5
Rendering Háptico e Transformadas de Distância

Este capítulo apresenta a estrutura básica de uma aplicação háptica convencional e mostra a estreita relação do *rendering* háptico com o modelo matemático de transformadas de distância. Conforme será visto, este é o modelo matemático mais adequada em interfaces espaciais reativas, pois está diretamente relacionado com propriedades geométricas de objetos gráficos que estão sendo manipulados pelo usuário.

5.1 Rendering Háptico

De uma forma geral, os sistemas que utilizam dispositivos hápticos possuem um *loop* principal semelhante ao descrito na Figura 5.1. Os passos que apresentam maiores dificuldades são 2 e 3. A detecção de colisões não é tão complexa, mas pode requerer a utilização de otimizações, tais como *bounding boxes*, *BSP's*, *quad-trees* etc. Entretanto, quando o *cursor* é representado por um ponto no espaço, a detecção de colisões torna-se muito mais simples. O passo 3 é, sem dúvida, o mais complexo. Freqüentemente recorre-se a modelos físicos para obter uma aproximação da força resultante.

Existe um diferencial importante nos sistemas que utilizam dispositivos hápticos. Nos sistemas tradicionais de visualização, uma taxa de atualização do *display* em torno de 30 Hz já é suficiente para dar a impressão aos nossos olhos de continuidade de movimento. Entretanto, isso não ocorre nos sistemas que utilizam dispositivos hápticos. Para criar a ilusão do tato é necessária uma taxa de atualização de forças muito maior, da ordem de 1000 Hz ou mais. Por sorte, enquanto a atualização do *display* requer o cálculo de centenas de milhares de *pixels*, a atualização de uma força requer um esforço computacional comparativamente muito menor. No caso mais comum de dispositivos onde a força é aplicada em um único ponto, apenas um *pixel* precisa ser atualizado.

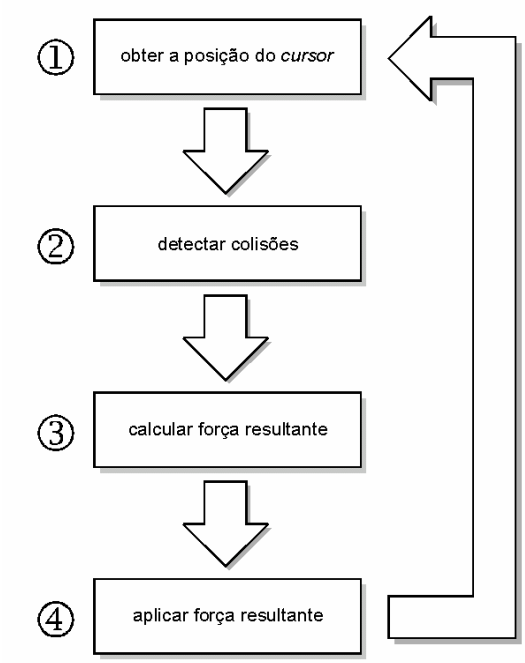
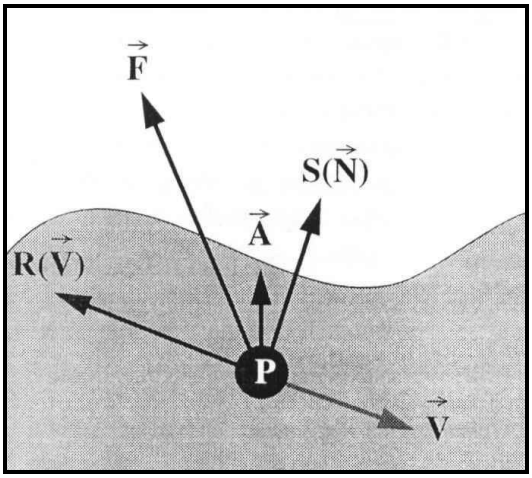


Figura 5.1: Loop principal do rendering háptico

A Figura 5.2 ilustra a equação geral que descreve a força a ser aplicada em ponto (Avila & Sobierajski, 1996). Nesta equação, **F** é o vetor força resultante, **A** é o vetor força ambiente, R(**V**) é o vetor força de retardo ao movimento do cursor (que se move com velocidade **V**) e S(**N**)é o vetor força de resistência do objeto cuja normal à superfície no ponto de contato com o *cursor* é **N**.

Nesta modelagem, a força ambiente (**A**) corresponde ao termo que não depende diretamente dos objetos da cena. Pode também não depender da posição do *cursor* (exemplo: força gravitacional) ou ser uma função dela. Neste último caso, algoritmos de atração do *cursor* (a uma superfície de um determinado objeto da cena, a um ponto da grade, a uma reta ou plano de referência etc.) podem ser implementados e utilizados para facilitar a interação com os objetos da cena. A força de retardo ao movimento (R(**V**)), proporcional à velocidade de movimento do *cursor*, pode ser utilizada para simular viscosidade do meio. Por fim, a força de resistência do objeto (S(**N**)) é uma indicação de sua rigidez. Esta força é aplicada na direção do campo gradiente e sua intensidade é tipicamente proporcional à penetração do cursor dentro do objeto (*Lei de Hook*).



$$\mathbf{F} = \mathbf{A} + R(\mathbf{V}) + S(\mathbf{N})$$

Figura 5.2: Equação geral da força resultante

A equação geral apresentada acima pode ser utilizada tanto para cenas contendo apenas objetos geométricos, apenas objetos volumétricos ou ambos. Tipicamente, em uma cena contendo apenas objetos geométricos, o termo que representa a viscosidade do meio ($R(\mathbf{V})$) não é considerado e a força de resistência do objeto ($S(\mathbf{N})$) só é efetivamente calculada quando há colisão do *cursor* com algum objeto da cena. Nesta situação, normalmente utiliza-se a *Lei de Hook* como forma de calcular a força de reação.

O grande problema em aplicar essa lei é que o vetor resultante será sempre perpendicular à superfície do objeto. Devido à inexistência de forças tangenciais, o usuário terá a impressão de estar manipulando um objeto extremamente polido. Por isso, este modelo não é capaz de simular adequadamente fricção estática e texturas. Massie (1998a) apresenta uma forma simples de contornar essa limitação.

5.2 Transformada de Distância

A transformada de distância T de um objeto O, $T(O)$, define, para cada ponto do espaço, a distância mínima deste ponto ao objeto O e, opcionalmente, o ponto mais próximo pertencente a O (Figura 5.3). Esta informação é bastante útil em computação gráfica, podendo ser utilizada para produzir metamorfose de objetos 3D (Cohen-Or et al., 1998), reconstruir superfícies a partir de fatias

bidimensionais (Peixoto & Gattass, 2000) e calcular de esqueletos de objetos volumétricos (Peixoto & Carvalho, 2000).

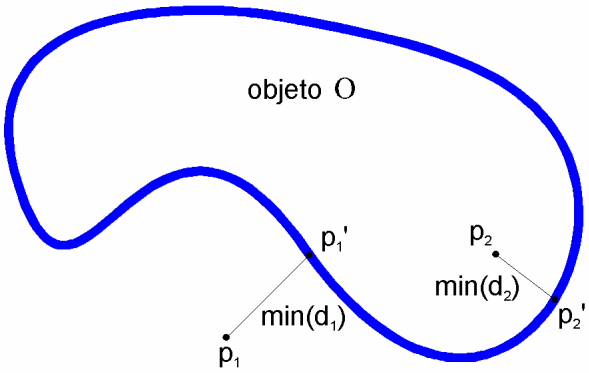


Figura 5.3: Transformada de Distância

Apesar de ser conceitualmente simples, o cálculo da transformada de distância não é trivial. A maior dificuldade está relacionada com a natureza contínua do espaço onde se encontra o objeto cuja transformada de distância se quer calcular. Neste caso, a transformada também seria uma função contínua e isto implica em grandes dificuldades para representá-la no computador. Por isso, há necessidade de discretizar o mundo caso seja necessário calcular a transformada de distância em uma etapa de pré-processamento.

Dependendo da aplicação, pode ser conveniente utilizar uma determinada métrica para o cálculo de distâncias. A mais natural, a métrica Euclidiana, computa a distância através da expressão:

$$\mathtt{dist}(\mathbf{p},\mathbf{q}) = \|\mathbf{p}-\mathbf{q}\|$$

Figura 5.4: Métrica Euclidiana

Esta métrica é a ideal, pois além de computar a distância real entre dois pontos, pode ser utilizada tanto para objetos definidos no espaço contínuo como para objetos no espaço discreto. Entretanto, seu custo computacional é elevado, sendo, por isso, muitas vezes evitada. A Figura 5.5 ilustra o cálculo da transformada de distância de um objeto geométrico (uma garrafa) utilizando essa métrica. Nessa figura, quando mais escuro for o *pixel*, mais distante está da borda do objeto. Outras métricas mais simples também podem ser utilizadas como, por exemplo, *city block* e *chessboard* (Peixoto & Velho, 2000). Entretanto, essas

métricas só podem ser aplicadas para objetos definidos no espaço discreto como, por exemplo, dados puramente volumétricos.

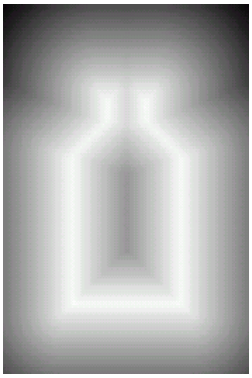
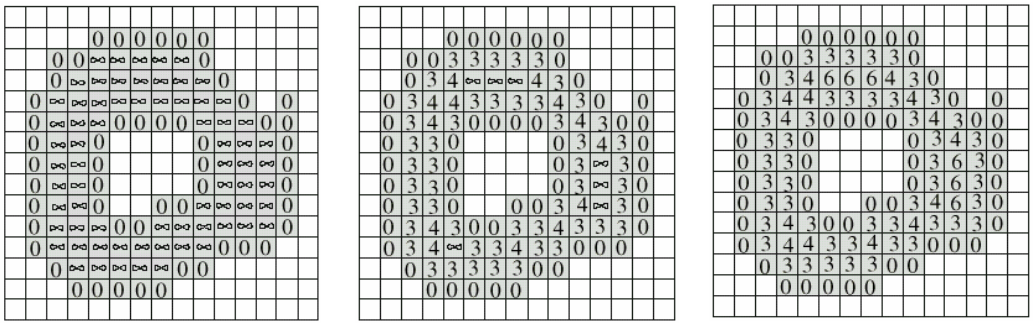


Figura 5.5: Campo de distância, métrica Euclidiana (Mauch, 2000)

A transformada de distância de um objeto tipicamente é representada no computador através de uma matriz de *pixels* (uma imagem em 2D ou volume em 3D), ou seja, a solução do problema está no espaço discreto. Entretanto, o problema pode ser formulado no espaço contínuo ou discreto, de acordo com a representação utilizada pelos objetos. A formulação contínua, por ser mais precisa, é mais utilizada em modelagem geométrica (por exemplo, reconstrução de superfícies) enquanto a formulação discreta é menos precisa e mais utilizada em visão computacional.

5.2.1 Transformada de um Objeto Discreto

O cálculo da transformada de distância de um objeto discreto definido em um mundo discreto (uma imagem ou volume) não apresenta maiores dificuldades algorítmicas. O processo se inicia a partir de um subconjunto de *pixels/voxels* do mundo e se propaga para os demais *pixels*/*voxels* vizinhos, armazenando-se neles a distância de acordo com a métrica utilizada. O processo se repete até que todos os *pixels*/*voxels* do mundo sejam visitados. Isto resulta uma nova imagem ou volume onde cada *pixel* ou *voxel* armazena a distância mínima ao objeto considerado (Figura 5.6).



(a) Pontos do objeto (∞´s) e conjunto inicial V_0 (0´s)

(b) Processamento dos pixels de código 0

(c) Campo distância final

Figura 5.6: Campo de distância, métrica 3-4 (Peixoto & Velho, 2000)

5.2.2 Transformada de um Objeto Contínuo

Quando o objeto em questão é definido através de uma representação contínua, o cálculo de sua transformada de distância torna-se mais complexo. Uma das alternativas é utilizar a teoria de evolução de interfaces, uma analogia a propagação de uma frente de onda em um determinado meio (Figura 5.7). Pode-se citar dois métodos eficientes dessa natureza: *Fast Marching* (Sethian, 1996) e *Level Set* (Sethian, 1999).

O método *Fast Marching* parte de uma interface inicial (que tipicamente corresponde à borda do objeto considerado) e, a cada instante, calcula a nova posição da interface, segundo a equação *eikonal*. Desta forma, a informação de distância é incrementalmente calculada e propagada juntamente com a interface. Neste método, a interface só pode propagar em uma única direção, permitindo maior eficiência. O método *Level Set* é mais genérico e menos eficiente, por não possuir essa restrição.

Uma das dificuldades encontradas nos métodos baseados na evolução de interfaces está relacionada com o "mal comportamento" desta evolução. Em algumas situações, a interface pode perder a suavidade, apresentando singularidades e interseções com as demais interfaces. Essas situações devem ser evitadas, pois invalida a idéia de propagação da distância de acordo com a evolução da interface. Além disso, com a perda da diferenciabiliade da interface, o vetor normal à interface também fica indefinido. Peixoto & Velho (2000) fazem uma discussão mais detalhada sobre os métodos mencionados.

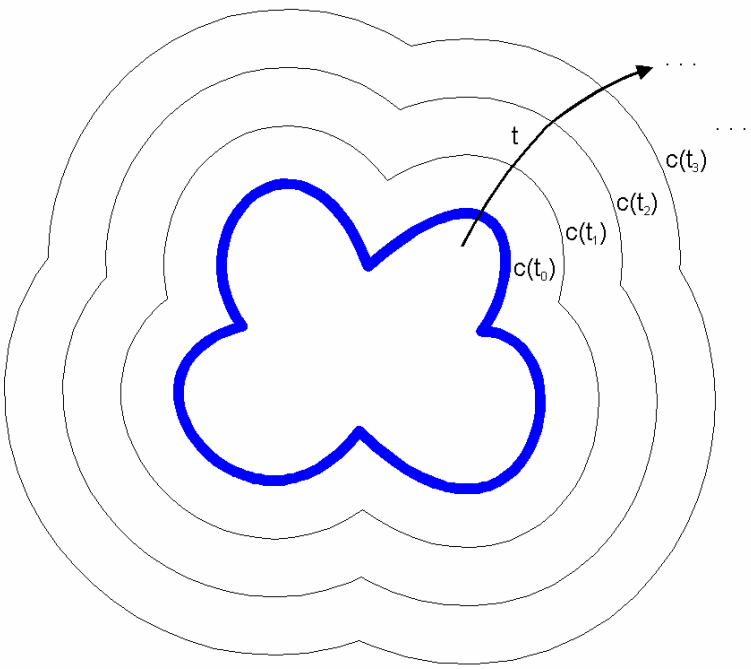
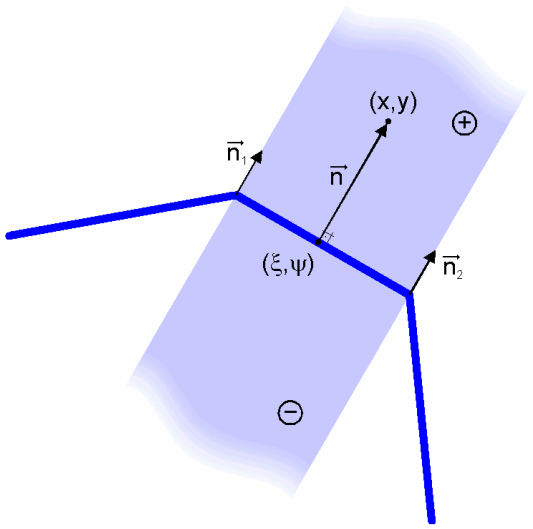


Figura 5.7: Evolução de interfaces

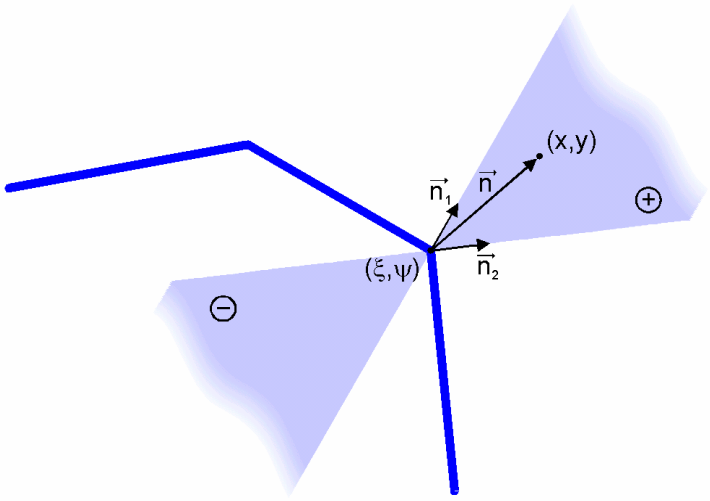
5.2.3 Algoritmo de Mauch

Um outro método para o cálculo da transformada de distância para objetos com representação contínua é apresentado por Mauch (2000). Este método só pode ser aplicado para curvas lineares por partes (no caso 2D) ou superfícies definidas por uma malha triangular (no caso 3D). Portanto, caso o objeto possua outra representação, é necessário aproximá-la antes de calcular a transformada de distância.

No caso 2D, o método de Mauch utiliza a seguinte observação: para um dado ponto (x,y) do espaço, o ponto (ξ,ψ) mais próximo em uma curva linear por partes ou pertence a um segmento da curva ou corresponde a um vértice da mesma. No primeiro caso, quando (ξ,ψ) pertence a um segmento da curva, o vetor $\mathbf{n}$ de (x,y) a (ξ,ψ) é ortogonal ao segmento considerado. Portanto, (x,y) deve estar localizado na região do espaço definido pelo segmento da curva e os vetores $\mathbf{n_1}$ e $\mathbf{n_2}$ normais à curva em seus dois vértices (Figura 5.8a). No segundo caso, quando (ξ,ψ) corresponde a um vértice da curva, o vetor $\mathbf{n}$ de (x,y) a (ξ,ψ) está entre os vetores $\mathbf{n_1}$ e $\mathbf{n_2}$ normais aos segmentos adjacentes do vértice considerado (Figura 5.8b).



(a)



(b)

Figura 5.8: Algoritmo de Mauch

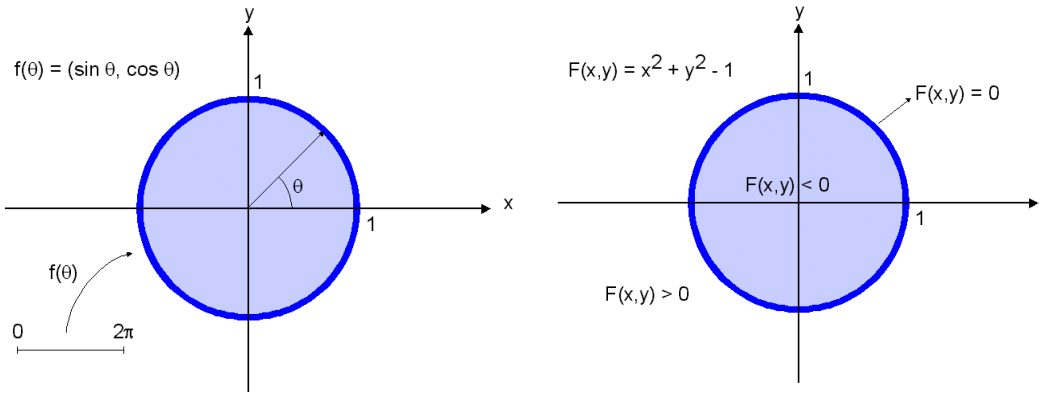
A observação acima permitiu implementar um algoritmo simples para o cálculo da transformada de distância de um objeto. Ainda no caso bidimensional, o espaço é discretizado e todos os pontos da grade são assumidos inicialmente distância infinita ao objeto. Para cada segmento linear do objeto, o algoritmo define a região do espaço onde os possíveis pontos `(x,y)` podem pertencer. Através de *scan conversion*, o algoritmo seleciona todos os pontos da grade desta região e atualiza a distância de cada ponto ao segmento de reta, caso esta seja menor que a já calculada. O mesmo procedimento é utilizado para cada vértice da curva. Ao final do algoritmo, cada ponto da grade armazenará a menor distância ao objeto.

O algoritmo apresentado pode ser otimizado de duas formas. A primeira é reduzindo o tamanho das regiões antes de realizar o *scan conversion* (*local clipping*) e a segunda reduzindo o tamanho das regiões onde a curva é côncava (*global clipping*). Este algoritmo também pode ser facilmente estendido para o caso 3D.

5.3 Vantagens da Utilização de Transformadas de Distância

Essencialmente, a função mais importante do *rendering* háptico é o cálculo (e atualização) de forças. Quando o usuário toca a superfície de um objeto virtual o algoritmo responsável pelo *rendering* háptico calcula a força reativa adequada de forma tornar possível a simulação da presença do objeto virtual e de diversas propriedades físicas táteis, tais como rigidez e texturas, conforme será apresentado no Capítulo 6.

A geometria de um objeto (superfície ou sólido) pode ser descrita de duas formas diferentes: utilizando o modelo paramétrico ou implícito (Gomes & Velho, 1992), conforme mostra a Figura 5.9. A forma mais comum utiliza modelos paramétricos, tais como polígonos, *splines* e NURBS. Este modelo fornece uma representação explícita da superfície e é bastante eficiente em algumas situações, tais como, desenho (*rendering* visual), subdivisão, *clipping* etc. Entretanto, representações paramétricas não trazem diretamente informações sobre o interior e exterior dos objetos. Tais modelos necessitam de estruturas de dados, tipicamente representações de borda (*B-rep*), para a maioria das operações.



(a) descrição paramétrica (b) descrição implícita

Figura 5.9: Descrição da Geometria de Objetos

Objetos implícitos, por outro lado, são definidos através de uma função implícita que está diretamente associada com a superfície e todas as isosuperfícies (ou curvas de nível) de um objeto. Neste trabalho, a função implícita que representa um objeto corresponde a transformada de distância deste objeto. Esta representação é eficiente para definir a forma geométrica de um objeto e traz vantagens em relação à representação paramétrica, pois o interior, exterior e borda do objeto é facilmente identificado pelo valor da função implícita: negativo para o interior, positivo para o exterior e nulo para a borda. Detecção de colisão, portanto, torna-se trivial através desta representação implícita.

Além disto, a normal de uma superfície implícita pode ser obtida através do gradiente da função implícita (Figura 5.10a e Figura 5.10b). Uma aproximação do gradiente pode ser facilmente obtida através de diferenças finitas em uma vizinhança do campo escalar de distâncias (Figura 5.10c).

$$\mathbf{n} = \nabla\mathbf{F}/\|\nabla\mathbf{F}\| \quad (a)$$

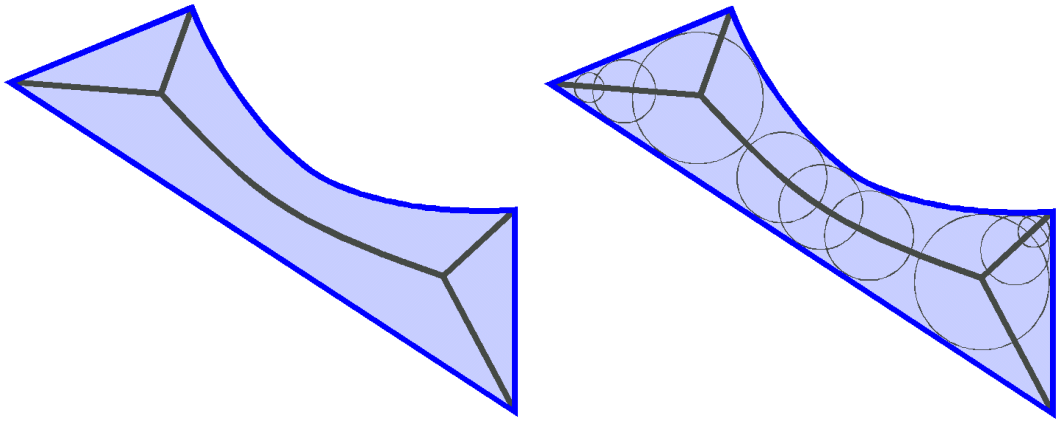
$$\nabla\mathbf{F} = (dF/dx, dF/dy) \quad (b)$$

$$\nabla\mathbf{F}_{i,j} \approx (Fx_{i+1} - Fx_{i-1}, Fy_{j+1} - Fy_{j-1}) \quad (c)$$

Figura 5.10: Normal a uma Superfície Implícita

Por fim, pode-se obter o eixo medial de um objeto através de sua transformada de distância. O eixo medial é formado pelo conjunto de pontos eqüidistantes a mais de um ponto na superfície do objeto (Figura 5.11a), ou seja é a união dos centros das circunferências de maior raio possível contidas no interior do objeto. Estas circunferências tocam as bordas do objeto em mais de um ponto (Figura 5.11b). De forma mais simples, o eixo medial corresponde exatamente aos máximos locais da transformada de distância de um objeto. Por representar propriedades geométricas e topológicas de forma bastante concisa e eficiente, o eixo medial também é chamado de esqueleto do objeto.

Por todas as propriedades mencionas acima, nota-se que a transforma de distância de um objeto define diversas características geométricas e topológicas sendo, portanto, muito importante para o *rendering* háptico e o modelo matemático mais adequado para o cálculo de forças reativas, conforme será visto no Capítulo 6.



(a) (b)

Figura 5.11: Eixo Medial

Como os modelos paramétrico e implícito são, na verdade, complementares, pode-se combinar os dois, aproveitando as vantagens oferecidas por de cada um separadamente. Neste trabalho, adotou-se o modelo paramétrico para criar a representação visual dos objetos e o modelo implícito para criar a representação háptica. Esta proposta híbrida também foi utilizada por Kim et al. (2002) para implementar um algoritmo eficiente de *rendering* háptico. Este algoritmo permite o usuário sentir a superfície de objetos tridimensionais de maneira suave, sem, no entanto, perceber descontinuidades na aplicação de forças reativas. Os autores utilizam o PHANToM e o algoritmo de Mauch para o cálculo do campo de distâncias.

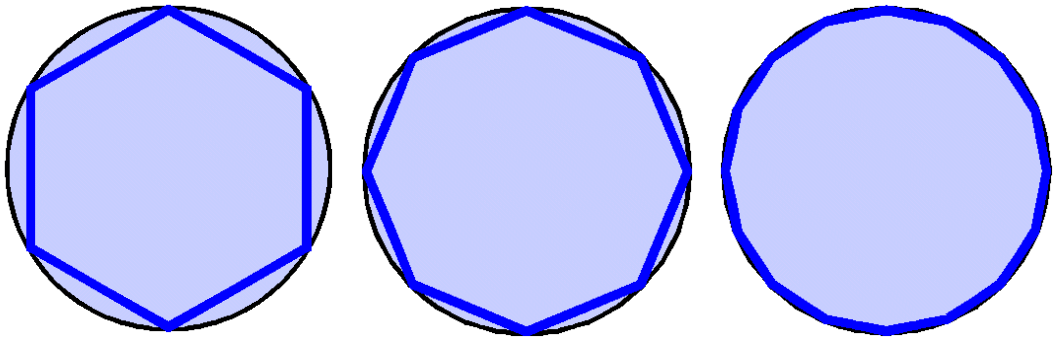
5.4 Escolha do Algoritmo para Calcular a Transformada de Distância

Conforme citado neste capítulo, existem diversos algoritmos que podem ser utilizados para gerar o campo de distância de um determinado objeto. As características de tais algoritmos dependem basicamente da representação escolhida para o objeto, que pode ser contínua ou discreta.

Geralmente o cálculo da transformada de distância de um objeto contínuo tem maior precisão, entretanto, a complexidade deste cálculo também é maior. Neste trabalho, os objetos têm representação contínua, porém são formados por linhas poligonais, ou seja, os objetos são lineares por partes. Por isto, foi escolhido o algoritmo de Mauch, pois ele é bastante eficiente neste caso particular. A complexidade do algoritmo é linear tanto em relação ao número de

pontos na grade onde as distâncias são computadas (i.é., em cada *pixel* da tela) como também na complexidade do objeto considerado.

Vale notar que além de objetos poligonais, pode-se também utilizar o algoritmo de Mauch para outros tipos de objetos, tais como círculos, elipses, *splines* e NURBS. Entretanto, será necessário aproximar o objeto de forma a tornar-se linear por partes. Dado que a representação paramétrica do objeto é conhecida, esta aproximação é simples e pode ser realizada com um nível arbitrário de precisão (Figura 5.12). Nesta tese, além de objetos lineares por partes, tais como linhas poligonais, triângulos e quadrados, foi implementado também o círculo. Entretanto, como a transformada de distância deste objeto é muito simples, optou-se, neste caso particular, não realizar esta aproximação. Desta forma, o cálculo das distâncias é realizada analiticamente.



(a) n = 6 (b) n = 8 (c) n = 16

Figura 5.12: Aproximação de uma Curva