , que significa um cluster que possui código maior do que todos os seus clusters adjacentes.

Sem perda de generalidade, será usado um modelo 2D para exemplificar o método aqui discutido e as definições acima. A figura 14a mostra o conjunto de clusters 2D definidos a partir do single field da figura 13b. Na figura, o cluster com single-code 0 é formado por apenas um pixel e o cluster com single-code 1 é formado por dois pixels. Há dois clusters com single-code *6*, dois com single-code 7, 8, 9, 10, 11, 12 e 18. Os dois clusters de código 18 também contêm apenas um voxel. A figura 14b mostra o grafo representando a conectividade do objeto, onde cada nó representa um cluster e o número representa o código de cada cluster. Os clusters com código 5 e 17 são dividing clusters (início de um buraco) e o com código 13 é um merging cluster (final de um buraco). Um dos clusters com código 18 e o com código 19 são LMclusters.



a) Conjunto de cluster



b) Grafo: cada nó representa um cluster

Figura 14: Definição do grafo a partir do single field.

O grafo da figura 14b, calculado a partir do single field, deixa claro como deve ser a conectividade do esqueleto.

A partir das informações dos campos distância single field e boundary field devem ser extraídos vários caminhos ou sequências de voxels conectados que formarão o esqueleto do objeto volumétrico. Cada um destes caminhos será chamado de *centerline*.

Extração de uma Centerline. A extração de uma centerline, iniciada em um cluster *c*, é feita em duas etapas:

Etapa 1: Primeiro procura-se o *voxel médio* do cluster *c*. O voxel médio de *c* é definido como o voxel de *c* que possui o maior boundary-code (código lido a partir do boundary

field). Se há mais de um voxel médio, escolhe-se o que estiver mais próximo do centro geométrico de *c*. O voxel médio do cluster *c* é usado como voxel inicial da centerline e o voxel inicial do single field (v_0) é usado como voxel final da centerline. Dado um voxel p_i da centerline, seu próximo voxel p_{i+1} é calculado como sendo o vizinho por face (e não qualquer vizinho) de p_i que possui o menor single-code. Se houver mais de um vizinho por face satisfazendo a condição, então a ordem de escolha dos vizinhos por face é: frente, atrás, direita, esquerda, topo, abaixo. Após a etapa 1, a centerline possui as seguintes propriedades: sempre há o próximo voxel e seu single-code é igual ao do antecessor mais 1 (já que o single field foi gerado com a métrica "1-2-3"). Cada voxel da centerline corresponde a um cluster diferente (é importante esta informação), ou seja, a centerline passa através de uma sequencia de clusters adjacentes. A centerline vai sempre acabar no voxel v_0 , caso nenhuma condição de parada tenha ocorrido antes. A centerline não representa o menor caminho entre os voxels inicial (voxel médio de *c*) e final (v_0), pois o próximo voxel escolhido não era o vizinho que possuia o menor single-code, mas o vizinho por face que possuia o menor single-code. E finalmente a centerline ainda não está centralizada em relação ao objeto.

Etapa 2. Esta etapa visa centralizar a centerline. Exceto o primeiro voxel (voxel médio de *c*), todos os outros precisam ser centralizados. Assim, cada voxel p_{i+1} é substituído pelo voxel médio do seu cluster correspondente Se o cluster tiver mais de um voxel médio, escolhe-se o que estiver mais próximo a p_i (já centralizado). Após todas as substituições, tem-se de fato, uma centerline. É importante observar que, após as substituições, voxels adjacentes podem não estar conectados.

Extração de Múltiplas Centerlines. O algoritmo para extrair uma única centerline é utilizado diversas vezes (conforme explicado abaixo) até que todas as centerlines que compõem o esqueleto sejam calculadas. Na extração das mútiplas centerlines deve-se considerar os seguintes fatores:

a) As centerlines são tão longas quanto possível.
b) Duas centerlines quaisquer não podem se interceptar, exceto em suas extremidades.
c) Todas as centerlines extraídas devem compor o esqueleto.

O item (c) alerta quanto às informações topológicas que devem ser passadas ao esqueleto corretamente. Como os clusters representam a conectividade do objeto (e consequentemente a topologia, como buracos e bifurcações) é essencial que as centerlines que compõem o esqueleto passem por todos os clusters definidos no single field, ou seja, contenham pelo menos um voxel de cada cluster.

A primeira centerline é extraída a partir do maior LMcluster, a segunda, a partir do segundo maior LMcluster, e assim por diante, até que todos os LMclusters tenham extraído uma nova centerline

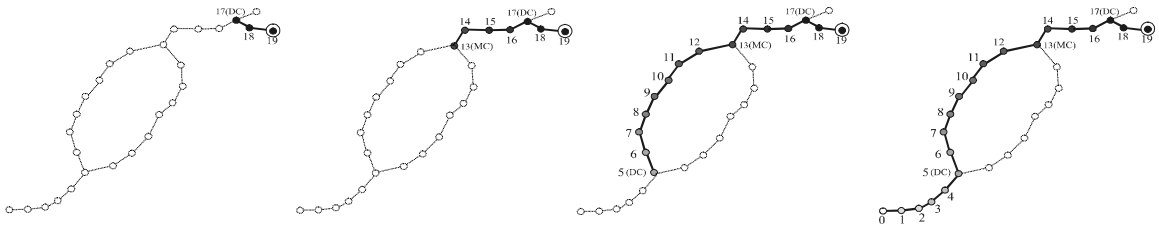
Conforme o algoritmo de extração de uma única centerline, cada próximo voxel que irá compor a centerline pertence a um cluster adjacente, de menor single-code. Se o cluster é um merging cluster, então há pelo menos dois clusters adjacentes com single-code menor,

o que significa que a centerline só vai poder seguir por um dos clusters. Por exemplo, na figura 14b, o cluster de código 13 é um merging cluster, e há dois clusters adjacentes com código 12. Como a centerline só vai poder seguir por um cluster, o outro deve ser processado posteriormente (lembre-se do item (c)). Para tal, os merging clusters também são pontos de partida para uma nova centerline. Ou seja sempre que ocorrer um merging cluster com *m* clusters adjacentes, os *m*-1 clusters não processados naquele momento devem, futuramente, dar início à extração de *m*-1 novas centerlines (estas novas extrações só devem ser inicadas depois que as centerlines iniciadas nos LMclusters forem extraídas).

A primeira centerline extraída (que começou no maior LMcluster) terá como voxel final o voxel v_0 (início do single field), ou seja, o cluster de single-code 0. Durante o cálculo do próximo voxel, sempre que um dividing cluster for processado, deve-se guardar esta informação, pois isto indica que este diving cluster será o cluster final de outra centerline extraída posteriormente. Ou seja, quando um dividing cluster é processado, na busca do próximo cluster, ele é sempre marcado, de modo que passará a ser o cluster final de uma outra centerline.

Como isto, ficam bem definidos os papéis dos merging clusters e dos dividing clusters. Enquanto os primeiros serão utilizados como clusters iniciais das centerlines, os segundos são clusters finais.

Por exemplo, na figura 14b a primeira centerline extraída se inicia no LMcluster com single-code 19 e vai sempre buscando o próximo voxel (do cluster adjacente). Quando esta busca chega ao cluster com código 17, este cluster é marcado (figura 15a), por ser um dividing cluster (isto indica que quando alguma outra centerline chegar a este cluster, será finalizada). A busca pelo próximo cluster continua. Ao chegar ao merging cluster com código 13, apenas um dos cluster adjacentes será escolhido para dar continuidade à centerline. Então este cluster com código 13 é marcado (figura 15b) de modo que, quando todas as centerlines forem extraídas a partir de LMclusters, uma nova centerline será extraída a partir dele. A busca continua até que chega ao dividing cluster com código 5. Este cluster é marcado, indicando que, quando alguma outra centerline chegar até ele, deve ser finalizada neste ponto (figura 15c). A busca continua até que o cluster com código 0 é processado, finalizando assim a primeira centerline. Ou seja a primeira centerline contém um voxel (neste caso pixel) dos clusters com código: 19, 18, 17, 16, 15, 14, 13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1 e 0, nesta ordem (figuras 15d e 16a).

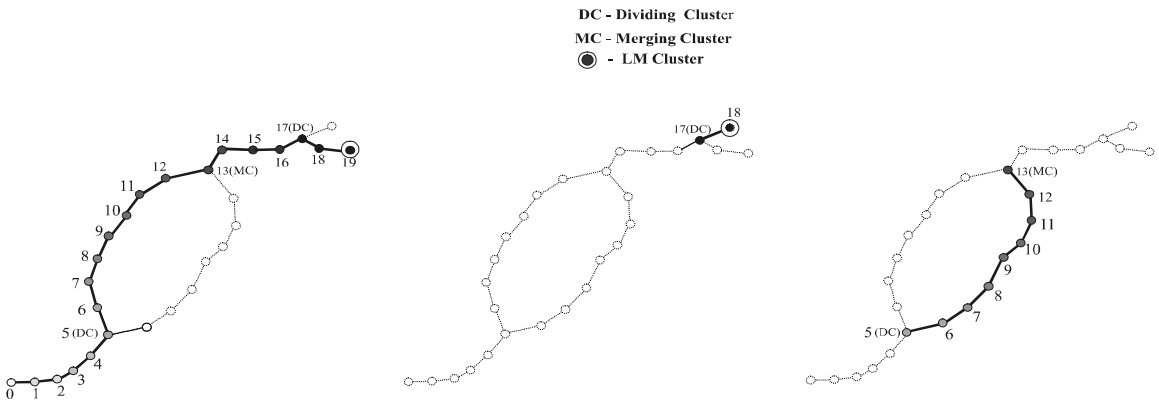


a) Dividing cluster b) Merging cluster c) Dividing Cluster d) Sequência de clusters

Figura 15: Extração da centerline que se inicia no maior LMcluster

A segunda centerline é extraída a partir do LMcluster com código 18. Quando a busca chega ao diving cluster de código 17, verifica-se que este já foi processado e portanto a segunda centerline deve ser finalizada, passando então pelos clusters de código 18 e 17 (figura 16b).

Como acabaram os LMclusters serão agora extraídas as centerlines a partir dos merging clusters. O único é o cluster de código 13. A terceira centerline começa neste merging cluster e é finalizada no dividing cluster de código 5, pois este já foi processado pela primera centerline. A terceira centerline fica, então, composta pelos clusters de código: 13, 12, 11, 10, 9, 8, 7, 6 e 5 (figura 16c).



a) Primeira centerline b) Segunda centerline c) Terceira centerline

Figura 16: As três centerlines extraídas do objeto.

Após a extração de todas as centerlines, este método aplica um pós-processamento no esqueleto para garantir que os voxels de fato fiquem conectados uns aos outros e também para que o esqueleto tenha uma aparência suavizada.

4-Métodos de *Thinning*

A idéia básica dos métodos de thinning é que as camadas do objeto vão sendo iterativamente removidas. Esta subtração, camada por camada, é feita através da identificação de voxels cuja remoção não afeta a topologia do objeto. Isto, em geral, é feito através de um processo repetitivo, com considerável tempo de processamento dedicado a testes e remoções, uma vez que não são quaisquer tipos de voxels que podem ser removidos. Esta remoção deve levar em conta a conectividade do objeto. Alguns métodos de cálculo de esqueletos através de operações de thinning podem ser encontrados em [20][14][13][9][2].

4.1-Preservação da Topologia do Objeto

Uma das maiores preocupações dos métodos de thinning é que os esqueletos extraídos preservem a topologia dos objetos. Para manter a topologia é essencial que os voxels do esqueleto estejam conectados corretamente. Para garantir que o método calcula o esqueleto de forma correta, é necessário demonstrar que ele preserva a topologia do objeto. Em [12], Ma estabelece condições suficientes para que um algoritmo de thinning de objetos volumétricos preserve a topologia.

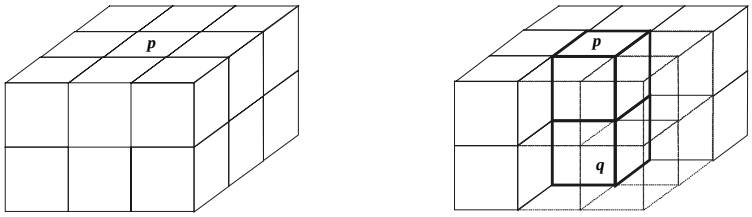
Como definido na seção 2.1, um volume de dados binários contém voxels com valores 0 (background) e com valores 1 (objeto). Mesmo que o volume não seja binário, a separação entre background e objeto é determinada a partir da função de densidade do volume e de uma densidade de busca.

Uma operação sobre o volume binário é chamada de *redução* se ela converte voxels de valor 1 para o valor 0. Em outras palavras, uma redução converte voxels do objeto para o background, ou simplesmente remove voxels do objeto. Assim thinning é uma operação de redução. Neste sentido, uma operação de thinning *não* preserva a topologia do objeto se:

a) qualquer objeto no volume de dados é dividido em dois ou mais objetos ou é completamente removido,
b) qualquer cavidade do objeto é unida ao background ou a outra cavidade, ou
c) uma nova cavidade é criada no objeto.

Um voxel pertencente ao objeto será chamado de *neutro* caso a sua remoção não altere a topologia do objeto. Os algoritmos de thinning são ditos paralelos, uma vez que removem simultaneamente, a cada iteração, todos os voxels neutros da borda do objeto (é natural pensar assim, uma vez que o paradigma da evolução de uma frente em chamas representa um processo paralelo, ou seja, a cada instante as chamas avançam para o interior da região em todas as direções).

A remoção de um único voxel neutro, em geral, não altera a topologia do objeto, porém a remoção de dois ou mais voxels neutros, simultaneamente, pode alterar esta topologia. A figura 17a mostra um conjunto de voxels de um objeto, onde os voxel *p* e *q* estão situados no centro do conjunto, conforme mostra a figura 17b, que destaca estes dois voxels. Se o voxel *p* ou o voxel *q* for removido do objeto, sua topologia não é alterada. Porém a remoção simultânea dos dois voxels altera esta topologia, pois cria uma cavidade no meio do objeto. Como nos algoritmos de thinning paralelo há remoção simultânea de um conjunto de voxels neutros a cada iteração, então, para que o método seja considerado válido, é necessária a demonstração de que a topologia é preservada, quando ocorrer a remoção dos voxels.



a) Objeto com 18 voxels b) Voxels *p* e *q* em destaque

Figura 17: A remoção simultânea de *p* e *q* altera a topologia do objeto.

4.2-Metodologias Utilizadas

De um modo geral, os métodos de thinning podem ser implementados a partir do algoritmo básico:

Repita
Remova todos os voxels que podem ser removidos durante esta iteração.
Até que nenhum voxel tenha sido removido.

A idéia deste algoritmo é que, a cada vez que o laço acima for executado, uma camada do objeto seja removida.

Existem basicamente três categorias principais de algoritmos de thinning, definidas a partir do núcleo da estrutura de repetição do algoritmo acima: algoritmos sem sub-iterações, algoritmos com sub-iterações e algoritmos com sub-divisões do volume .

Cada uma das categorias depende basicamente da forma como os voxels do objeto são removidos.

Algoritmos sem Sub-Iterações. Nesta categoria a iteração não é dividida em sub-iterações. A cada iteração todos os voxels do objeto são testados e os voxels que satisfizerem a uma condição global são removidos.

Os algoritmos sem sub-iterações podem ser descritos como:

Repita

Remova simultaneamente os voxels que satisfazem à condição global.

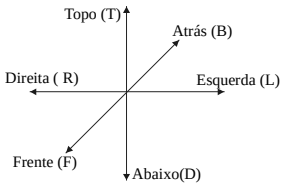
Até que nenhum voxel tenha sido removido.

A condição global de remoção dos voxels depende de critérios que o método de thinning pré-estabelece. Exemplos de dois algoritmos desta categoria foram implementados em [13] e [14]. De modo que a topologia dos objetos seja preservada, estes dois algoritmos investigam alguns pontos em uma vizinhança 5x5x5, em vez de usar uma vizinhança 3x3x3.

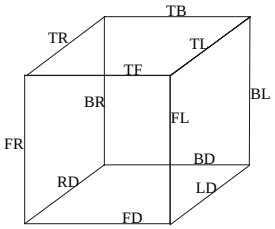
Algoritmos com Sub-Iterações. Esta categoria de algoritmos investiga, a cada iteração, a vizinhança 3x3x3 de cada voxel da borda do objeto volumétrico. Cada iteração destes algoritmos é dividida em um número de sucessivas sub-iterações. Cada sub-iteração utiliza uma regra diferente para a remoção de voxels. Cada sub-iteração é executada em paralelo, ou seja, todos os voxels da borda que satisfazem à condição de remoção da sub-iteração corrente são removidos simultaneamente. Estes algoritmos são também chamados de *direcionais* ou *seqüenciais por bordo.* A maioria dos algoritmos de thinning paralelos pertencem a esta categoria.

Como em dados volumétricos há 6 direções principais (à frente, atrás, à direita, à esquerda, ao topo e abaixo) , a maioria destes algoritmos propõe 6 sub-iterações, uma para cada direção principal [9][2][19]. A figura 18a mostra as 6 principais direções (estas direções correspondem aos vizinhos por faces de um voxel).

Outros algoritmos [20][28] usam 12-subiterações, correspondentes às direções dos 12 vizinhos por arestas de um voxel. A figura 18b mostra estas 12 direções.



a) 6 direções principais



b) 12 direções principais

Figura 18: Direções que definem as sub-iterações do thinning.

Os algoritmos desta categoria, com *k* sub-iterações, podem descritos como:

Repita

Para i=1 até k faça

Remova simultaneamente os voxels da borda que satisfazem à condição associada à sub-iteração k.

Até que nenhum voxel tenha sido removido.

Algoritmos com Sub-divisões do volume. Nesta categoria de algoritmos o volume de dados é dividido em subconjuntos disjuntos. Dentre estes subconjuntos apenas um é ativado por vez. A cada iteração, apenas os voxels da borda do subconjunto ativo são investigados e, dependendo da condição de remoção estabelecida àquele subconjunto, os voxel são removidos. Cada subconjunto tem a sua própria condição de remoção de voxels.

Um algoritmo desta categoria aplicado a um volume de *k* subconjuntos pode ser descrito como:

Repita
    Para i=1 até k faça
        Remova simultaneamente os voxels da borda do k-ésimo subcojunto, que satisfazem a condição de remoção deste subconjunto.
Até que nenhum voxel tenha sido removido.

Dois casos possíveis de divisão do volume em subconjuntos são:

- Dividir o volume em dois subconjuntos, onde cada subconjunto é formado por voxels conectados por arestas, ou seja, cada dois voxels *p* e *q* pertencem a um mesmo subconjunto se e somente se existir uma seqüência de voxels 18-connected (figura 8b) entre *p* e *q*, mas os voxels adjacentes desta seqüência são vizinhos por aresta (e não por faces).
- Dividir o volume em quatro subconjuntos, onde cada um dos subconjuntos é formado por voxels conectados por vértices, ou seja, cada dois voxels *p* e *q* pertencem a um mesmo subconjunto se e somente se existir uma seqüência de voxels 26-connected (figura 8c) entre *p* e *q*, mas os voxels adjacentes desta seqüência são vizinhos por vértices (e não por faces, nem por arestas).

Dois algoritmos para thinning 3D baseados na divisão do volume em subconjuntos são apresentados em [1] e [23]. Ambos os algoritmos dividem o volume em oito subconjuntos. O uso destes oito subconjuntos é utilizado para garantir a preservação da topologia do objeto.

5-Conclusões

Este trabalho apresentou uma conceituação geral de esqueletos de objetos gráficos, dando ênfase em esqueletos de objetos gráficos volumétricos. Os esqueletos são uma forma de se ter uma representação compacta, contendo as informações geométricas centrais do objeto, sem perder sua topologia.

A extração de esqueleto ou eixo medial de um objeto é baseada no paradigma de uma frente em chamas que avança, queimando uma região coberta de gramas. Quando é ateado fogo na borda de uma região coberta de grama (frente inicial), esta vai queimando (frente em evolução) até que as chamas se encontram no interior da região. O local do encontro define o esqueleto da região. Dentre os vários métodos de cálculo de esqueletos inspirados neste paradigma, destacam-se: os métodos baseados em uma transformada distância, os métodos de thinning ou erosão e os métodos de Voronoi.

Objetos volumétricos são objetos discretos compostos por uma grande quantidade de pequenos blocos, denominados voxels. O valor associado a cada voxel (através da função de densidade) decide quais voxels compõem o objeto, em si, e quais voxels fazem parte do *background.* O cálculo de esqueletos a partir de objetos volumétricos deve resultar em uma seqüência de voxels conectados e centralizados em relação ao objeto, que podem representar uma curva (eixo medial) ou uma superfície (superfície medial).

Os métodos para o cálculo de esqueletos de objetos volumétricos devem garantir algumas propriedades destes esqueletos, como: conectividade, ou seja, os voxels do esqueleto devem estar conexos, para garantir que o esqueleto mantém a mesma topologia do objeto; posição geométrica centralizada em relação ao objeto; sensibilidade a ruídos, ou seja, pequenas perturbações na borda do objeto não devem alterar o esqueleto de forma significante; cálculo computacional dentro de uma intervalo de tempo tolerante, uma vez que objetos volumétricos exigem muito processamento.

Os principais métodos utilizados para extrair esqueletos de objetos volumétricos são os métodos baseados em transformada distância e os métodos de thinning.

Os métodos baseados em transformada distância extraem o esqueleto diretamente como o conjunto de voxels internos eqüidistantes em relação à borda do objeto. O cálculo da transformada é feito através de alguma métrica. Dentre as métricas mais usadas está a euclidiana, que, apesar de muito precisa, apresenta problemas de eficiência computacional e dificuldade de implementação quando aplicada a objetos discretos. Para substituir a métrica euclidiana algumas opções são a métrica city block, também chamada de 6-connected ou métrica “1-2-3”, a métrica chessblock, também chamada de 26-connected ou métrica “1-1-1” e a métrica “n_f-n_a-n_v”.

Os métodos de thinning calculam o esqueleto, iterativamente, através da remoção de camadas do objeto. A cada iteração, um conjunto de voxels com determinadas propriedades é removido, até que reste apenas o esqueleto do objeto volumétrico.

Bibliografias

[1] Bertrand, G. & Aktouf, Z. 1994. A 3D Thinning Algorithms Using Subfields. *SPIE Conference on Vision Geometry III.* **2356**, 113-124.

[2] Bertrand, G. 1995. A Parallel Thinning Algorithm for Medial Surface. *Pattern Recognition. Lett.* **16**, 979-986.

[3] Blum, H. 1967. A Transformation for Extrating New Descriptors of Shape. Wather-Dunn, W., editor. *Models for the Perception of Speech and Visual Form.* MIT Press, Cambridge MA.

[4] Blum, H. 1973. Biological Shape and Visual Science, Part I. *J Theo Biol*, **38**, 205-287.

[5] Borgefors, G. 1986. Distance Transformations on Digital Images. *Computer Vision and Image Processing*, **34**, 344-371.

[6] Dorst, L. 1986. Pseudo-Euclidian Skeletons. *The 8th International Conference on Pattern Recognition, Paris France, Washington, D.C. IEEE Computer Society Press, Los Angeles, CA.* 286-289.

[7] Gomes, J. M. & Velho, L. 1995. Abstractions Paradigms for Computer Objects. *The Visual Computer*, **11**, 227-239.

[8] Gomes, J. M.; Costa, B.; Darsa, L. & Velho, L. 1996. Graphical Objects. *The Visual Computer*, **12**, 269-282.

[9] Gong, W. X. & Bertrand, G. 1990. A Simple Parallel 3D Thinning Algorithm. *Proceedings 10th IEEE International Conference on Pattern Recognition.* 188-190.

[10] Hong, L.; Kaufman, A.; Wei Y.; Viswambharn, A.; Wax, M. & Liang, Z. 1995. 3D Virtual Colonoscopy. *Proceedings of the 1995 Symposium on Biomedical Visualization Atlanta Ga., IEEE Computer Society Press, Los Angeles, CA*, 26-32.

[11] Levoy, M. 988. Efficient Ray Tracing of Volume Data. *ACM Transaction on Graphics*, **9**, 29-37.

[12] Ma, C. M. 1994. On Topology Preservation in 3D Thinning. *Computer Vision, Graphics and Image Processing: Image Understanding*, **59**, 328-339.

[13] Ma, C. M. 1995. A 3D Fully Parallel Thinning Algoritmo for Generating Madial Faces. *Pattern Recognition. Lett.* **16**, 83-87.

[14] Ma, C. M. & Sonka, M. 1996. A Fully Parallel 3D Thinning and Its Applications. *Computer Vision and Image Understanding*, **64**, 3, 420-433.

[15] Niblack, C. W.; Gibbons, P. B. & Capson, D. W. 1992. Generating Skeletons and Centerlines from the Distance Transform. *Graphical Models and Image Processing.* **54**, 5, 420-437.

[16] Ogniewicz, R. L. 1993. Discrete Voronoi Skeletons. Hartung-Gorre.

[17] Ogniewicz, R. L. & Kubler, O. 1995. Hierarchic Voronoi Skeletons. *Pattern Recognition*, **28**, 3, 343-359.

[18] Paik, D. S.; Beaulieu, C. F.; Jeffrey, R. B.; Rubin, G. D. & Napel, S. 1998. Automated Flight Path Planning for Virtual Endoscopy. *Med Phys*, **25**, 629-637.

[19] Palágyi, K & Kuba, A. 1998. A 3D 6-Subteration Thinning for Extracting Medial Lines. *Pattern Recognition Lett*, **19**, 613-627.

[20] Palágyi, K & Kuba, A. 1999. A Parallel 3D 12-Subiteration Thinning Algoritm. *Graphical Models and Image Processing*, **61**, 199-221.

[21] Pizer, S. M.; Eberly D. & Fritcsh, D. S. 1998. Zoom-Invariant Vision of Figural Shape: The Mathematics of Cores. *Computer Vision and Image Understanding*, **69**, 1, 55-71.

[22] Rosenfeld, A. & Pfaltz, J. L. 1996. Sequential Operations in Digital Pictures Processing. *J ACM*, **13**, 471-494.

[23] Saha, P. K.; Chaudhury, B. B. & Majumder, D. D. 1997. A new Shape-Preserving Parallel Thinning Algorithm for 3D Digital Images. *Pattern Recognition*, **30**, 1939-1955.

[24] Saito, T. & Toriwaki, J. I. 1994. New Algorithms for Euclidian Distance Transformation of a n-Dimensional Digitized Pictures with Applications. *Pattern Recognition*, **27**, 11, 1551-1565.

[25] Seixas, R. B. 1997. Visualização Volumétrica com Ray-Casting num Ambiente Distribuído. Tese de Doutorado, Departamento de Informática, Pontifícia Univerdade Católica do Rio de Janeiro .

[26] Sheery, D. J.; Armstrong, C. G. & Robinson, D. J. 1996. Shape Description by Medial Surface Construction. *IEEE Trans. Visualization Comput. Graph.*, **2**, 62-72.

[27] Sherbrooke, E. C.; Patrikalakis, N. M. & Brisson, E. 1996. An Algorithm for the Medial Axis Transform of 3D Polyhedral Solids. *IEEE Trans. Visualization Comput. Graph.*, **2**, 44-61.

[28] Srihari, S. N.; Udupa, J. K. & Yau, M. M. 1979. Understanding the Bin of Parts. *Proceedings IEEE Conference on Decision Control.*, 44-49.

[29] Sudhalkar, A.; Gursoz & Prinz, F. 1996. Box-skeletons of Discrete Solids. *Computer Aided Design.*, **28**, 507-517.

[30] Zhou, Y.; Kaufman, A. & Toga, A. W. 1998. Three-Dimensional Skeleton and Centerline Generation Based on an Approximate Minimum Distance Field. *Visual Computer*, **14**, 303-314.

[31] Zhou, Y. & Toga, A. W. 1999. Efficient Skeletonization of Volumetric Objects. *IEEE Transactions on Visualization and Computer Graphics* , **5**, 3, 196-209.