MODELAGEM E VISUALIZAÇÃO DE OBJETOS À MÃO LIVRE USANDO SUPERFÍCIES IMPLÍCITAS VARIACIONAIS

Alvaro Ernesto Cuno Parari

TESE SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS PROGRAMAS DE PÓS-GRADUAÇÃO DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Aprovada por:

_____

Prof. Claudio Esperança, Ph.D.

_____

Prof. Antonio Alberto Fernandes de Oliveira, D.Sc.

_____

Prof. Paulo Roma Cavalcanti, D.Sc.

_____

Prof. Luiz Carlos Pacheco Rodrigues Velho, Ph.D.

RIO DE JANEIRO, RJ - BRASIL

JUNHO DE 2004

CUNO PARARI, ALVARO ERNESTO

Modelagem e Visualização de objetos à mão livre usando superfícies implícitas variacionais [Rio de Janeiro] 2004

XIV, 78 p. 29,7 cm (COPPE/UFRJ, M.Sc., Engenharia de Sistemas e Computação, 2004)

Tese – Universidade Federal do Rio de Janeiro, COPPE

1 - Modelagem a mão livre. 2 - Funções de bases radiais. 3 - Poligonização de superfícies implícitas.

I. COPPE/UFRJ II. Título (série)

Aos meus pais e irmãos,
por estarem sempre comigo.

Agradecimentos

Quero começar expressando minha maior gratidão ao meu orientador, o prof. Claudio Esperança, pela confiança dada e por ter me dado a oportunidade de trabalhar com ele. Suas idéias, sugestões e críticas positivas – no período de créditos, na determinação do tema, durante a implementação e finalmente na escrita desta dissertação – tornaram possível a culminação deste trabalho.

Ao prof. Antonio Oliveira, seu vasto conhecimento sempre foi útil para apagar as muitas dúvidas que apareceram ao longo desta jornada. Ao prof. Paulo Roma, pelas facilidades e o suporte necessário durante estes dois anos. Ao prof. Luiz Velho, membro externo da banca, por ter aceitado o convite e pelo tempo brindado.

Meu especial agradecimento aos meus colegas de turma e ao pessoal do LCG. Saulo, Disney, Cesar, Ricardo, Margareth (meg), Marcelo, Karl, Yalmar, Vitor, Leandro, Fabio, todos vocês criaram um ambiente ótimo de trabalho e amizade no laboratório. Ao Saulo meu amigo e companheiro de moradia durante bom tempo. Ao Disney e ao Cesar por me permitir conhecer mais um poco de suas familias. Ao Ricardo por me incursionar naquelas sambinhas e me levar conhecer lugares legais – Sana, Santa Teresa, etc. – e pelas boas dicas sempre que perguntei das bondades da "cidade maravilhosa". À Sissy e ao Disney por ter lido parte do texto desta dissertação, ajudando tirar vários erros ortográficos. Ao Antonio Lopes (alopes) com que tirei muitas dúvidas e pelas boas dicas durante a fase de implementação.

Ao PESC, professores e pessoal técnico-administrativo. Em particular, à Claudia, Lúcia, Solange, Sueli, Sonia e Mercedes, por me facilitar as coisas sempre que precisei.

A todos os amigos que me ajudaram quando cheguei por primeira vez no Rio de Janeiro. Diluvina muito obrigado. Alexis, Franklin, Raúl, Toninho, Jorge, obrigado por ter me ajudado, mesmo sem me conhecer.

Não posso deixar de citar à "legião estrangeira" no PESC. Em especial ao meu grande amigo e colega Erik, sempre com a mesma motivação e concentração pelos seus estudos; segue na frente amigo que está no caminho certo. À Mariela, uma pessoa muito especial,

sua espontaneidade e entusiasmo fizeram destes últimos meses tempos inesquecíveis. À Guina, Kely, Marisa, Felipe, Karl e Yalmar, com vocês ao redor quase que nem reparei que estava longe de casa ;). Ao Alberto, Manuel e Michel, pelo tempo que moramos juntos.

A todos os amigos que conheci no Rio e à turma peruana/brasileira da pelada dos sábados na Urca, entre estes, Edson Soares(eu também sou da fúria jovem!), Gladys Maquera, Mariela Berrocal, Raúl Carita, Hugo Fernández, Luis Acosta, Roberto Macoto, Jorge Zavaleta, Pedro, Ricardo.

Aos meus amigos e professores da EPIS/UNSA no Perú. Aos meus colegas de faculdade – Omar, Ochoa, David, Arturo T., Fernando, Jesús, "los cristians" (Paz, Galindo, Lopez, Fernandez), Nelly, Abigail, Gustavo, Denis, Jessica, Lizeth, Liliana, Cesar, Oliver, Paola, Eliana, Rosario, Marcela, Miluska, Herly, Soledad, Arturo P., Viviana, José M., Richard A., Maribel, Raquel –, pela sua amizade. Aos meus professores, Ernesto Cuadros, Luis Alfaro, Jesús Zuñiga e Lucy Delgado, por ter me apoiado em muitas etapas previas ao inicio deste curso.

Devo fazer um agradecimento especial ao doutor Miguel Ascón, presidente da RMCP, pelo seu constante apoio aos jovens e sua preocupação pelo estado da ciência no Perú.

Ao CNPq que deu o suporte financiero para a realização deste trabalho.

Aos meus pais Susana e Constantino que me inculcaram dedicação ao estudo e ao trabalho, aos quais devo toda minha formação acadêmica e profissional. Aos meus irmãos Delia, Enrique e Andrea, por me dar seu apoio e carinho sempre.

A Deus por me dar todos estes presentes.

Resumo da Tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc)

MODELAGEM E VISUALIZAÇÃO DE OBJETOS À MÃO LIVRE USANDO SUPERFÍCIES IMPLÍCITAS VARIACIONAIS

Alvaro Ernesto Cuno Parari

Junho/2004

Orientador: Claudio Esperança

Programa: Engenharia de Sistemas e Computação

Este trabalho estuda técnicas para a construção e visualização de superfícies de modelos 3D usando interfaces baseadas em traços como o principal paradigma de interação e superfícies implícitas variacionais como o esquema de representação. Baseado em traços 2D feitos pelo usuário, o sistema protótipo apresentado constrói modelos 3D plausíveis e permite editá-los interativamente. Além das técnicas de modelagem, apresentamos um algoritmo eficiente para poligonizar superfícies implícitas variacionais permitindo que as malhas assim obtidas possam ser visualizadas utilizando hardware gráfico padrão.

Abstract of Thesis presented to COPPE/UFRJ as partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

MODELING AND VISUALIZATION OF FREE-FORM OBJECTS USING VARIATIONAL IMPLICIT SURFACES

Alvaro Ernesto Cuno Parari

June/2004

Advisor: Claudio Esperança

Department: Computing and Systems Engineering

This work studies techniques for the construction and visualization of 3D surface models using free-hand strokes as the main interface paradigm and variational implicit surfaces as the representation scheme. Based on users 2D freeform sketching, the presented prototype system constructs plausible 3D smooth models and permits its interactive modification. Additionally, we present an efficient technique to polygonize variational implicit surfaces, thus enabling the obtained triangle meshes to be later rendered using standard graphics hardware.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Sistemas de modelagem 3D são normalmente projetados para a criação precisa e detalhada de modelos complexos [22]. Tais sistemas são pouco adequados para a criação rápida de modelos de aparência *livre*, isto é, modelos não técnicos ou exatos, e que têm um aspecto mais artístico: animais de pelúcia, por exemplo. Além disso, as interfaces estilo "WIMP" (*Windows, Icons, Menus, Pointers*) com que estes sistemas são implementados costumam representar uma dificuldade adicional para usuários casuais. Este problema sugere a necessidade de se explorar novos paradigmas de interação homem-computador que permitam a criação rápida e intuitiva desse tipo de modelos.

Modelos de aparência *livre* distinguem-se dos modelos técnicos porque, enquanto que os últimos buscam modelar objetos do mundo real com uma cuidadosa precisão, os primeiros representam uma aproximação grosseira e informal de um objeto, capturando mais suas características conceituais do que suas medidas e geometria.

Em particular, quando falamos de modelos tridimensionais de aparência *livre*, nos referimos a objetos de contornos suaves e projetados sem o auxílio de instrumentos, de maneira similar aos desenhos bidimensionais feitos à mão livre. Mais ainda, queremos distinguir esses modelos daqueles construídos usando, por exemplo, planos, cilindros, esferas e outros objetos geométricos facilmente reconhecíveis.

Em termos de representação matemática, objetos de aparência livre são tipicamente modelados através de curvas e superfícies paramétricas tais como as diversas classes de *splines*. Recentemente, entretanto, várias propostas para representação de objetos usando equações implícitas [54, 5, 52] – coletivamente denominados esquemas implícitos ou representações implícitas – vêm sendo sugeridas, particularmente aquelas que empregam as assim chamadas funções de base radiais (RBFs-*Radial Basis Functions*) [42, 50, 33, 26, 12, 27, 36, 45].

Neste trabalho, procuramos explorar os esquemas implícitos na representação da geometria dos objetos de aparência *livre*. De forma similar a Karpenko et. al. [25] elegemos uma representação implícita baseada em funções de base radiais uma vez que este tipo de representação matemática suporta operações de modelagem bastante flexíveis, além de oferecer naturalmente muitas das características desejadas, tais como, propriedades de suavidade e controle direto na criação dos modelos. Além disso, objetos implícitos oferecem algumas vantagens sobre outros esquemas de representação. Em particular, funções implícitas permitem definir superfícies complexas usando uma única função analítica e possibilitam efetuar testes de classificação de pontos do tipo interior/exterior.

Porém, esta representação obriga-nos lidar com a principal propriedade desfavorável das RBFs: o alto custo computacional para a criação e avaliação da função implícita interpolante. Desvantagem que se manifesta na criação, edição e visualização interativa dos modelos representados. Isto sugere a exploração de novas técnicas e algoritmos a fim de enfrentar estes problemas.

Por último, para a criação rápida de modelos de aparência *livre* apoiamo-nos numa interface de usuário simples, interativa e de manipulação direta, onde os traços de entrada se dão da forma menos restrita possível a fim de lograr uma interação fácil e natural.

Em resumo, nesta dissertação descrevemos uma interface baseada em traços (*sketching interface*) para a criação rápida de modelos 3D de aparência *livre*. Este paradigma de interação e a representação implícita subjacente, baseada em RBFs, permitem-nos explorar diversos algoritmos para efetuar operações de edição nos modelos. Para a criação dos mesmos, introduzimos o uso de uma malha poligonal grosseira como guia para a construção da função implícita e com a finalidade de evitar problemas de vazamentos de superfícies.

Como um mecanismo de mediação fazemos uso básico das interfaces “sugestivas” [21] para lidar com os problemas de ambigüidade inerentes às interfaces baseadas em traços. Além disso, apresentamos um algoritmo eficiente para a poligonização de superfícies implícitas baseadas em RBFs.

No capítulo 2 apresentamos os principais conceitos relacionados com os sistemas de modelagem de superfícies, tais como: esquemas de representação da geometria de objetos 3D, primitivas para a especificação de objetos implícitos, paradigmas de construção de modelos, técnicas de visualização de objetos definidos implicitamente, e uma introdução ao paradigma de interação homem-computador baseado em traços.

No capítulo 3 são introduzidas as funções de base radiais e enumeramos algumas

das suas aplicações. Revisamos as soluções propostas ao problema do alto custo computacional inerente às RBFs (funções radiais de suporte compacto e métodos iterativos aproximados). Por último, revisamos a aplicação das RBFs à Computação Gráfica, descrevemos as suas principais vantagens na area de modelagem e fazemos um resumo dos principais trabalhos realizados até a data.

No capítulo 4 propomos uma interface para modelagem de superfícies baseada em traços à mão livre. Descrevemos as operações do ponto de vista do usuário e apresentamos os algoritmos que as implementam, junto com as limitações e resultados obtidos.

O problema da visualização de superfícies implícitas, particularmente aquelas baseadas em RBFs, e o algoritmo proposto são abordados no capítulo 5. Por último, no capítulo 6 são apresentadas as conclusões e algumas idéias para trabalhos futuros.

Capítulo 2

Sistemas de modelagem 3D

Um sistema típico de modelagem é aquele que permite definir modelos que representam a forma e outras características geométricas de um objeto.

No desenvolvimento destes tipos de sistemas existem, principalmente, duas questões a serem abordadas: **representação** e **especificação**. A representação do modelo lida com o problema de como caracterizar objetos e como converter esta caracterização em estruturas concretas. A especificação do modelo se relaciona com as técnicas usadas para construir essas estruturas e a interface do sistema com o usuário [52].

2.1 Esquemas de representação

Os esquemas usados na representação da geometria de objetos e superfícies podem ser categorizados em três classes gerais: discreta, paramétrica e implícita [17].

2.1.1 Representação discreta

Nesta representação, o modelo é uma coleção de complexos simpliciais que podem incluir pontos, arestas e polígonos. A principal vantagem desta representação é a sua generalidade, ou seja, a capacidade de representar formas de topologia arbitrária com acurácia também arbitrária. O uso comum de representações poligonais, em particular de triângulos, tem conduzido ao desenvolvimento de hardware e software especializados em processos de renderização acelerada desses modelos. Entretanto, representações discretas apresentam desvantagens. Em particular, tendem a ocupar grandes quantidades de memória (complexidade de espaço) e podem apenas aproximar objetos curvos dentro de uma precisão estipulada. Veja na Figura 2.1 ilustrações desta representação.

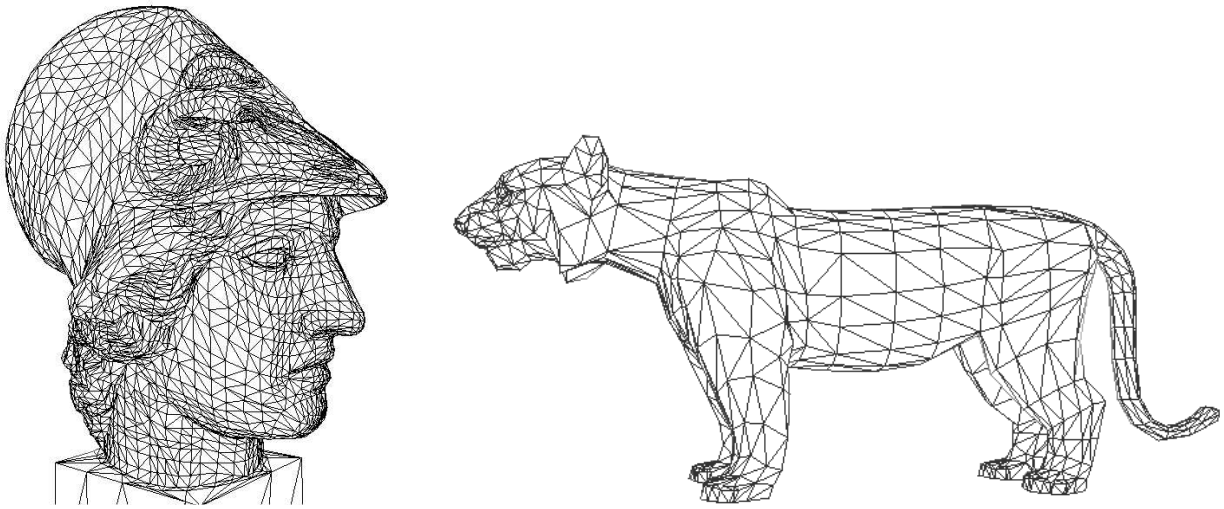


Figura 2.1: Representação discreta de modelos.

2.1.2 Representação paramétrica

A superfície é representada por retalhos (*patches*), descritos por equações paramétricas. Exemplos deste tipo de representação são retalhos *Bézier*, *Hermite* cúbicos, *B-Splines*, *NURBS (Non-Uniform Rational B-Splines)* e superfícies de subdivisão. As superfícies paramétricas podem ser amostradas em resolução arbitrária e usadas para representar superfícies suaves. A sua principal desvantagem é que para formar um objeto fechado é preciso combinar muitos retalhos, e a costura suave entre retalhos é problemática.

2.1.3 Representação implícita

Nesta representação, a superfície é um conjunto de nível (*level set surface*), e se define como o conjunto de pontos onde a função implícita assume um valor especificado, usualmente zero. As representações implícitas podem descrever objetos de topologia arbitrária e não requerem costuras para representar superfícies fechadas. Elas podem ser expressas analiticamente ou amostradas discretamente. Exemplos de funções implícitas discretas incluem as grades volumétricas, as quais apresentam as mesmas desvantagens que as representações discretas. Por outro lado, as representações implícitas analíticas são mais compactas do que as discretas, representam bem as superfícies suaves e podem ser avaliadas em resoluções arbitrárias.

2.2 Primitivas para especificação de objetos implícitos

A representação implícita é um esquema de representação construtiva e funcional, baseada em funções primitivas de classificação de pontos. De acordo com a maneira como estas funções são especificadas, primitivas podem ser agrupadas em quatro classes básicas: analíticas, procedurais, baseadas em amostras e baseadas em esqueletos *.

2.2.1 Analíticas

As primitivas analíticas são objetos implícitos definidos por uma função analítica. Essas incluem funções algébricas, como as quádricas, e funções não-algébricas, como as superquádricas.

A função geral é dada pela equação

$$f(x,y,z) = \sum_{i=0}^{l}\sum_{j=0}^{m}\sum_{k=0}^{n} A_{ijk}x^i y^j z^k. \tag{2.1}$$

O grau deste polinômio é o grau máximo dos termos não-nulos. Entre primitivas desta categoria temos:

- O **Plano** é a primitiva algébrica mais simples, dado por um polinômio afim

$$Ax + By + Cz + D = 0, \tag{2.2}$$

  onde (A, B, C) é um vetor ortogonal ao plano e D é a distância do plano à origem.

- As **Quádricas** são dadas por equações polinomiais de segundo grau

$$Ax^2 + By^2 + Cz^2 + 2Dxy + 2Exz + 2Fyz + 2Gx + 2Hy + 2Jz + K = 0. \tag{2.3}$$

  A combinação dos coeficientes da equação (2.3) resulta em superfícies quádricas com formas particulares. Entre as principais temos a esfera, o cilindro, o cone, o parabolóide e o hiperbolóide (veja a Figura 2.2).

- O **Toro** é um produto cartesiano de dois círculos de raio A e B. É definido por um polinômio de quarto grau

$$(x^2 + y^2 + z^2 - (A^2 + B^2))^2 - 4A^2(B^2 - z^2) = 0. \tag{2.4}$$

*O material das seções 2.2, 2.3 e 2.4 foi quase que inteiramente retirado de [52], que contém uma excelente introdução à modelagem e visualização de objetos implícitos.

Figura 2.2: Principais superfícies quádricas.

- As **Super-quádricas** não são algébricas, por serem definidas por equações polinomiais que envolvem potências fracionais. Elas generalizam as quádricas dando parâmetros de controle para manipular a forma destas primitivas. As superquádricas são uma classe de superfícies que possuem descrição implícita e paramétrica de forma natural. São exemplos deste tipo as superelipsóides, superhiperbolóides, supertoróides, etc.

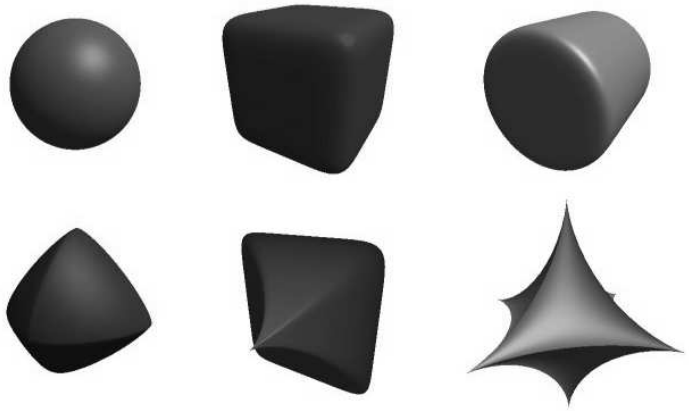


Figura 2.3: Alguns exemplos de superfícies superquádricas.

2.2.2 Procedurais

Primitivas procedurais são objetos implícitos cuja função característica f é definida algoritmicamente (isto é, por um programa). Devido ao fato de qualquer função poder ser gerada proceduralmente, uma primitiva é classificada como procedural apenas quando a sua descrição é inerentemente algorítmica.

- **Fractais** têm uma descrição algorítmica natural, podem ser especificados por um procedimento iterativo ou recursivo. O procedimento recursivo começa com uma aproximação do objeto, o qual é refinado pela alteração repetitiva das suas partes. O procedimento iterativo determina quando um ponto é parte do objeto baseado num

comportamento assintótico. Este último é o mais adequado para descrever fractais implicitamente. Um bom exemplo deste tipo é o "conjunto de Julia" (*The Julia Set*).

- **Hipertextura** é um esquema de modelagem procedural baseado em composição funcional. Os objetos são descritos por primitivas implícitas cuja função característica é controlada por uma aplicação sucessiva de "funções de forma" (*shaping functions*).

2.2.3 Baseadas em amostras

Primitivas baseadas em amostras discretas são objetos implícitos definidos por uma coleção de valores, $f_i = f(x_i)$ da função f, num conjunto discreto de posições $x_i, i = 1, ..., n$. Para a geração de uma função contínua a partir destas amostras usa-se um esquema de interpolação.

Estas primitivas podem ser classificadas com base em dois critérios: o padrão de amostragem e o esquema de interpolação usado para reconstruir f. O padrão de amostragem pode ser regular ou irregular. A reconstrução depende do tipo de padrão de amostragem e do esquema de interpolação.

Em alguns casos, este tipo de primitiva precisa de uma estrutura de suporte. O propósito de tal estrutura é duplo: organizar os dados por questões de armazenamento e definir as relações topológicas para a reconstrução da função f.

- **Amostragem irregular**. Corresponde à obtenção de amostras em posições quaisquer do espaço. Normalmente, requer que se realize algum tipo de estruturação antes de reconstruir a função f. Isto pode ser feito usando diversas técnicas:

  – *Interpolação Linear Baricêntrica*. Requer que as amostras sejam estruturadas numa triangulação tridimensional de Delaunay. Para cada tetraedro T, com vértices $\{v_1, v_2, v_3, v_4\}$, $v_i \in \Re^3$ e correspondentes valores $\{f_1, f_2, f_3, f_4\}$, realiza-se uma interpolação linear baricêntrica. Isto é,

  $$f(x)_{|T} = \sum_{i=1}^{4} \alpha_i f_i, \tag{2.5}$$

  onde α_1, α_2, α_3 e α_4 são coordenadas baricêntricas tais que $\sum_{i=1}^{4} \alpha_i = 1$.

  Devido à adjacência entre tetraedros, f resulta numa reconstrução linear por partes da superfície.

– *Interpolação do Fecho Simplicial* (*Interpolating simplicial hull*). Inicia-se com um poliedro que define a forma da superfície implícita. Em seguida é construída uma triangulação 3D deste poliedro. Para cada tetraedro do complexo simplicial, calcula-se um polinômio *Bézier-Bernstein* tal que o conjunto zero deste seja um termo da descrição polinomial por partes da superfície. Cada uma destas partes pode ser considerada como um retalho da superfície implícita.

– *Interpolação variacional*. É definida como um método de interpolação de dados desorganizados usando métodos variacionais.

Dado um conjunto de amostras, nas posições x_i e com valores correspondentes f_i, encontra-se uma função f que minimize o funcional de energia $E_f(x)$ e interpole $f(x_i) = f_i$. O funcional de energia pode medir a energia de primeira, segunda ou terceira ordem de f ou uma combinação delas. No capítulo seguinte entraremos em maiores detalhes sobre este método.

- **Amostragem regular**. As amostras são obtidas em posições que seguem uma estrutura regular, em geral, associados com os vértices de uma grade tridimensional.

  – *Arranjos de voxels* (*Voxel Arrays*). São coleções estruturadas de amostras. A estruturação refere-se ao posicionamento dos dados sobre os vértices de uma grade regular. Na maioria dos casos, esta grade é um arranjo retangular uniforme. Freqüentemente f é reconstruída a partir das amostras usando um esquema de interpolação trilinear. Para lidar com grandes quantidades de amostras podem ser empregadas técnicas de compressão de dados. Porém, para determinados tipos de dados – por exemplo amostras de campos de distância – é possível adotar estruturas baseadas em $octrees$, resultando num esquema mais econômico. Este tipo de representação é bastante comum em aplicações médicas.

  – *Malhas Volumétricas* (*Volume Meshes*). Utilizam como estrutura uma grade regular ou semi-regular. Esta grade é não-uniforme em geral e pode ser tetraedral ou hexaedral. A estrutura regular é adequada para métodos de reconstrução que usam funções polinomiais de ordem alta. No caso de grades hexaedrais pode-se usar polinômios *Bézier-Bernstein* como as funções base de f. Geralmente, esta representação é usada em Visualização Científica, onde

o conjunto de dados é gerado por simulações numéricas tais como análise de elementos finitos.

2.2.4 Baseadas em esqueletos

Primitivas baseadas em esqueletos são objetos implícitos definidos em termos de uma medida de distância a algum esqueleto [52]. São exemplos de esqueletos: pontos, curvas e superfícies. A fronteira de um objeto é uma superfície de nível afastada *c* unidades do esqueleto. Normalmente *c* é um parâmetro da primitiva, e sua função característica é expressa por $f(x, y, z) - c$.

- **Pontos**. O esqueleto mais simples é aquele formado por um conjunto isolado de pontos. Normalmente, a maior preocupação na construção destas primitivas está na criação de uma função distância flexível. As propriedades da função distância determinam o aspecto das primitivas e como elas se misturam. Nesta categoria estão caracterizados diversos modelos, tais como: *Blobby Models*, *Metaballs*, *Soft Objects*, etc.

- **Curvas**. Um esqueleto baseado em curvas é construído a partir de um segmento de curva em $\Re^3$. Esta curva define um cilindro generalizado com raio possivelmente variável. Exemplos deste tipo de primitivas são as retas e *splines*.

- **Superfícies**. Esta primitiva é definida como um deslocamento da superfície, restrita a posicionar-se a uma distância fixa e normal a outra superfície. A superfície esqueleto pode ser representada em forma paramétrica ou implícita. Na prática, a complexidade do cálculo da distância só permite a implementação das formas mais simples deste modelo. São exemplos deste tipo de primitiva os polígonos e campos de altura (*height fields*).

2.3 Construção de modelos 3D

Como indicamos no início deste capítulo, a especificação de modelos lida com as técnicas usadas para construir modelos de objetos e a interface de usuário do sistema, a qual deve suportar meios para criação, modificação e acesso à representação dos objetos.

Os objetos são **especificados** por meio de descrições de entrada que podem ser procedurais, interativas ou uma combinação de ambas. Esquemas procedurais são essencialmente linguagens de programação baseados em comandos de modelagem. Tais lin-

guagens podem incluir estruturas algorítmicas avançadas, tais como funções e cláusulas de controle de fluxo. Esquemas interativos são programas gráficos que apresentam ao usuário um conjunto de operações para desenho e modificação de objetos. Estes dois esquemas são complementares e podem ser implementados sob uma única interface de usuário, dando um mecanismo poderoso para especificação de modelos.

Outro importante aspecto do problema da **especificação** de modelos refere-se aos paradigmas de construção de modelos. As técnicas de modelagem constituem a metodologia básica para a criação de objetos. No contexto da modelagem de objetos implícitos, podemos agrupá-las em duas classes: construtivas e de "forma livre".

2.3.1 Técnicas construtivas

Técnicas de modelagem construtivas são usadas para construir objetos combinando partes básicas que vão formar uma estrutura composta.

Este método está estreitamente relacionado ao enfoque CSG (*Constructive Solid Geometry*), o qual usa um esquema de representação de sólidos através de operações booleanas ou combinações de objetos sólidos a partir de operações de conjuntos (união, interseção e diferença).

Entre outras técnicas do enfoque construtivo podemos mencionar, o instanciamento de primitivas e a combinação de objetos [4].

No instanciamento de primitivas criam-se novos objetos através do posicionamento de objetos por transformações geométricas (mudanças de escala, rotação, translação, etc.) ou pelo uso de primitivas parametrizáveis. No primeiro caso, Figura 2.4(a), uma cadeira pode ser modelada pela justaposição de paralelepípedos. No segundo caso, cria-se um conjunto de peças geralmente complexas e de uso comum para um determinado fim – por exemplo, engrenagens ou parafusos – e, a partir dessas primitivas, podemos criar uma infinidade de variações desses objetos com apenas alguns comandos para alteração de seus parâmetros (Figura 2.4(b)), como mudança de alturas e diâmetros.

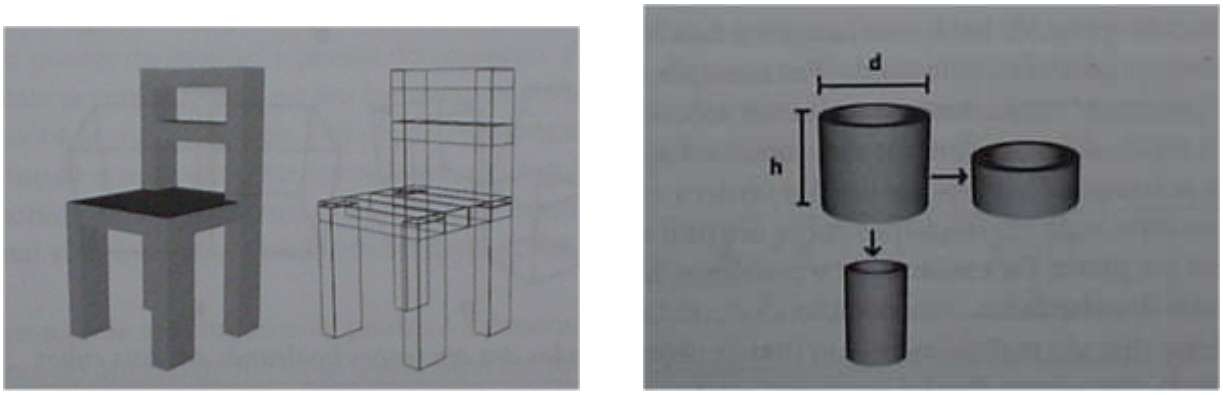


Figura 2.4: (a) Modelagem pela justaposição de paralelepípedos. (b) Alteração dos parâmetros de primitivas instanciadas [4].

Por outro lado, a habilidade de combinar objetos para criar outros é, sem dúvida, o método mais intuitivo e popular. Essa combinação é, na sua forma mais simples, feita por justaposição ou colagem de formas como mostrado na Figura 2.5. Outra forma simples de combinar objetos é fazendo uso de operações booleanas, isto é, união, interseção e diferença.

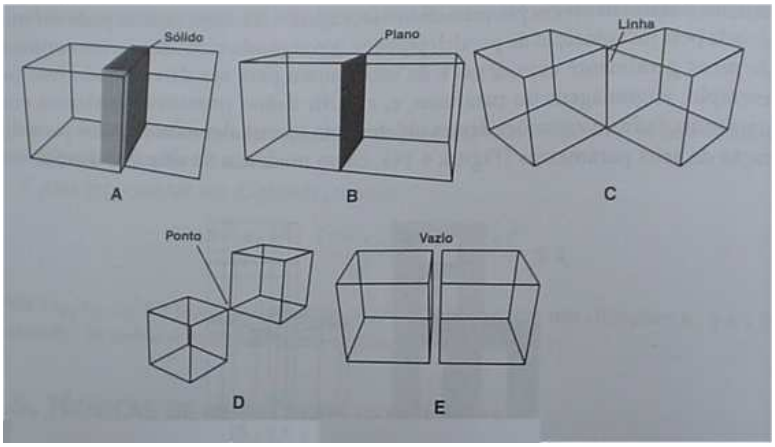


Figura 2.5: Resultados da aplicação de operações booleanas em dois cubos [4].

Finalmente, exemplos de sistemas implícitos baseados em técnicas construtivas são o sistema *F-Rep* [38, 39] e o sistema *Blob* [55]. Eles usam uma interface baseada em linguagem textual, assim como técnicas de modelagem interativa. *F-Rep* trabalha com *R-functions* e tem sua própria linguagem. O sistema *Blob* usa a linguagem de extensão *Python*.

2.3.2 Técnicas de “forma livre”

Uma maneira efetiva de desenvolver técnicas de modelagem de “forma livre”, é através do uso de primitivas implícitas baseadas em amostras, onde, as amostras dão o controle necessário para modelar a superfície do objeto desejado. Basicamente, o método recai no contexto do problema de interpolação de dados não-estruturados, isto é, onde os dados não requerem nenhum tipo de estruturação.

Usa-se o termo “forma livre” já que não é necessário conhecimento prévio da forma da superfície modelada. Esta técnica é empregada em diversos trabalhos [12, 51, 56, 19, 25], e serviu como ponto de partida para a presente dissertação.

Outro método é especificar o objeto implícito através de um modelo parametrizado, onde os parâmetros controlam o modelo localmente e de um modo diferencial. Witkin and Heckert [54] apresentam um método, baseado em partículas, para a amostragem e controle de superfícies implícitas.

2.4 Visualização de objetos implícitos

Todos os sistemas que usam objetos implícitos precisam realizar sua visualização. As principais técnicas de visualização deste tipo de objetos podem ser classificadas segundo a primitiva de desenho empregada [52]:

- **Pontos**. Os objetos implícitos podem ser exibidos como uma nuvem de partículas (pontos) distribuídos na sua superfície. O processo de espalhamento das partículas sobre a superfície segue em geral um modelo baseado na física. Forças de atração são impostas para manter as partículas sobre a superfície, enquanto que forças de repulsão são usadas para mantê-las com um determinado afastamento umas das outras, garantindo assim uma distribuição mais uniforme. As forças de atração usualmente são calculadas em função do gradiente e do sinal da função.

  O uso de sistema de partículas evita que procedimentos adicionais que garantam consistência topológica tenham que ser aplicados. Uma vez que as partículas não aparecem conectadas explicitamente, a interpretação da topologia fica por conta do usuário. Isto torna a visualização mais rápida, mas também pode trazer problemas quanto à interpretação visual dos resultados por parte do usuário. A Figura 2.6 ilustra esta técnica.

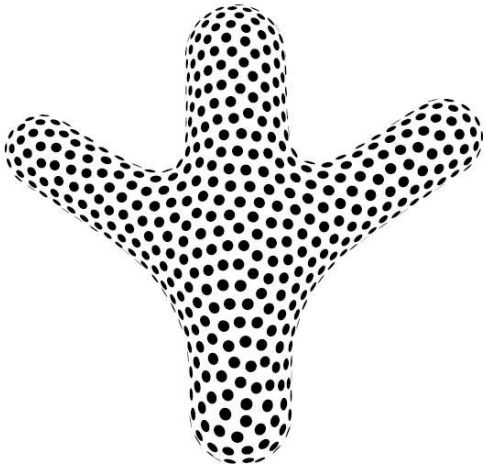
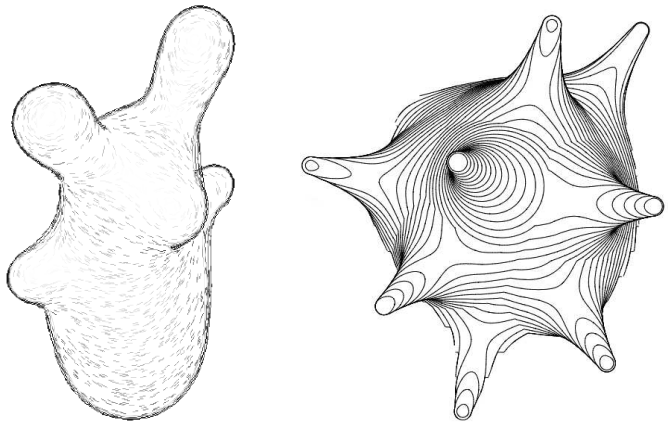


Figura 2.6: Visualização por sistemas de partículas [5].

- **Curvas**. Uma técnica efetiva para visualizar objetos implícitos é o uso de curvas. Se estas são cuidadosamente selecionadas, as imagens resultantes capturam as principais características dos objetos com poucos traços. O problema é como determinar quais curvas podem descrever melhor os aspectos geométricos mais importantes de um objeto. Duas classes de traços podem ser identificadas:

  – Curvas de silhueta: Um ponto p de silhueta numa superfície S é um ponto $p \in S$ tal que a linha entre o olho do observador e p seja tangente a S. Uma curva de silhueta em M é uma curva constituída por pontos de silhueta. As curvas de silhueta mais comuns delimitam a borda entre o objeto e o fundo. Elas são definidas como um conjunto de curvas em que o produto interno entre a normal à superfície e o vetor na direção da visão é igual a zero ou a normal à superfície é descontínua (Figura 2.7(a)).

  – Curvas de nível: São obtidas como produto da intersecção do objeto com uma série de planos perpendiculares à linha de visão e que dele se afastam, da frente para trás. Estas linhas dão uma representação geométrica sem ambigüidade e estão entre as mais úteis para a visualização destes tipos de modelos (Figura 2.7(b)).



(a) (b)

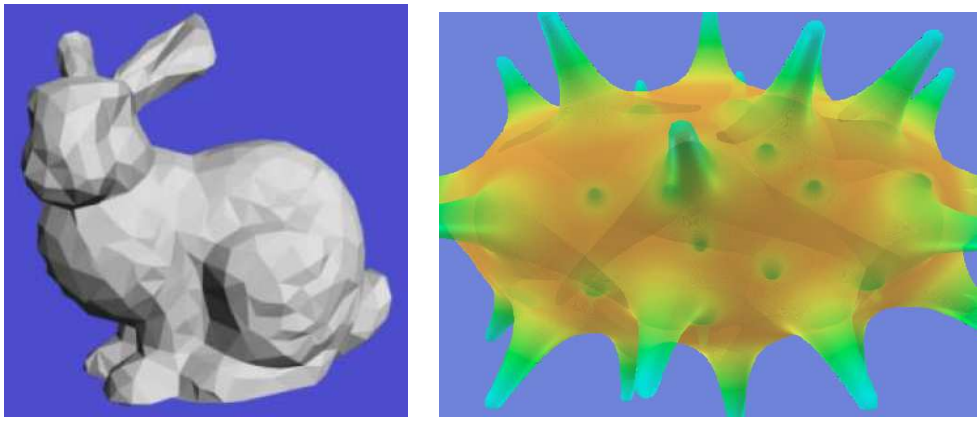
Figura 2.7: Visualização de objetos implícitos usando curvas. (a) Visualização por curvas de silhueta. (b) Visualização por curvas de contorno [5].

- **Superfícies**. Uma renderização realística de um modelo de objeto definido implicitamente é alcançado por meio da visualização da sua superfície juntamente com um modelo de iluminação.

  – Renderização por polígonos: Uma aproximação linear por partes da borda de um objeto implícito, junto com algoritmos de renderização por polígonos podem ser usados para visualizar a superfície do objeto. Este método possibilita incorporar os objetos implícitos em sistemas baseados em polígonos permitindo tirar vantagem de hardware gráfico especializado. Maiores detalhes desta técnica são abordados na subseção 2.4.1.

  – Traçado de raios (*Ray Tracing*): É um método tradicional usado para renderizar objetos implícitos. Em essência, esta classe de algoritmos promove o acompanhamento de um raio de luz no sentido inverso. Para cada pixel da imagem a ser gerada, um raio (ou mais, no caso de super-amostragem) é lançado. A cada interseção desse raio com um objeto da cena a ser renderizada, um cálculo de iluminação direta é feito e um (ou mais, no caso de objetos translúcidos, por exemplo) novo raio é lançado. Esse processo de acompanhamento termina quando um nível limite de reflexões é alcançado, ou é detectado que o raio possui pouca intensidade, ou que o mesmo não intersecta nenhum outro objeto.

    Apesar de todo o esforço no sentido de diminuir o seu custo computacional, os algoritmos de *ray tracing* não são rápidos o bastante para a visualização em tempo real.



(a) (b)

Figura 2.8: Visualização da superfície de objetos implícitos. (a) Modelo poligonal iluminado [50]. (b) Traçado de raios (*Ray Tracing*) [5].

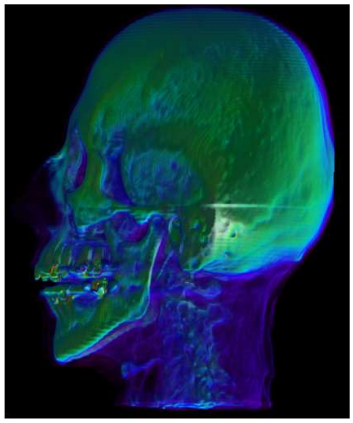


Figura 2.9: Visualização Volumétrica, lançamento de raios (*Ray casting*).

- **Volumes**. Existem duas situações em que as técnicas de visualização descritas anteriormente não são adequadas. A primeira acontece quando o objeto implícito é tão complexo que a noção da superfície se acaba. A segunda ocorre quando é necessário visualizar alguma propriedade espacial da função implícita. Em ambos casos, a solução é usar técnicas de visualização volumétricas. Os dois métodos principais para fazer rendering volumétrico são métodos de projeção e lançamento de raios (*Ray casting*).

  – Métodos de projeção: Neste método o volume é decomposto num arranjo tridimensional de voxels. Cada voxel é projetado no plano da imagem e sua contribuição à imagem é calculada.

  – Lançamento de raios: Raios do ponto de visão através de cada pixel são lançados no volume. A contribuição ao pixel é calculada integrando a função de

densidade ao longo do raio (Figura 2.9).

2.4.1 Poligonização

Para capturar a geometria de um objeto implícito é necessário amostrar a função implícita f que a define. Como estes objetos estão definidos indiretamente por f, não há uma maneira direta de gerar o conjunto de pontos f^{-1} que o definem. Por esta razão, freqüentemente, é necessário recorrer a aproximações.

Em geral, uma aproximação é uma decomposição de células que produz uma parametrização por partes de um objeto implícito. A ordem da aproximação é determinada pela geometria de cada célula.

Métodos de aproximações lineares para objetos implícitos geram uma decomposição do domínio da função implícita, assim como também sua parametrização linear por partes associada. Como esta última é uma aproximação poligonal da borda do objeto, é conhecida também como um método de poligonização.

Métodos de poligonização podem ser classificados de acordo com o tipo de decomposição usado. Podem ser extrínsecos, quando o domínio espacial da função implícita é subdividido, ou intrínsecos, quando o objeto implícito é subdividido diretamente. Outros critérios de classificação podem ser a estratégia de amostragem e o método de estruturação.

Os métodos de poligonização extrínsecos podem ser subclassificados de acordo com [52]:

- A classe de complexo celular usado na decomposição do espaço.
  - Métodos não-simpliciais: Triangulam o domínio de f construindo um complexo celular.

    O método de poligonização não-simplicial mais conhecido é representado pelo algoritmo *Marching Cubes* [29]. Este subdivide o espaço regularmente em células cúbicas. O método consiste de três etapas principais (Figura 2.10). O complexo celular é examinado para identificar os cubos transversos; a topologia dos polígonos em cada cubo é determinada por meio de uma tabela de conectividade de vértices; e as posições dos vértices são calculadas por interpolação linear.

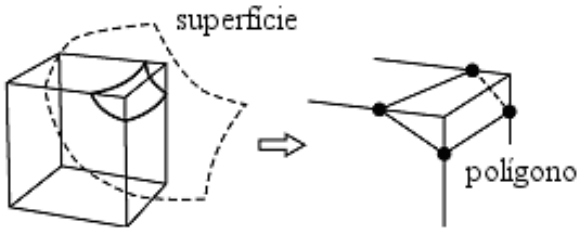


Figura 2.10: Poligonização.

Estes métodos são rápidos e simples de implementar. A principal desvantagem é que eles não são robustos. Por exemplo, veja na Figura 2.11 um caso de ambigüidade onde a intersecção entre uma célula cúbica e a iso-superfície não pode ser determinada de uma única forma.

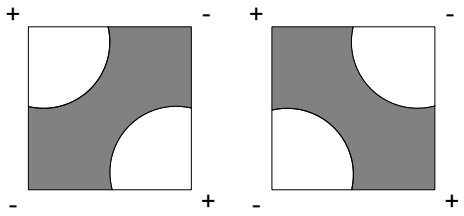


Figura 2.11: Ambigüidade na poligonização.

  - Métodos simpliciais: Triangulam o domínio da função implícita formando um complexo simplicial. A aproximação simplicial da superfície implícita é obtida calculando o núcleo de $\tilde{f}$ sobre cada simplexo. Isto requer a solução de um sistema de equações [1].

    A função $\tilde{f}$ é uma aproximação afim de f, definida em cada simplexo por uma interpolação linear em coordenadas baricêntricas.

- A estratégia de *tracking* adotada para visitar as células intersectantes.

  - Métodos baseados em continuação: Começam com uma célula semente que sabidamente intersecta a superfície. Procura-se nas suas células vizinhas imediatas para determinar qual delas também intersecta. Este processo é repetido até que todas as células transversas sejam encontradas.
  - Métodos *Full-scan*: Visitam e classificam todos os elementos do complexo simplicial para descobrir quais células intersectam a superfície.

- O tipo de subdivisão usada.

– Métodos de subdivisão uniforme: Subdividem o espaço em intervalos regulares gerando uma decomposição de resolução fixa.

– Métodos adaptativos: Os métodos adaptativos criam uma decomposição espacial que é sensível a alguma característica do objeto. Eles começam com uma subdivisão inicial do domínio do objeto e o refinam recursivamente até que um critério de adaptação seja encontrado. A Figura 2.12 apresenta o resultado de um processo de poligonização adaptativa.

  Há dois métodos básicos para a adaptabilidade. A principal diferença está na maneira que eles restringem a subdivisão. Um restringe as células da decomposição e o outro as arestas da poligonização.

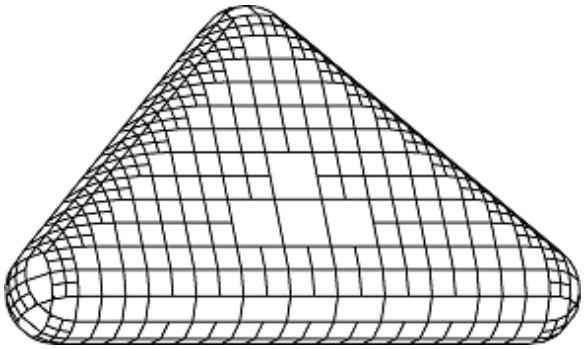


Figura 2.12: Poligonização adaptativa [5].

2.5 Interfaces de usuário baseadas em traços

A queda de preços dos computadores – além de características tais como redução das suas dimensões, alta potencialidade e ubiqüidade – está permitindo explorar novos tipos de aplicações; com uma tendência a dar suporte a cada aspecto da vida humana sob um estilo de interação mais "natural". Entre estes paradigmas de interação homem-computador, pós-WIMP, temos a realidade virtual, realidade aumentada, interfaces multi-modais e multimídia, interfaces de linguagem natural, interfaces de reconhecimento de som, fala, traços, etc.

No contexto da modelagem geométrica, interfaces baseadas em traços estão sendo usadas para expressar idéias visuais em forma de *strokes*, isto é, traços simples especificados pelo movimento do ponteiro do mouse, ou da caneta óptica, sobre a tela computador, o qual gera ações apropriadas analisando as características perceptivas dos traços.

A natureza fluente e leve deste tipo de interface a fazem adequada para atividades de desenho criativo e exploratório. Com elas pode-se expressar idéias visuais nos computa-

dores sem fazer uso de seqüências de comandos cansativos e sem ter de lidar com ícones, menus e ferramentas de seleção. A sua principal característica – a manipulação direta –, tenta fazer do desenho uma atividade sem restrições artificiais.

Metin et al. [32], definem algumas propriedades básicas para os sistemas baseados em traços:

- Devem permitir desenhar formas arbitrárias com um simples traço, sem requerer que o usuário desenhe objetos em peças.
- Os sistemas não devem ter "modos" de desenho para a especificação de diferentes formas geométricas ou de comandos (modos para criar, cortar, combinar, etc.).
- Devem ser naturais ao usuário.

"A meta é fazer que os computadores entendam o que o usuário está desenhando em lugar de exigir do usuário que desenhe de maneira que o computador entenda" [32].

Por outro lado, Igarashi et.al. [20] chamam este tipo de interface como *Freeform User Interfaces* e a caracteriza com as seguintes propriedades básicas:

- Entradas baseadas em traços: Internamente, os traços estão representados por uma seqüência de pontos. Durante uma operação de traçado, a trajetória do movimento do traço é mostrada na tela, e o sistema responde ao evento quando o usuário interrompe o traçado levantando a caneta. A reação do sistema está baseada na trajetória completa do movimento da caneta durante o traçado. Em contraste, na clássica operação de "arrastar" a resposta do sistema se baseia na posição final, e possivelmente na inicial, do cursor. A Figura 2.13 ilustra esta diferença. O traçado é uma maneira rápida, intuitiva e eficiente de expressar idéias gráficas num ambiente computadorizado. É especialmente adequada para desenhar modelos grosseiros à mão livre, o que é uma atividade quase natural para as pessoas.
- Processamento perceptual: Trata-se de um processamento avançado de traços irrestritos inspirado pela percepção humana. O processamento perceptual permite ao usuário realizar complicadas tarefas interagindo diretamente na cena, não requerendo que a tarefa seja decomposta numa seqüência de comandos complicados.
- Apresentação informal: Esta última propriedade estabelece que o sistema apresente o processo de desenho ou o resultado final com uma aparência informal, por exemplo usando renderização não fotograficamente realista (NPR-*Non-Photo Realistic*

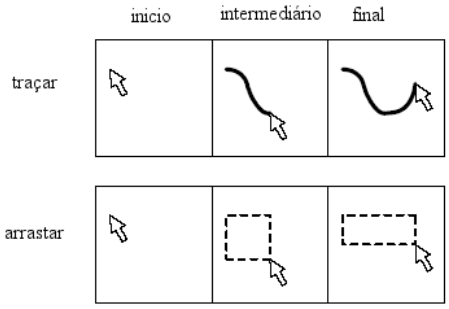


Figura 2.13: Diferença entre o traçado e a clássica operação de arrastar.

Rendering). Esta apresentação informal é importante para despertar uma apropriada expectativa do usuário acerca da funcionalidade do sistema. Se o sistema oferece cenas precisas e detalhadas, o usuário, naturalmente, esperará que os resultados da modelagem sejam precisos e detalhados. Por outro lado, se o sistema apresenta objetos de maneira informal, o usuário pode se concentrar na estrutura geral da cena sem se preocupar muito com os detalhes.

Mesmo conseguindo uma interação fluente, que não é possível com as interfaces estilo WIMP, as interfaces baseadas em traços têm uma grande limitação: a "ambigüidade". É difícil construir aplicações que reconheçam diagramas, desenhos e traços. O principal obstáculo é que estes tipos de entradas são altamente variáveis em muitos aspectos. Por exemplo, um traço fechado pode ser interpretado como a curva base para uma operação de extrusão ou como uma silhueta inicial para uma operação de criação.

Estes problemas de ambigüidade podem confundir o usuário, causar problemas de performance e resultar num estilo de interação não-fluido e cansativo, criando uma barreira para a aceitação e a aplicabilidade deste tipo de interações.

Assim, uma vez que a ambigüidade seja descoberta, ela deve ser corrigida. Este processo de correção é chamado de mediação, e geralmente envolve algum diálogo entre o sistema e o usuário para identificar corretamente a entrada pretendida pelo usuário. Na verdade, existe uma variedade de estratégias de mediação [32]:

- Repetição: O usuário repete sua entrada até que o sistema a interprete corretamente.
- Eleição: O sistema exibe diferentes alternativas e o usuário seleciona a resposta correta entre elas.
- Mediação automática: Esta estratégia consiste em eleger uma interpretação sem envolver completamente o usuário.

Capítulo 3

Funções de base radiais

3.1 Introdução

A aproximação de dados consiste no processo de estimar valores de uma função em posições arbitrárias a partir de um conjunto de dados, com valores e posições conhecidas. Nas posições originais, os valores estimados pela função não precisam ser iguais aos valores conhecidos. Isto distingue aproximação de interpolação.

Existem diferentes métodos de aproximação de dados multidimensionais. Entres os principais temos: os métodos polinomiais, os baseados em *splines*, os baseados em produto tensorial, métodos locais, globais, etc.

Porém, quando os dados estão arbitrariamente posicionados, isto é, sem seguir nenhuma estrutura regular, o método escolhido, com freqüência, é o baseado em funções de base radiais (RBFs-*Radial Basis Functions*). Esta escolha é feita devido às suas excelentes propriedades de aproximação e à sua aplicabilidade independente da dimensão à qual pertencem os dados.

Entre algumas aplicações práticas, nas quais as RBFs têm demonstrado grande utilidade, temos [10]:

- Mapeamento de imagens para fins de comparação.
- Interpolação de dados multidimensionais ou distribuições escalares, tais como medidas de densidade, temperatura, potencial elétrico, etc.
- Aproximação de variáveis de aprendizagem na área de redes neurais.
- Solução numérica de equações diferenciais parciais.

3.1.1 Funções de base radiais

Definição.- Seja um conjunto de dados espalhados nas posições $x_i \in \Re^n$, junto com correspondentes valores $v_i \in \Re$, encontrar o aproximante, $s : \Re^n \mapsto \Re$, tal que s aproxime suavemente o conjunto de dados, sendo que os v_i representam as avaliações de $f : \Re^n \mapsto \Re$ em x_i, isto é, $f(x_i) = v_i$, para $i = 1...n$.

Desta forma, dado um espaço linear de aproximantes S, usualmente de dimensão finita, há vários métodos de encontrar aproximantes $s \in S$ para uma função f. Entre estes, estão os métodos de aproximação por mínimos quadrados ou os métodos de quase-interpolação. Porém, nesta dissertação, a aproximação é dada por "interpolação", isto é, requeremos explicitamente que $s(x_i) = f(x_i)$, $i = 1...n$. Este processo de interpolação é feito sem nenhuma assunção acerca do conjunto de dados de entrada, o que significa que os dados estão posicionados sem obedecer a nenhum critério de regularidade.

Assim, a solução do problema de interpolação, baseada em RBFs, é uma função $f : \Re^n \mapsto \Re$, que consiste da combinação linear de translações de uma função base $\phi : \Re_+ \mapsto \Re$, centralizada nas posições x_i. Onde ϕ é radialmente simétrica –o que significa, que o valor da função só depende da distância Euclidiana entre o argumento e o centro da função– e invariante frente a rotações ou translações.

Desta forma, f tem a seguinte estrutura geral:

$$f(x) = \sum_{k=1}^{n} \lambda_k \phi(\|x - x_k\|), \quad x \in \Re^n, \ \lambda \in \Re. \tag{3.1}$$

Então, para satisfazer as condições de interpolação $f(x_i) = v_i$, temos:

$$\sum_{k=1}^{n} \lambda_k \phi(\|x - x_k\|) = v_i, \quad i = 1, ..., n, \tag{3.2}$$

que pode ser representado por um sistema de equações lineares:

$$A\lambda = b. \tag{3.3}$$

Onde, para $i = j = 1...n$,

$$A = \{\phi_{ij}\}, \quad \phi_{ij} = \phi(\|x_i - x_j\|),$$

$$\lambda = \{\lambda_i\},$$

$$b = \{v_i\}.$$

Diferentes funções radiais ϕ podem ser usadas na solução deste problema. Dependendo da escolha, a matriz A pode ser definida positiva, ou definida condicionalmente positiva de alguma ordem m. Para este último caso precisa-se adicionar um termo polinomial $p(x) \in P^{h-1}$ na equação (3.1), com o fim de garantir a unicidade da solução, onde P^{h-1} é o espaço de polinômios de grau $h-1$. Em conseqüência, a função interpolante terá a seguinte forma:

$$f(x) = \sum_{k=1}^{n} \lambda_k \phi(\|x - x_k\|) + p(x). \tag{3.4}$$

Neste caso, além de satisfazer as condições de interpolação (equação (3.2)), o conjunto $\{\lambda_k\}$ tem que ser ortogonal ao espaço polinomial P^{h-1}; portanto:

$$\sum_{l=1}^{n} \lambda_k q(x) = 0, \quad \forall q \in P^{h-1}. \tag{3.5}$$

Entre as diferentes funções radiais que têm sido usadas na solução deste problema, e considerando $\phi(r_k)$ centralizada em x_k, temos:

Multiquádricas de Hardy	$\phi(r_k) = \sqrt{r_k^2 + h^2}$
Multiquádricas inversas	$\phi(r_k) = \frac{1}{\sqrt{r_k^2+h^2}}$
Thin-plate spline	$\phi(r_k) = r_k^2 \log r_k$
Linear	$\phi(r_k) = r_k$
Cúbica	$\phi(r_k) = r_k^3$
Gaussiana	$\phi(r_k) = e^{h^2 r_k^2}$
Shifted Thin-plate	$\phi(r_k) = (r_k^2 + h^2)\log(r_k^2 + h_2)^{1/2}$

Tabela 3.1: Principais funções radiais usadas na solução do problema de interpolação.

Existem diversas variações do problema e, portanto, diferentes soluções para elas. Para uma revisão mais detalhada, demonstrações da existência dos interpolantes e análise de convergência do método, veja [28, 10].

3.1.2 RBFs de suporte compacto

Para resolver o sistema de equações lineares da equação (3.3), métodos diretos ou iterativos simples têm complexidade $O(n^2)$ para armazenamento e $O(n^3)$ para a execução das operações. Para problemas com grandes quantidades de dados esta complexidade representa um altíssimo custo computacional.

Além disso, para as funções radiais da tabela 3.1, exceto para as gaussianas e multiquádricas inversas, a matriz resultante pode ser afetada por problemas de mau condicionamento. Isto se deve à interpolação de grandes quantidades de dados e às entradas na matriz crescerem em valor, conforme se afastam da diagonal.

Estes problemas de instabilidade e alto custo computacional associados à construção da função interpolante têm sugerido pesquisas para obter funções que decaiam com a distância, as quais são conhecidas como funções de base radiais de suporte compacto (CSRBFs - *Compact Support Radial Basis Functions*).

Geralmente, estas funções têm a seguinte forma [53]:

$$\phi(r_k) = \begin{cases} (1-r_k)^p + P(r_k) & \text{se } r_k < 1 \\ 0 & \text{em outro caso} \end{cases} \tag{3.6}$$

Para vários graus de continuidade do interpolante, Wendland [53] derivou as seguintes CSRBFs:

$(1-r_k)^2$	C^0
$(1-r_k)^4 + (4r_k + 1)$	C^2
$(1-r_k)^6 + (35r_k^2 + 18r_k + 3)$	C^4
$(1-r_k)^8 + (32r_k^3 + 25r_k^2 + 8r_k + 1)$	C^6

Tabela 3.2: CSRBFs e sua respectiva ordem de suavidade da função interpolante.

Outro tipo de CSRBFs são as apresentadas por Buhmann [11]. As funções de Buhmann contêm, diferentemente das anteriores, um termo logarítmico e são similares às *thin-plate splines*, porém truncadas adequadamente. Alguns exemplos destas funções são:

$\phi(r_k) = 2r_k^4 \log r_k - \frac{7}{2}r_k^4 + \frac{16}{3}r_k^3 - 2r_k^2 + \frac{1}{6}$	$0 \leq r_k \leq 1$
$\phi(r_k) = \frac{1}{2}r_k^4 + \frac{4}{9}r_k^3 - \frac{4}{3}r_k^3 \log r_k - r_k^2 + \frac{1}{18}$	$0 \leq r_k \leq 1$

Tabela 3.3: CSRBFs derivadas por Buhmann [11].

Fazendo uso de CSRBFs, o sistema de equações lineares (Equação (3.3)) muda para um sistema linear esparso. As entradas na matriz serão zero para pontos com distâncias maiores que o raio de suporte. Os sistemas lineares esparsos podem ser eficientemente resolvidos por métodos diretos ou iterativos.

Apesar do procedimento de cálculo acelerado que as CSRBFs oferecem, elas têm alta sensibilidade à densidade dos dados a serem interpolados. Em contraste, as funções de suporte global podem facilmente lidar com este problema e, normalmente, oferecem melhores resultados.

Entretanto, a complexidade $O(n^3)$ para os métodos diretos tem originado uma intensiva pesquisa a fim de reduzir este alto custo computacional. Entre esses métodos podemos mencionar os métodos iterativos para funções de base radiais de suporte global, os quais são responsáveis pela solução do sistema de equações e pela rápida avaliação da função interpolante.

3.1.3 Métodos iterativos

Entre os principais métodos iterativos desenvolvidos para a interpolação de dados usando funções base de suporte global, temos (veja [10] para maiores detalhes desta classificação):

- O método BFGP(*Beatson-Faul-Goodsell-Powell*): As diferentes variantes deste método se baseiam no fato de que, apesar do suporte global das RBFs, estas atuam localmente. Portanto, pode-se assumir que cada coeficiente λ_i do interpolante (veja equação (3.3)) depende exclusivamente dos centros mais próximos ao centro correspondente. Após a aplicação de um método de decomposição espacial, obtém-se um conjunto de células não disjuntas, nas quais os coeficientes λ_i são aproximados com funções locais de *Lagrange*, aproveitando o comportamento "local" das funções radiais. Finalmente, o interpolante é aproximado iterativamente por uma combinação linear de tais funções locais de *Lagrange* até alcançar uma precisão desejada.

- O método *Fast Multipole*: Este método está baseado no fato de que RBFs podem ser expandidas em séries infinitas de *Laurent*. Com base numa subdivisão hierárquica do espaço e com células disjuntas, determinam-se aqueles centros que estão em células "distantes" e os que estão em células "próximas" de um ponto x. Desta forma, avalia-se f em x, calculando o valor exato dos termos do somatório cujos centros estão perto de x. Para os centros que estão longe estima-se o valor de seus termos usando uma aproximação de *Taylor*.

- O método de pré-condicionamento: Pré-condicionamento é uma técnica de aceleração de convergência usada em métodos iterativos. Este método consiste no pré-

condicionamento direto da matriz A, usualmente mal condicionada, com uma simples matriz de pré-condicionamento P. Desta forma, métodos iterativos padrões, tais como os do *Gauss-Seidel* ou gradientes conjugados, podem ser aplicados para resolver o sistema linear da equação (3.3).

3.2 RBFs na modelagem e visualização de dados 3D

O problema de interpolação de dados 3D consiste em modelar distribuições escalares, onde um atributo escalar v_i é descrito, normalmente, em função de uma posição x_i, onde $x \in \Re^3$ para $i = 1...n$. Com fins de analisar ou visualizar as relações implicadas pelos dados, constrói-se uma função interpolante f tal que $f(x_i) = v_i$.

Este tipo de dados surge em numerosas áreas de aplicação [34]:

- Medidas de temperatura em vários pontos de uma fornalha.
- Medidas de densidade em diferentes posições do corpo humano.
- Concentrações de minerais em diversas profundidades obtidas por perfuração de poços.
- Níveis de performance econômica, taxas de interesse e níveis de desemprego.

O uso de funções interpolantes baseadas em RBFs não exige nenhuma estruturação especial no posicionamento dos dados, permitindo trabalhar com grandes quantidades destes e resultando numa função, f, suave, contínua e que pode ser avaliada em qualquer lugar do espaço. Esta característica facilita enormemente a visualização de dados, não-estruturados, já que a função pode ser amostrada num plano para seccionamento 2D ou avaliada numa grade regular para realizar rendering volumétrico, ou avaliada com o fim de extrair iso-superfícies.

Na Figura 3.1 apresentam-se diferentes métodos de visualização de dados geofísicos 3D, aproveitando uma das principais vantagens das RBFs, isto é, avaliação da função em qualquer posição do espaço.

Algumas vezes não é desejável realizar a interpolação exata de dados; isto acontece, normalmente, quando os dados estão contaminados com ruído. Nestes casos, uma "aproximação" é mais apropriada. No contexto de aproximação baseada em RBFs, dois métodos têm demonstrado excelentes resultados: ajuste por suavização de *spline* (*spline smoothing fitting*) e ajuste por intervalo de erro (*error bar fitting*).

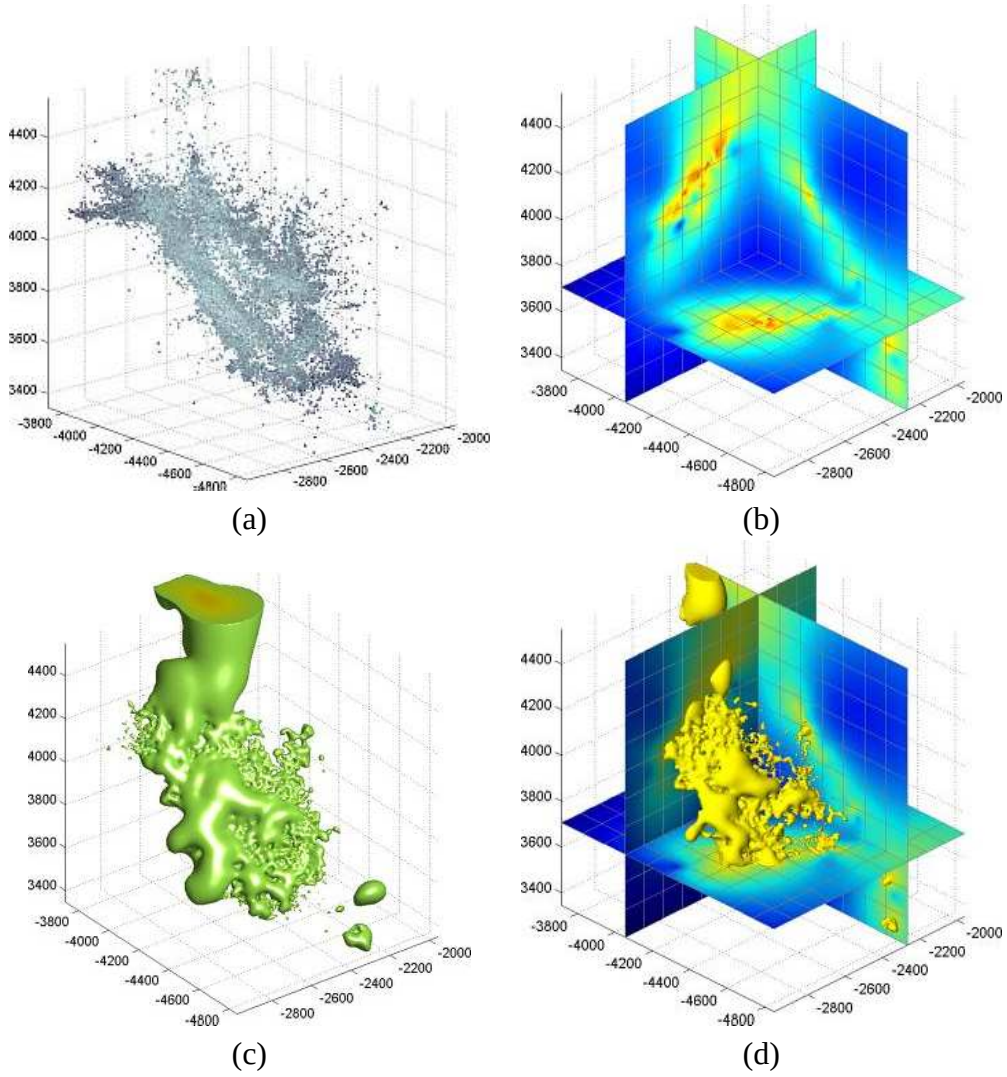


Figura 3.1: Diferentes formas de visualização de dados [44]. (a) Nuvem de pontos original em 3D, a qual será interpolada. (b) *Multi-planar reslicing*. (c) Iso-superfície. (d) Uma vista combinada.

No ajuste por intervalo de erro, tenta-se encontrar a função mais suave que respeita os intervalos de erro especificados nos pontos, enquanto que o ajuste por suavização de *spline* procura equilibrar a suavidade da RBF com a fidelidade aos dados brutos.

Por outro lado, mesmo que os dados não estejam afetados por ruído, a suavização pode ser requerida para remover detalhes não desejáveis ou para evitar *aliasing* quando avaliamos uma RBF numa malha ou numa grade, que é relativamente grosseira ao detalhe apresentado na RBF. Desta forma, podemos suavizar uma RBF, a qual interpolou exatamente os dados, aplicando um filtro passa-baixa (LPF) durante a avaliação do interpolante. O LPF pode ser implementado simplesmente pela troca da função base no momento da avaliação ou pela aproximação discreta de núcleos (kernels) de suavização

[13]. Na Figura 3.2 podemos observar resultados de diferentes técnicas de suavização de dados.

Interpolação original

(a) (b) (c)

Figura 3.2: Resultados de diferentes técnicas de suavização de dados [44]. (a) Aplicando filtros passa-baixa. (b) Ajuste por intervalo de erro. (c) Ajuste por suavização de *spline*.

3.3 RBFs na modelagem de superfícies

Nesta seção abordamos o problema de modelar superfícies implicitamente como o conjunto zero de uma função f. Basicamente, desejamos encontrar uma superfície que interpole um conjunto finito de n pontos $x_i \in \Re^3$, para $i = 1...n$. Problemas deste tipo surgem em diversas áreas de aplicação, tais como modelagem geométrica e visualização científica.

De forma geral, as principais soluções para este problema podem ser classificadas em cinco categorias [28]:

- Soluções Paramétricas: São uma das representações mais populares na academia e na indústria para resolver este tipo de problemas. A solução polinomial é representada por equações do tipo $x = x(u, v)$, $y = y(u, v)$ e $z = z(u, v)$. Portanto, cada

coordenada de um ponto na superfície é representada como função dos parâmetros u e v, sendo que a posição do ponto na superfície é fixada pelos valores dos parâmetros. Uma generalização é considerar soluções polinomiais racionais que possam ser expressas na forma $x = \frac{x(u,v)}{w(u,v)}$, $y = \frac{y(u,v)}{w(u,v)}$ e $z = \frac{z(u,v)}{w(u,v)}$, onde $w(u, v)$ também é uma função polinomial.

- Soluções Algébricas: Neste tipo de solução, a superfície é representada como o conjunto zero de alguma função f em $\Re^3$. Além disso, a função f é definida como um polinômio; o grau deste é conhecido como o grau algébrico da solução.

- Soluções baseadas em RBFs: Uma das principais vantagens desta técnica, em relação às anteriores, é que não requer nenhuma organização especial dos dados de entrada, nem informação acerca da sua conectividade. A solução do problema consiste na construção de uma função interpolante, que define implicitamente a superfície, e está formada por uma combinação linear de translações de alguma função de base radial escolhida.

- Soluções baseadas no método de *Shepard*: Este método e suas variantes constroem as funções interpolantes com base numa combinação de um conjunto de soluções locais, onde, para cada função local, é calculada uma função de peso. Em [35], observamos como este método é usado para resolver problemas de reconstrução de superfícies.

- Superfícies de Subdivisão: Em modelagem geométrica, estas técnicas usam uma malha de suporte a qual é prescrita com os dados. O processo consiste, basicamente, em gerar um modelo cada vez mais suave a partir de um modelo inicial grosseiro.

Modelagem implícita de superfícies usando RBFs é uma técnica relativamente recente em computação gráfica. Oferece a capacidade de interpolação de buracos grandes e irregulares em superfícies incompletas, sem restringir a topologia do objeto e sem precisar de conhecimento prévio da forma do objeto. Além disso, ela nos proporciona as seguintes vantagens:

- Representação compacta com só uma função analítica.

- As iso-superfícies extraídas são variedades (i.e. não têm auto-interseções).

- Pode interpolar dados esparsos e não uniformemente espaçados.

- Pode interpolar e/ou aproximar dados.
- As funções obtidas podem ser avaliadas em qualquer lugar do espaço e assim gerar malhas de qualquer resolução desejada.
- Os gradientes e derivadas superiores podem ser calculados analiticamente.
- Vetores normais à superfície são facilmente computados.

A Figura 3.3 ilustra algumas aplicações da modelagem de superfícies baseada em RBFs.

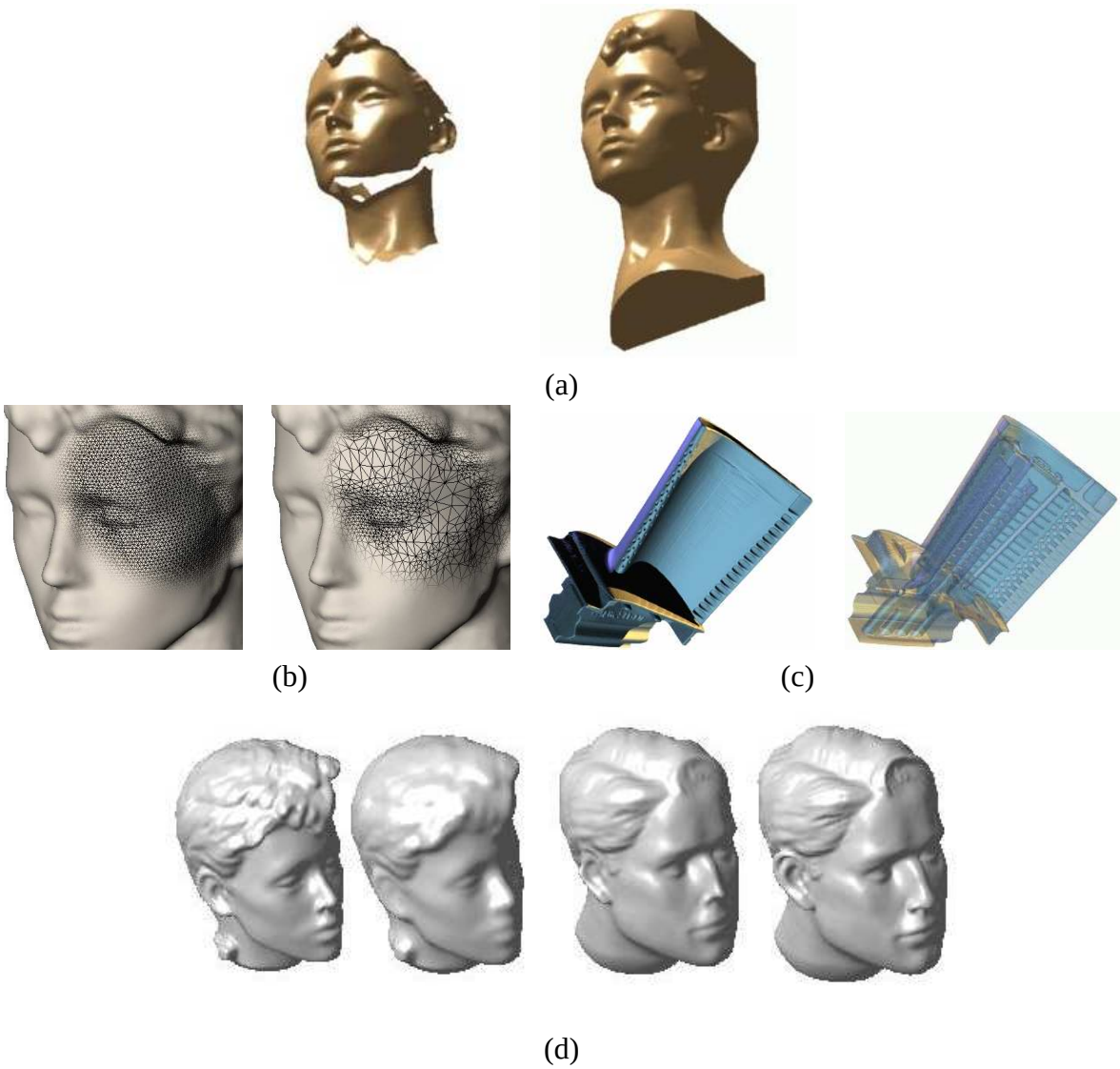


Figura 3.3: Diferentes aplicações de modelagem de superfícies baseada em RBFs [44]. (a) Reparação automática de buracos no modelo. (b) Simplificação de Malhas. (c) Visualização volumétrica. (d) *Morphing*.

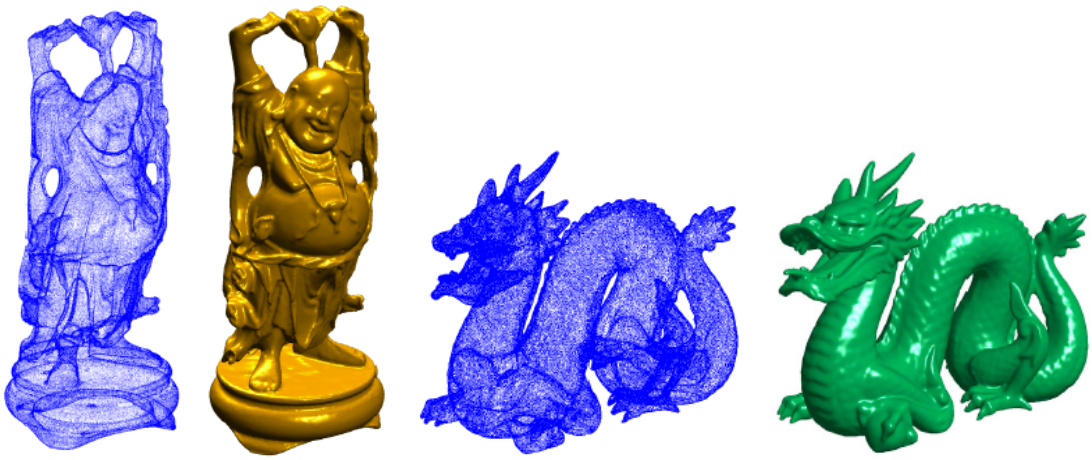


Figura 3.4: Reconstrução de modelos obtida usando o método *FastRBF* [12]. A nuvem de pontos para o Buda e o Dragão é constituída de 543,652 e 437,645 pontos respectivamente.

3.3.1 Trabalhos iniciais

A idéia de usar RBFs para modelagem de superfícies implícitas foi introduzida por Savchenko [42] e Turk e O'Brien [50]. O método consiste em produzir um campo escalar, do qual a superfície desejada é o conjunto zero, enquanto que pontos interiores e exteriores à superfície são mapeados para valores negativos e positivos. Infelizmente, a natureza global desta representação limita seu uso na modelagem de superfícies descritas por grande quantidade de pontos.

Morse et al. [33] usaram funções base de suporte compacto, introduzidas por Wendland [53] para confrontar este problema. Kojekine et al. [26] melhoram o método organizando a matriz esparsa produzida por Morse numa matriz esparsa de diagonal dominante, a qual pode ser resolvida mais eficientemente. Devido aos múltiplos conjuntos de nível zero criados por este método, a função resultante tem aplicações limitadas em CSG, interpolação ou aplicações similares [33].

Carr et al. [12] usaram RBFs para reconstruir a superfície de objetos para as quais dados de intervalos (*range data*) eram disponíveis. Para poder lidar com grandes quantidades de dados, eles se beneficiaram das otimizações reportadas por Beatson [7, 6]. A Figura 3.4 mostra alguns dos resultados obtidos na interpolação/aproximação de grandes quantidades de dados. Mais tarde, Laga et al. [27] introduziram as chamadas Funções Radiais Paramétricas (*Parametric Radial Basis Functions*), trabalho similar ao de Carr, mas adaptado para suportar objetos suaves como também objetos com pontas e vincos.

Recentemente, Ohtake et al. [36] introduziram a *Multi-level Partition of Unity Impli-*

cits, uma nova classe de representação implícita especificamente desenhada para a criação rápida e exata de superfícies constituídas de milhões de pontos. Basicamente, a função implícita global da superfície é dada por uma combinação de funções quádricas respeitando um peso derivado do método de partição de unidade.

Tobor et al. [45] apresentam um método similar ao anterior. Eles dividem o domínio de reconstrução global em pequenos subdomínios. Resolvem o problemas de reconstrução, para cada subdomínio, usando RBFs de suporte global e constroem uma solução global baseada na combinação das soluções locais empregando o método de partição de unidade.

3.3.2 Superfícies implícitas variacionais

Definição: Dado um conjunto de n pontos distintos $\{c_1, c_2, \ldots, c_n\}$, $c \in \Re^3$, e um conjunto de valores de funções para cada um dos pontos $\{v_1, v_2, \ldots, v_n\}$, $v \in \Re$, encontrar uma função f que interpole suavemente os dados, tal que $f(c_i) = v_i$, para $i = 1...n$, onde a suavidade de $f : \Re^3 \mapsto \Re$ é medida minimizando o funcional de energia de segunda ordem

$$\int_{\mathbf{x}\epsilon\Re^3} \sum_{i,j} (\frac{\partial^2 f(\mathbf{x})}{\partial \mathbf{x}_i \partial \mathbf{x}_j})^2 d\mathbf{x}. \tag{3.7}$$

Existe uma única solução para este problema: é a interpolação *thin-plate* dos pontos [28, 10]. O problema pode ser resolvido por meio de métodos numéricos, e os dois comumente usados são os métodos de elementos finitos e diferenças finitas. Alternativamente a solução pode ser expressa em termos de uma combinação linear de RBFs:

$$f(\mathbf{x}) = \sum_{j=1}^{n} \lambda_j \phi(\mathbf{x} - c_j) + p(\mathbf{x}), \tag{3.8}$$

onde c_j são as posições dos pontos de restrição conhecidos, λ_j são os pesos das funções radiais centralizadas naqueles pontos e $p(\mathbf{x})$ é um termo polinomial.

Para a solução *thin-plate* do problema de reconstrução de superfícies, Turk e O'Brien [50] usaram $p(\mathbf{x}) = a + bx + cy + dz$ e o *triharmonic spline*[*] $\phi(\vec{r}) = \|\vec{r}\|^3$, como função base. Na prática, para obter uma função que defina implicitamente a superfície, eles usaram dois conjuntos de pontos. No primeiro, todos os pontos têm valor $v_i = 0$, e portanto são os pontos que definem a superfície. O segundo conjunto é constituído

[*]Embora eles tenham usado a função base *thin-plate*, deve-se notar que outras funções radiais podem também ser empregadas.

por pontos com valores positivos ou negativos, gerados a partir dos vetores normais à superfície. Eles definem a orientação da superfície. O conjunto zero da função implícita f é chamado de superfície implícita variacional.

Para resolver os λ_j e os coeficientes do polinômio a, b, c, d, precisamos fazer satisfazer as restrições de interpolação $f(c_i) = v_i$. Substituindo o lado esquerdo na equação (3.8), temos:

$$\sum_{j=1}^{n} \lambda_j \phi(c_i - c_j) + p(c_i) = v_i. \tag{3.9}$$

Além disso, os pesos λ_j têm que satisfazer as condições de ortogonalidade (veja Seção 3.1.1):

$$\sum_{i=1}^{n} \lambda_i = \sum_{i=1}^{n} \lambda_i c_i^x = \sum_{i=1}^{n} \lambda_i c_i^y, \sum_{i=1}^{n} \lambda_i c_i^z = 0, \tag{3.10}$$

onde $c_i = (c_i^x, c_i^y, c_i^z)$.

Posto que as equações (3.9) e (3.10) são lineares com respeito aos desconhecidos λ_j, a, b, c e d, formulamos o problema como um sistema de equações lineares. Seja $\phi_{ij} = \phi(c_i - c_j)$. Então, o sistema linear pode ser escrito como:

$$\begin{pmatrix} \phi_{11} & \phi_{12} & \dots & \phi_{1n} & 1 & c_1^x & c_1^y & c_1^z \\ \phi_{21} & \phi_{22} & \dots & \phi_{1n} & 1 & c_2^x & c_2^y & c_2^z \\ \vdots & \vdots & & \vdots & \vdots & \vdots & \vdots & \vdots \\ \phi_{k1} & \phi_{k2} & \dots & \phi_{kn} & 1 & c_k^x & c_k^y & c_k^z \\ 1 & 1 & \dots & 1 & 0 & 0 & 0 & 0 \\ c_1^x & c_2^x & \dots & c_n^x & 0 & 0 & 0 & 0 \\ c_1^y & c_2^y & \dots & c_n^y & 0 & 0 & 0 & 0 \\ c_1^z & c_2^z & \dots & c_n^z & 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_n \\ a \\ b \\ c \\ d \end{pmatrix} = \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \tag{3.11}$$

Este sistema é simétrico e semi-definido positivo e pode ser resolvido por métodos diretos ou iterativos.

Capítulo 4

Uma interface baseada em traços para modelagem 3D

4.1 Introdução

Uma vez que sistemas de modelagem 3D típicos são projetados para a construção cuidadosa de modelos precisos, a criação direta e rápida de modelos simples e de aparência *livre* ainda é uma tarefa difícil e tediosa. Além disso, as interfaces estilo *WIMP* (*Windows, Icons, Menus, Pointers*) com que esses sistemas são implementados normalmente requerem que se aprenda a manipular operações de edição complicadas, o que representa uma dificuldade adicional para usuários casuais.

Ultimamente, a modelagem interativa e de manipulação direta de superfícies de objetos 3D usando interfaces baseadas em traços tem se tornado alvo de grande interesse. Embora várias propostas experimentais [22, 25, 37, 3] apresentem excelentes resultados, ainda há muito a se explorar neste campo.

Neste sentido, e estendendo as idéias iniciais sugeridas por Igarashi et al. [22], este capítulo explora o problema descrito e apresenta uma interface baseada em traços para a modelagem interativa de objetos simples, de aparência *livre* e topología arbitraria.

A geometria das superfícies dos objetos modelados é especificada usando uma representação implícita baseada em RBFs [50, 12, 18, 19, 51, 56]. Em particular, adotamos o enfoque analítico das superfícies implícitas variacionais proposto por Turk et. al. [50, 49, 51], e introduzimos o uso de uma malha poligonal grosseira como suporte para a criação da representação implícita.

O sistema protótipo apresenta duas operações principais de modelagem, criação e combinação. Adicionalmente, são apresentadas as operações de extrusão e de furação, as quais estão implementadas usando basicamente a operação de combinação. Esta última

está baseada em testes do tipo interior/exterior para a localização de pontos que pertencem à superfície do objeto combinado.

Além da representação analítica, cada objeto da cena usa uma malha poligonal para fins de visualização e como suporte auxiliar para a execução das operações de extrusão e de furação. Esta malha poligonal é produzida pelo processo de poligonização do nível zero da função implícita.

Por outro lado, os problemas inerentes à ambigüidade das interfaces baseadas em traços fizeram necessário explorar uma arquitetura de resolução de ambigüidades para sua adaptação ao nosso protótipo [2].

Em situações nas quais o sistema não pode resolver uma ambigüidade, é necessária a intervenção do usuário. Por exemplo, um traço fechado pode ser interpretado como a curva base para a execução de uma operação de extrusão ou como uma silhueta inicial para uma operação de criação. Abordamos este problema fazendo uso de uma interface sugestiva [21] como mecanismo de mediação. Uma das possíveis ações é apresentada numa pequena imagem (*thumbnails*), que será escolhida pelo usuário caso represente a ação desejada.

Este capítulo está organizado como se segue. A seção 4.2 apresenta brevemente trabalhos relacionados ao problema em questão. A descrição do sistema é apresentada na Seção 4.3 e o detalhamento dos seus algoritmos é feito na Seção 4.4. Os aspectos relacionados aos detalhes da implementação são abordados na Seção 4.5, e os resultados e limitações na Seção 4.6.

4.2 Trabalhos relacionados

Nos últimos anos, têm sido apresentadas diversas propostas experimentais que oferecem interfaces para a especificação e construção de diferentes tipos de cenas tridimensionais, a partir de traços 2D [57, 46, 30, 22, 15, 14, 47, 21, 43].

Zeleznik et al. [57] apresentam *Sketch*, que é uma interface baseada em gestos, para a criação de cenas 3D compostas de um conjunto de primitivas geométricas predefinidas.

Wyvill et al. [55] introduzem um sistema de modelagem, *BlobTree*, baseado em superfícies implícitas, que permite a descrição de modelos complexos. *BlobTree* tem por base a composição de objetos arbitrários por meio de operações booleanas. É construído sobre um enfoque hierárquico do tipo CSG e com suporte a operações tais como *blending* e *warping*. A especificação dos objetos é feita pelo traçado de seus esqueletos.

Porém, o problema de criar superfícies de objetos a partir de traços 2D, foi recentemente abordado por Igarashi et al. [22]. Eles apresentam o sistema *Teddy*, que consiste de uma interface para modelagem de objetos simples de topologia esférica. O usuário especifica um traço 2D fechado e o sistema automaticamente constrói um modelo 3D plausível da superfície do objeto.

Operações de edição posteriores (extrusão, dobra, corte e suavização) podem ser realizadas para modificar o modelo inicial criado. A aplicação destas operações implica necessariamente o uso de algoritmos de suavização da malha poligonal editada. Apesar disso, alguns modelos resultam em malhas poligonais grosseiras com protuberâncias e rugas, compostas de triângulos com características desagradáveis. Outra característica deste sistema é que permite criar somente objetos arredondados, ou seja, não é possível criar um touro ou objetos similares. Além disso, não tem suporte para a criação de objetos múltiplos na mesma cena; logo, operações para combinar estes não podem ser feitas.

Como um esquema alternativo e favorecendo-se da suavidade natural das representações implícitas, Karpenko et al. [25] abordam o problema usando uma representação implícita baseada em RBFs para a representação da superfície dos objetos. Com este enfoque, algumas questões são mais facilmente abordadas, porém outras são muito mais complicadas (por exemplo, o suporte de modelos com vincos e pontas). Devido aos vértices das malhas poligonais produzidas pelo processo de visualização serem usados como pontos de restrição nas operações de edição, e sendo estes numerosos, o sistema apresenta problemas de performance.

Owada et. al. [37] apresentam um sistema que gera modelos volumétricos a partir de traços 2D. Além de possibilitar criar, cortar, extrudar, permite a especificação de estruturas internas nos modelos. A representação volumétrica permite a criação de modelos com estruturas topológicas variáveis. Não obstante, para alcançar superfícies suaves requere-se pagar um alto costo de armazenamento em alguns casos.

Blobmaker [3] é um protótipo similar ao apresentado por Karpenko. A principal contribuição deste trabalho é a proposta de uso de um esqueleto na construção dos modelos. O uso deste esqueleto permite a criação de objetos com formas arbitrárias e a aplicação eficiente de suas operações de edição. Entretanto, o uso de pontos de restrição posicionados irregularmente na superfície a se criar pode provocar vazamentos de superfície depois da aplicação de algumas operações de edição.

4.3 Descrição do sistema

O sistema protótipo permite a criação rápida de modelos simples 3D, recebendo como entrada traços 2D feitos diretamente na janela do sistema. Uma vez criado o modelo, este pode ser editado fazendo-se uso de operações de edição, tais como: combinação, furação e extrusão. O usuário especifica estas operações usando traços 2D, deste modo, a execução de uma operação depende da forma do traço e onde foi realizado, sem ser necessário apertar nenhum botão de comando ou da seleção de opções de menus.

A interface de usuário está composta de uma janela de desenho e cinco botões de comando que suportam as seguintes operações básicas: *init*, *save*, *undo*, *redo*, *quit* (veja a Figura 4.1). O botão *init* inicializa o sistema para começar uma nova sessão de modelagem, *save* salva a malha poligonal do objeto modelado, *undo* desfaz a última operação realizada, *redo* refaz a última operação *undo*; e o botão *quit* faz sair do sistema. As operações *undo*/*redo* são de natureza linear simples e ilimitada, ou seja, com base neste mecanismo o sistema pode salvar e percorrer todas as operações realizadas pelo usuário durante uma sessão de modelagem.

Para efetuar os traços na janela de desenho, o usuário deve usar o botão esquerdo do mouse. O botão direito é usado para efetuar operações de translação no plano *xy*, além de operações de rotação. Por último, o botão do meio é usado para realizar translações no eixo *z*.

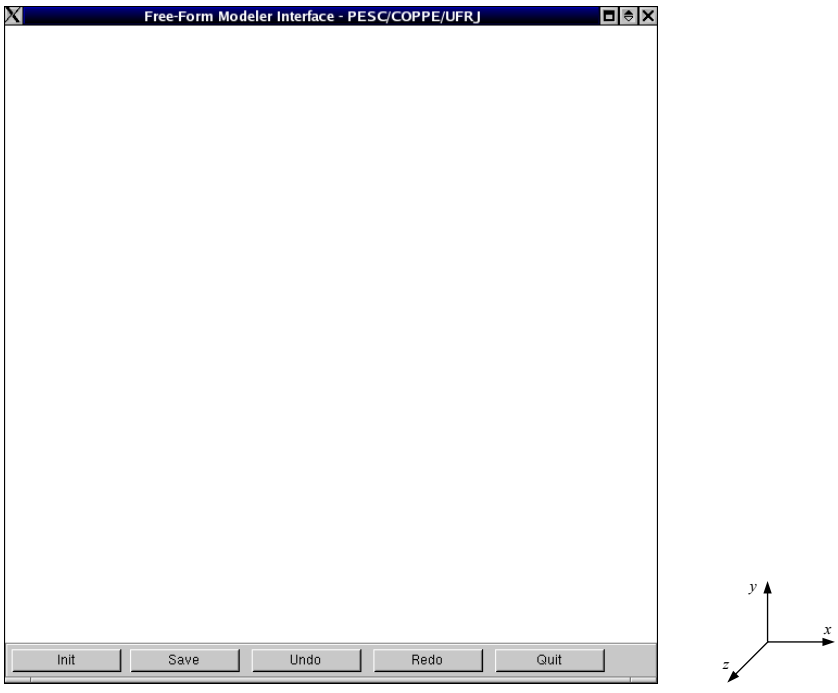


Figura 4.1: Interface do usuário e sistema de coordenadas de referência.

4.3.1 Operações

Uma sessão de modelagem começa com a janela de desenho em branco. O usuário especifica a silhueta do modelo a construir, desenhando um polígono simples num único traço. O sistema constrói automaticamente um modelo 3D baseado na silhueta desenhada, inflando o polígono em ambas as direções numa medida proporcional à sua largura, isto é, áreas estreitas originarão regiões esguias enquanto que áreas amplas gerarão regiões volumosas. Se a curva traçada não representar um polígono simples, o sistema simplesmente não realiza nenhuma operação. A Figura 4.2 ilustra exemplos de traços de entrada e os correspondentes modelos 3D criados pelo sistema.

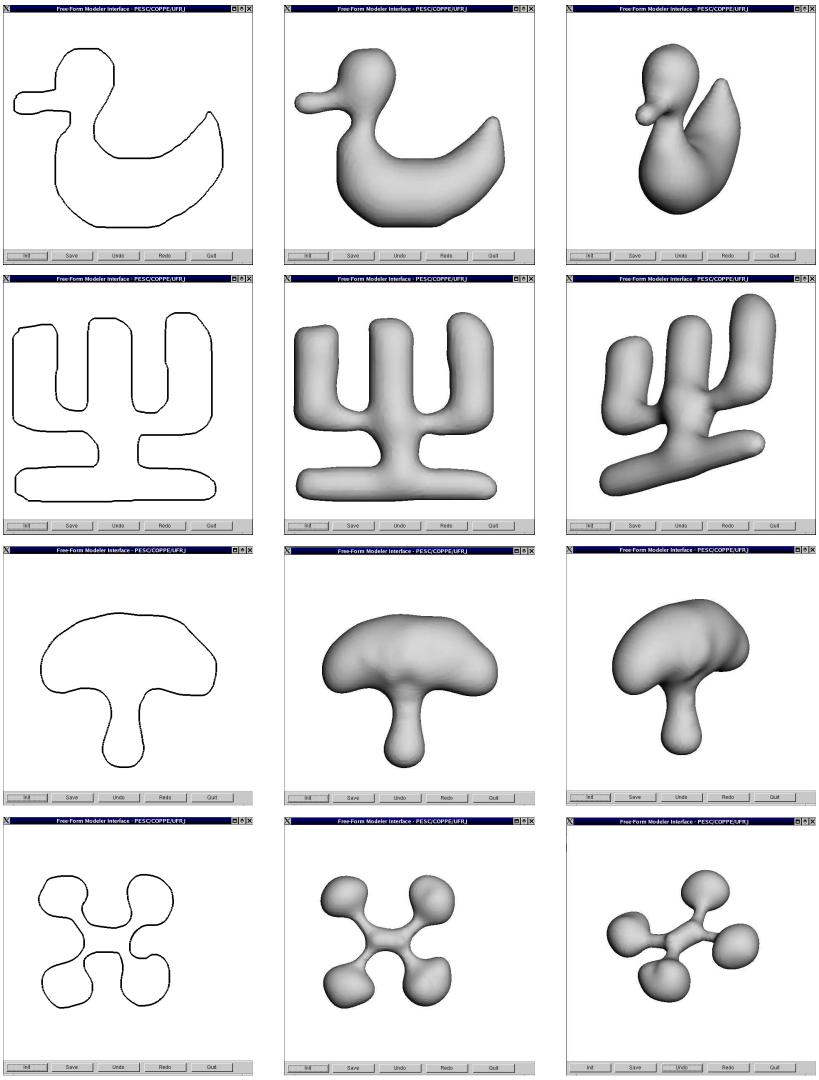


Figura 4.2: Exemplos da operação de criação.

A operação de *criação* pode ser realizada várias vezes; assim, o sistema protótipo tem suporte para modelagem de cenas constituídas por múltiplos objetos (Figura 4.3).

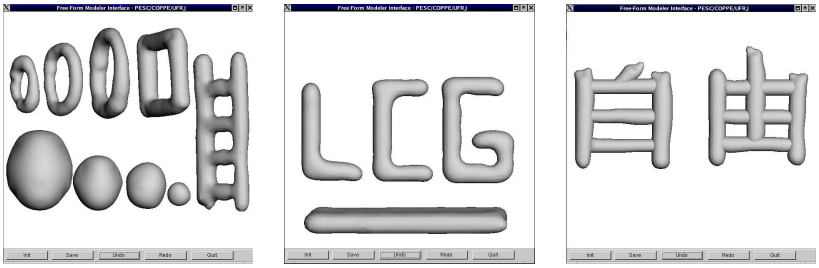


Figura 4.3: Suporte de cenas com múltiplos objetos.

Combinação é uma operação que acontece quando é efetuado um traço, aberto e simples, que começa dentro da silhueta de um modelo e termina dentro da silhueta de outro modelo sobreposto ao primeiro. O sistema automaticamente cria um novo modelo produto da combinação dos dois modelos iniciais (Figura 4.4).

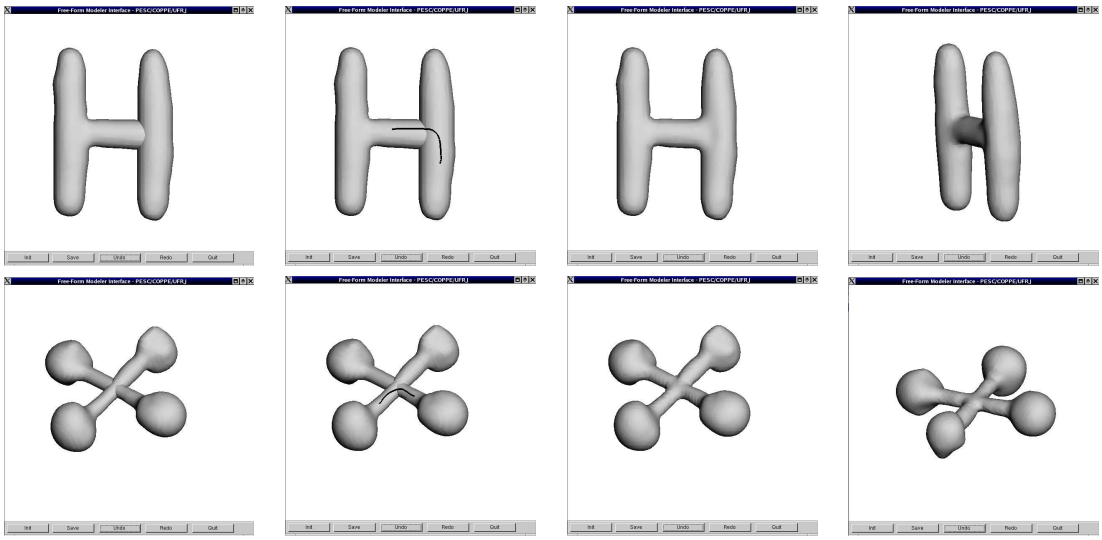


Figura 4.4: Exemplos da operação de combinação.

O sistema executa uma operação de *furação* quando o usuário desenha um traço fechado e sem auto-interseção diretamente sobre a superfície de um objeto (Figura 4.5).

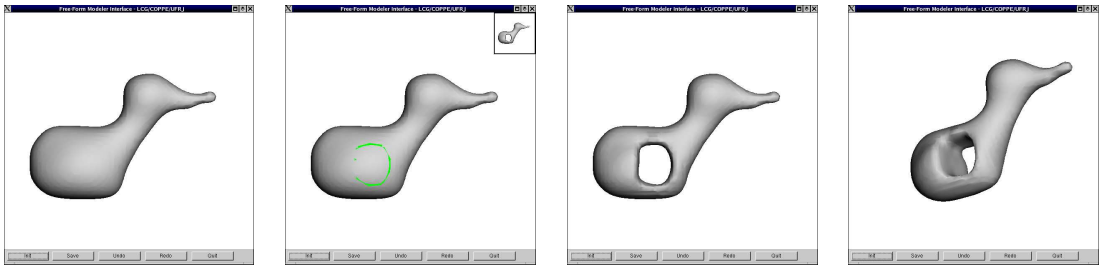
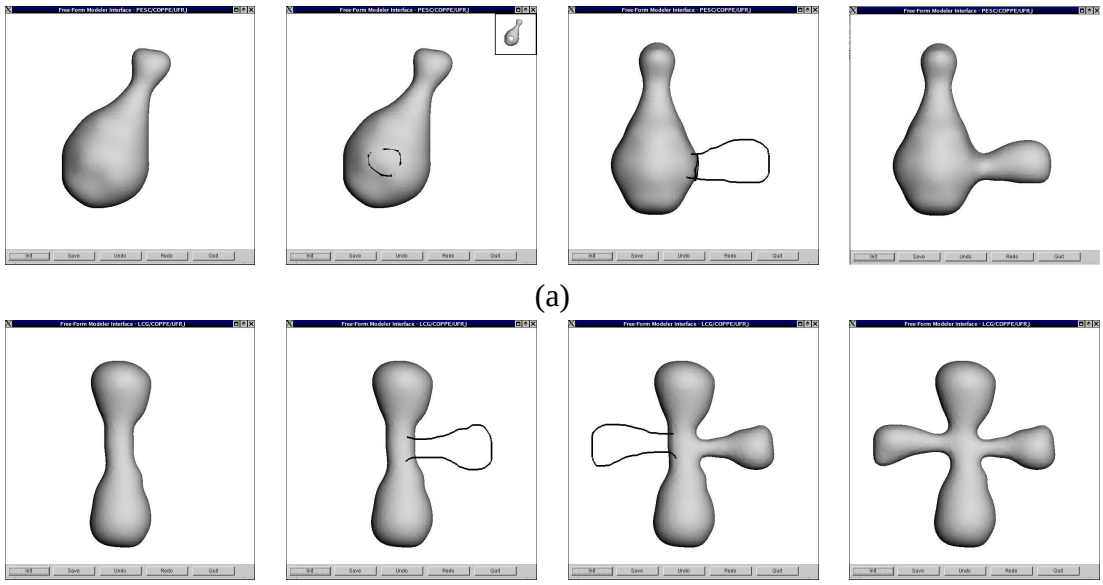


Figura 4.5: Exemplo da operação de furação.

Extrusão é uma operação que pode ser realizada de duas formas. Através da especificação de só uma curva (que representará o perfil de extrusão) ou com a especificação de duas curvas: uma curva base (que indica a área a extrudar) e outra que descreve o perfil de extrusão. Para que a curva base seja considerada como válida, ela deve representar um polígono simples e ter sido desenhada totalmente sobre a superfície do modelo a ser editado. Uma vez especificada, o sistema projeta a curva base sobre a superfície do modelo e a destaca, indicando estar pronto para realizar uma operação de extrusão. Em seguida, o usuário roda o modelo posicionando a curva base de lado e desenha o traço que representa o perfil de extrusão (Figura 4.6(a)).

O perfil de extrusão deve ser uma curva aberta e sem auto-interseção, que começa e termina dentro da superfície do modelo, e desenhada bem perto da curva base, se for o caso. A diferenciação entre os dois tipos de extrusão está baseada na presença/ausência da curva base no momento da especificação do perfil de extrusão (Figura 4.6(b)).



(a)

(b)

Figura 4.6: Exemplos da operação de extrusão. (a) Extrusão usando uma curva base. (b) Extrusão automática (sem curva base).

Quando um traço fechado e sem auto-interseção é desenhado sobre um modelo, o sistema enfrenta um caso de ambigüidade. O usuário pode querer furar o modelo ou especificar uma curva base para realizar uma operação de extrusão. Para lidar com este problema, o protótipo utiliza um mecanismo básico de mediação que consiste em apresentar numa imagem miniatura (*thumbnail*) na parte superior direita da tela o resultado da operação de furação, a qual será escolhida pelo usuário com um clique sobre ela, caso

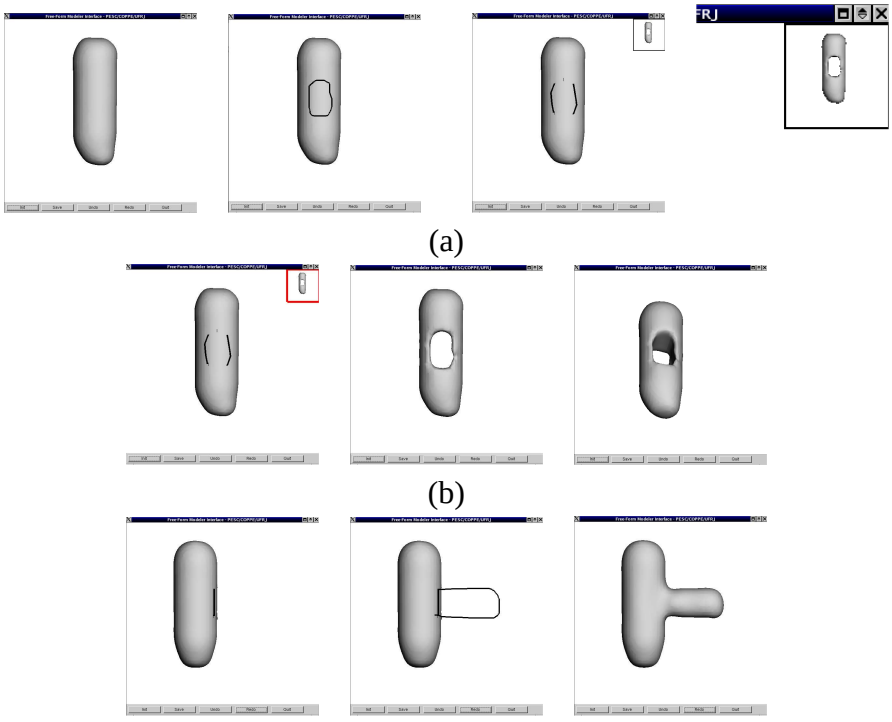


Figura 4.7: Mediação sugestiva em caso de ambigüidade. (a) A especificação de um traço fechado e sem auto-interseção sobre um modelo origina um problema de ambigüidade. Pode-se querer furar ou extrudar o modelo. (b) O usuário escolhe a operação de furação clicando sobre o *thumbnail* apresentado pelo sistema. (c) O usuário ignora o *thumbnail* apresentado em (a), rotaciona o modelo e especifica o perfil de extrusão.

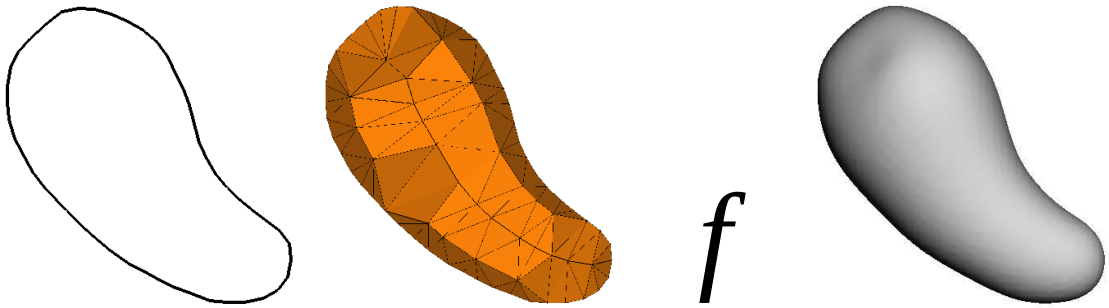
deseje executar essa operação. Se o usuário quiser realizar uma operação de extrusão, simplesmente ignora o *thumbnail* apresentado (clicando fora do *thumbnail*) e roda o modelo numa posição adequada para receber o traço que representará o perfil de extrusão. Veja na Figura 4.7 uma ilustração do exposto.

Fazendo uso de translações e rotações, o usuário pode selecionar o melhor perfil do modelo para a execução das operações de edição. Para transladar um modelo, no plano *xy*, deve-se simplesmente arrastá-lo com o botão direito do mouse. A translação no eixo *z* é feita da mesma forma, só que com o botão do meio. A rotação é realizada, especificando-se inicialmente o ponto de rotação com um simples clique (com o botão direito) sobre a superfície do modelo a rotacionar; depois basta arrastar o modelo com o botão direito do mouse.

4.4 Algoritmos

4.4.1 Criação

O algoritmo para a criação de modelos simples 3D consiste na especificação dos pontos que representam a superfície do objeto modelado a partir de um traço 2D, na construção da função f que representa implicitamente esta superfície e na visualização desta última. A figura 4.8 ilustra uma idéia global do algoritmo.



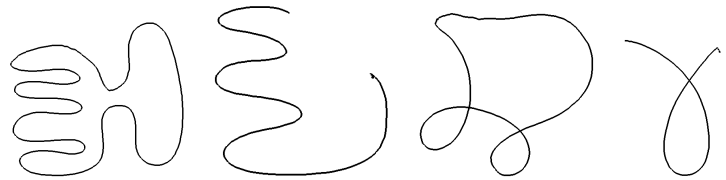
(a) (b) (c) (d)

Figura 4.8: (a) Traço inicial 2D. (b) Malha poligonal de suporte para a especificação da superfície. (c) Função analítica $f : \Re^3 \mapsto \Re$. (d) Superfície implícita poligonizada.

1. Geração de pontos de restrição. O objetivo desta etapa é a geração de uma malha poligonal grosseira M, construída a partir de um polígono simples 2D. Os vértices de M serão chamados de pontos de restrição de fronteira e serão usados na etapa seguinte do algoritmo. A construção de M baseia-se no método empregado por Igarashi et al. [22] e consiste dos seguintes passos:

   - Pré-processamento: O usuário especifica um traço de entrada clicando com o botão esquerdo do mouse e arrastando-o na janela até que o botão seja solto. Enquanto o mouse é arrastado, o sistema capta todas as posições (pontos 2D) que atravessa, desenhando segmentos de reta entre os pontos visitados.

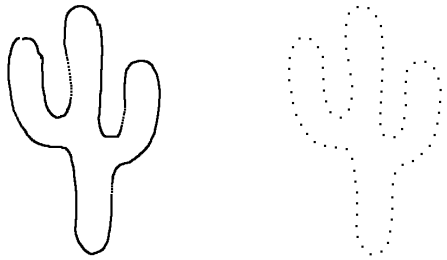
     Uma vez que o traço 2D é especificado, ele é classificado numa das seguintes categorias (Figura 4.9): traço fechado simples, aberto simples, fechado não-simples e aberto não-simples. Um traço fechado é aquele com seus pontos inicial e final próximos, contrariamente a um traço aberto, que tem seus pontos inicial e final afastados. Um traço simples é aquele que não tem auto-interseção.



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

Figura 4.9: Classificação de traços 2D. (a) Fechado simples. (b) Aberto simples. (c) Fechado não-simples. (d) Aberto não-simples.

Depois que o traço é classificado, e sendo ele do tipo fechado simples, os pontos gerados pelo arrasto do mouse são reamostrados para ficarem distribuídos de modo uniforme (Figura 4.10). Internamente, este traço é representado como um polígono simples com arestas de comprimento uniforme. Chamaremos a este polígono de S e as suas arestas de arestas externas, enquanto que as arestas adicionadas pelo processo de triangulação seguinte serão chamadas de arestas internas.



(a) (b)

Figura 4.10: (a) Traço 2D efetuado pelo usuário. (b) Amostragem uniforme realizada pelo sistema.

- Construção da malha poligonal M: Este processo está subdividido nas seguintes etapas:
  - Triangulação 2D: As arestas de S são inseridas numa triangulação de Delaunay restrita, a qual será chamada de T (Figura 4.11(b)).
  - Classificação dos triângulos: As faces de T são classificadas em quatro grupos: faces terminais (triângulos com duas arestas adjacentes a S), faces de borda (triângulos com uma aresta adjacente a S), faces de junção (triângulos sem arestas adjacentes a S) e faces externas (triângulos que se encontram fora de S). Veja a Figura 4.11(c).
  - Obtenção do eixo cordal CA: Este processo se inicia na aresta interna de um triângulo do tipo terminal e conecta os pontos médios das arestas

internas dos triângulos de borda vizinhos. Caso o triângulo visitado seja do tipo junção, calcula-se o centro do triângulo e visita-se recursivamente T a partir das outras duas arestas (Figura 4.11(d)).

- Identificação de triângulos irrelevantes: Para cada ramo de CA, e começando numa face terminal, procede-se à identificação de triângulos irrelevantes, isto é, triângulos que estão contidos no círculo cujo diâmetro é igual ao comprimento da última aresta, não restrita, visitada neste processo (Figura 4.11(e)).
- Cálculo da espinha: A espinha de S é constituída por todos os vértices de CA, exceto aqueles cujas arestas pertencem somente a triângulos irrelevantes (Figura 4.11(f)).
- Suavização da espinha: As arestas da espinha são passadas por um processo de suavização. (Figura 4.11(g)).
- Inserção da espinha na triangulação: Todos os vértices da espinha são inseridos na triangulação; em seguida, são elevados a uma altura proporcional à distância média entre ele e os seus vértices externos diretamente conectados (Figura 4.11(h)).
- Remoção de arestas: A triangulação resultante é simplificada com base num processo de remoção de arestas internas de comprimento menor que t_1 ($t_1 = 20$ na nossa implementação). Veja Figura 4.11(i).
- Inserção de vértices adicionais: Cada aresta interna e_i de T, excluindo as arestas da espinha, é convertida em um quarto de elipse. O número de pontos que constituem o quarto de elipse é diretamente proporcional ao comprimento de e_i (Figura 4.12). Estes pontos são inseridos em T e evitarão o vazamento de superfícies (veja Figura 4.16) produto da aplicação de operações de edição posteriores nos modelos.
- Geração da malha poligonal: Finalmente, constrói-se uma malha poligonal grosseira, fechada e simétrica, copiando a triangulação T do outro lado do plano de suporte de S (Figura 4.13).

2. Construção da função implícita interpolante. Esta etapa começa com a especificação dos pontos de restrição que constituirão os dados de entrada para a construção da função implícita baseada em RBFs, função que chamaremos de f, onde o conjunto de nível zero de f representará a superfície do objeto modelado (veja Capítulo

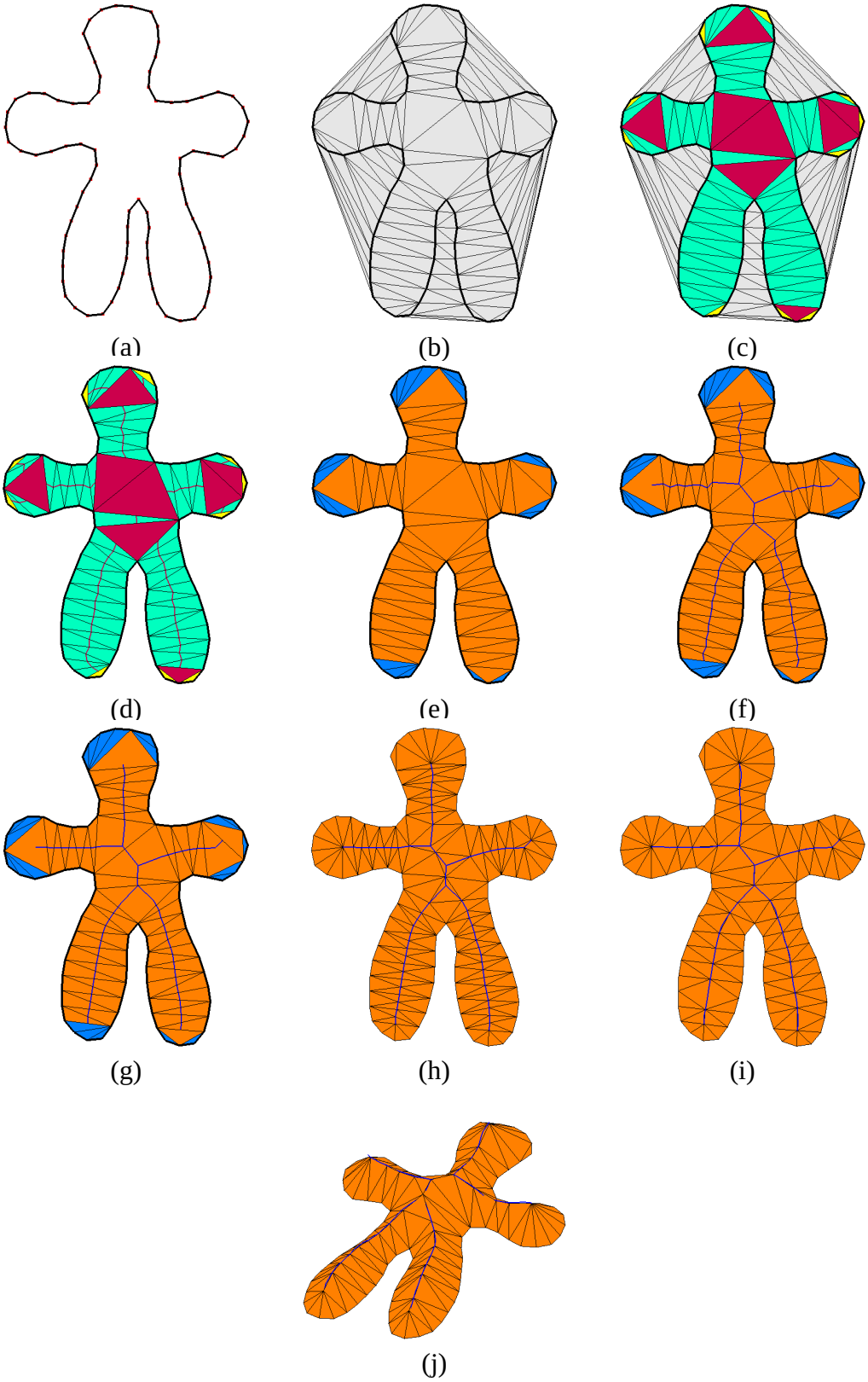


Figura 4.11: (a) Polígono de entrada. (b) Triangulação de Delaunay restrita. (c) Classificação dos triângulos. (d) Cálculo do eixo cordal. (e) Identificação de triângulos irrelevantes. (f) Determinação da espinha. (g) Suavização da espinha. (h) Inserção dos vértices da espinha na triangulação inicial. (i) Remoção de arestas pequenas. (j) Triangulação com elevação da espinha.

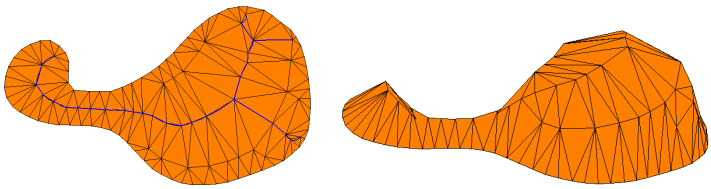
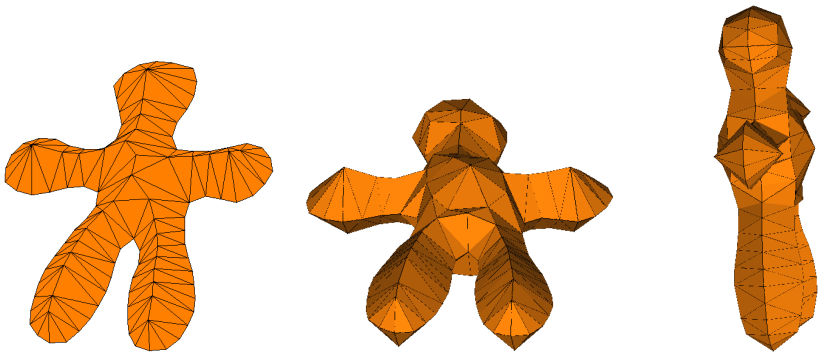


Figura 4.12: (a) Triangulação 2D restrita. (b) Elevação da espinha e geração de novos vértices entre as arestas internas.



(a) (b)

Figura 4.13: (a) Triangulação com vértices elevados. (b) Duas vistas da malha poligonal grosseira construída (177 vértices e 350 triângulos).

3).

- Especificação dos pontos de restrição: Uma função implícita baseada em RBFs é modelada através da especificação de um conjunto de pontos de restrição 3D junto com um correspondente conjunto de valores escalares. Os pontos de restrição se dividem em dois subconjuntos: os pontos de restrição de fronteira e os pontos de restrição de normais. Os pontos de restrição de fronteira são as posições pelas quais a superfície implícita criada passará exatamente. Os pontos de restrição de normais são as posições que definirão a orientação da superfície.

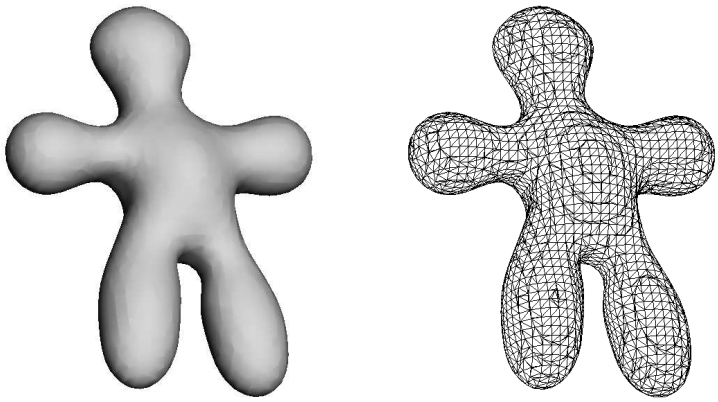
  Seja M a malha poligonal grosseira construída na etapa anterior. Os vértices q_i de M são usados como pontos de restrição de fronteira. Além disso, para cada vértice q_i é estimado um vetor normal (média das normais das faces incidentes), e gera-se um ponto de restrição de normal n_i, deslocando q_i a uma pequena distância d ($d = 1.0$ em nossa implementação) ao longo desse vetor [56].

- Construção e resolução do sistema de equações: Uma vez obtidos os pontos de restrição de fronteira q_i e de normais n_i, construímos o sistema de equações

apresentado no capítulo 3 seção 3.3.2. Nos pontos que determinarão a superfície do modelo utilizaremos zero ($w = 0$) como o valor da função implícita, e o valor de $w = 1.5$ para aqueles que determinarão a orientação da superfície. A razão do uso destes valores assim como a quantidade de deslocamento d serão explicados no capítulo seguinte.

Por último, a resolução do sistema de equações é realizada usando o método de decomposição LU.

3. Visualização. Finalmente, a iso-superfície de nível zero da função implícita f deve ser visualizada. Para este propósito, usamos o algoritmo de poligonização apresentado no capítulo 5. A Figura 4.14 apresenta resultados do algoritmo de visualização e a malha poligonal resultante.



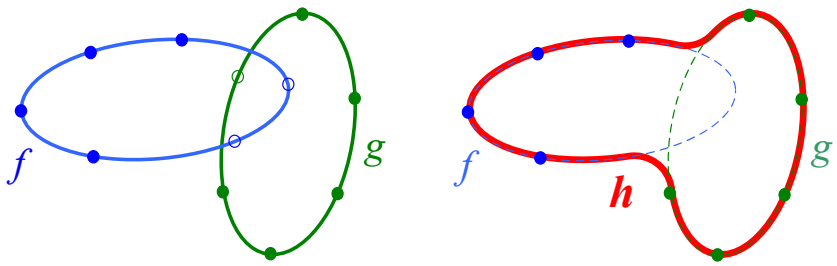
(a) (b)

Figura 4.14: (a) Visualização do nível zero de f. (b) Malha poligonal produzida pelo processo de poligonização de f (3620 vértices e 7236 triângulos).

Devido à forte dependência do algoritmo de visualização em relação ao posicionamento dos pontos de restrição (de fronteira e de normais), e sendo que depois de algumas operações de edição se pode perder aquelas relações entre os valores w e d (valores de função e quantidade de deslocamento), a poligonização da superfície pode gerar resultados inesperados. Isto deve ser resolvido assegurando-nos que a relação w/d seja sempre válida ou usando um algoritmo de poligonização que não dependa do posicionamento dos pontos de restrição.

4.4.2 Combinação

A operação de combinação consiste no processo de criação de um novo objeto representado implicitamente pela função h, a partir da combinação de dois objetos representados



(a) (b)

Figura 4.15: Ilustração 2D da operação de combinação. (a) Os pontos de restrição posicionados dentro da interseção dos modelos representados por f e g são eliminados. (b) O novo modelo, representado pela função h, é construído com aqueles pontos que ficaram depois do processo de eliminação.

pelas funções f e g.

A operação de combinação consiste no processo de eliminar todos os pontos de restrição de f e g que estão posicionados dentro da interseção de ambos modelos. Isto é:

1. todo ponto de restrição x_i que pertence ao modelo representado por f deve ser eliminado se $g(x_i) < 0$; e

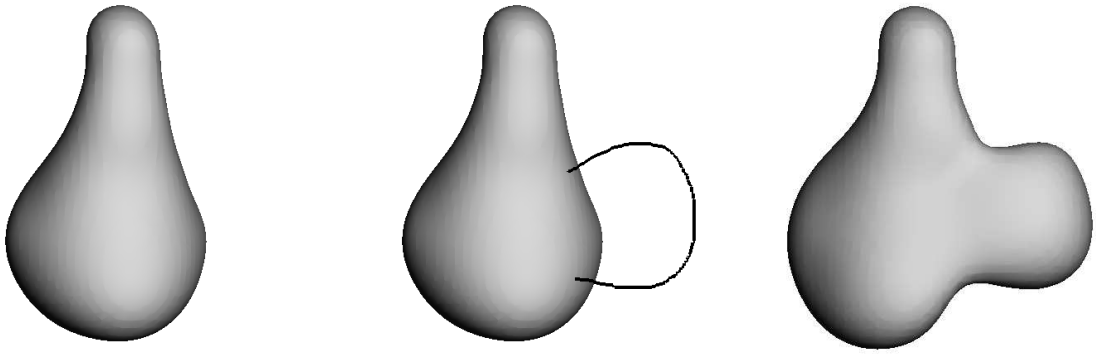
2. todo ponto de restrição y_i que pertence ao modelo representado por g deve ser eliminado se $f(y_i) < 0$.

Desta forma, a função implícita h será construída com o procedimento descrito na seção 4.4.1, porém usando como pontos de restrição todos aqueles pontos (e seus correspondentes valores de função) de f e g, que ficaram do processo de eliminação anterior. A Figura 4.15 ilustra a idéia desta operação.

A aplicação de operações de edição nos modelos pode resultar em problemas de vazamento de superfícies, devido ao posicionamento irregular dos pontos de restrição na criação dos modelos. A Figura 4.16 ilustra este problema. Lidamos com este problema fazendo uso da malha poligonal grosseira introduzida no processo de criação dos modelos (Veja seção 4.4.1).

4.4.3 Furação

Seja f a função que representa o modelo a editar, C a curva base 2D traçada pelo usuário (representada por um polígono simples), e h o modelo resultante deste processo de edição. O procedimento de furação consiste em:



(a) (b) (c)

Figura 4.16: Ilustração do problema de vazamento de superfícies. (a) Interpolação inicial. (b) Extrusão. (c) Vazamento no resultado da extrusão.

1. Projetar C na parte frontal da superfície do modelo poligonizado, gerando uma curva 3D que chamaremos de Cf.

2. Projetar C na parte traseira da superfície do modelo poligonizado, gerando uma curva 3D que chamaremos de Cb.

3. Calcular o *bounding box* B de Cf e Cb.

4. Para cada vértice Cf_i e Cb_i, gerar recursivamente um conjunto de pontos médios $\{pm_i\}$. Em nossa implementação, geram-se três pontos médios para cada par de vértices Cf_i e Cb_i.

5. Criar uma função interpolante g, a qual será construída com o procedimento de criação apresentado na seção 4.4.1, porém usando como pontos de restrição de fronteira os vértices de Cf, Cb e o conjunto de pontos $\{pm_i\}$. Para a definição da orientação da superfície gera-se um ponto de restrição p, tal que seja o centro de massa de B. O valor de função de p será um valor negativo (-1 em nossa implementação).

6. Eliminar todos os pontos de restrição fc_i que estão posicionados dentro do modelo representado por g. Isto é, todo ponto de restrição x_{fi} que pertence ao modelo representado por f deve ser eliminado se $g(x_{fi}) < 0$.

7. Criar a função interpolante h, com o procedimento da seção 4.4.1, porém usando como pontos de restrição de fronteira o conjunto de pontos $\{pm_i\}$ e os pontos de restrição de f que ficaram depois do processo de eliminação anterior. Adicionalmente, incluem-se os vértices de Cf e Cb como pontos de restrição de normais.

4.4.4 Extrusão

O primeiro tipo de extrusão é definido por dois traços: uma curva base feita diretamente sobre a superfície do modelo, a qual definirá a área base afetada pelo processo de edição, e uma curva que definirá o perfil de extrusão. Esta operação de extrusão, feita sobre o modelo representado pela função f, consiste dos seguintes passos:

1. Projetar a curva base na parte frontal da superfície do objeto poligonizado.
2. Projetar a curva que define o perfil de extrusão no plano que passa no baricentro da curva base (rotacionada junto com o modelo) e paralelo ao plano de visão do usuário.
3. Criar uma função interpolante g, com o procedimento da seção 4.4.1, porém usando como pontos de restrição aqueles pontos correspondentes à curva base e ao traço que representa o perfil de extrusão.
4. Aplicar a operação de combinação entre f e g.

O outro caso de extrusão é realizado apenas com a curva que define o perfil de extrusão. O procedimento é o seguinte:

1. Encontrar os vértices S e E da malha poligonal mais próximos ao ponto inicial e final do traço 2D ingressado.
2. Projetar a curva do perfil de extrusão no plano que passa no ponto meio de S e E e paralelo ao plano de visão do usuário.
3. Criar a função interpolante g, com base no procedimento descrito em 4.4.1, usando como traço inicial a curva projetada no passo anterior.
4. Aplicar a operação de combinação entre f e g.

4.5 Detalhes da implementação

O sistema protótipo foi implementado usando C++ padrão. Para a renderização das cenas empregamos a biblioteca *OpenGL*. Todos os modelos foram construídos rodando o protótipo num PC equipado com um processador AMD-Duron correndo a 1.3 GHz com 256 MB de memória principal.

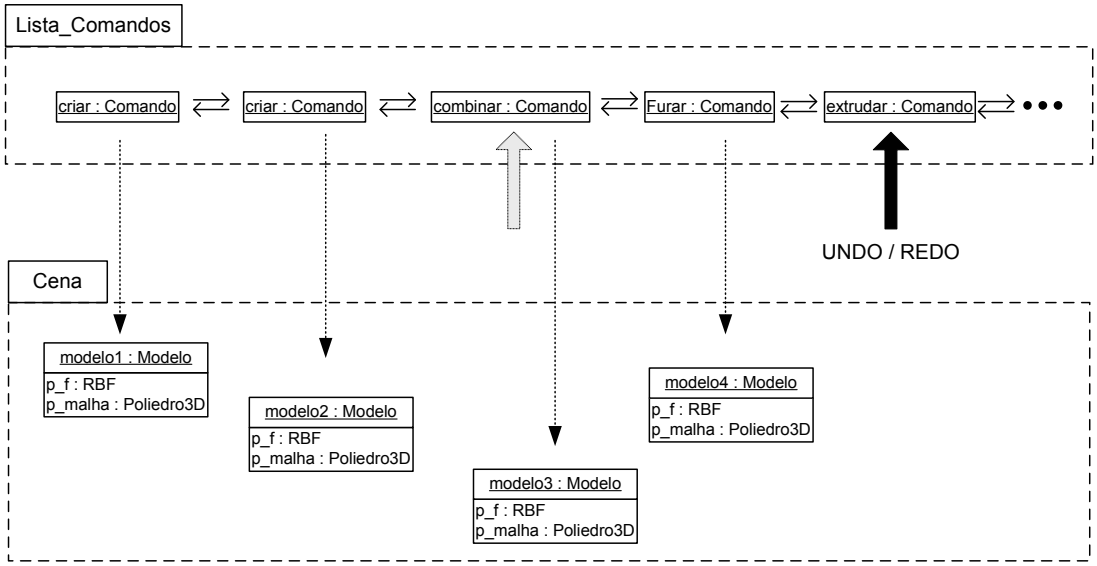


Figura 4.17: Principais estruturas de dados do sistema.

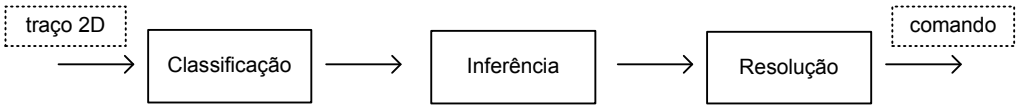


Figura 4.18: Passos para a determinação do comando a se executar, a partir de um traço 2D.

O sistema emprega duas estruturas de dados principais: uma representação da cena e uma lista de comandos. A cena é o repositório de modelos e a lista de comandos é o mecanismo que permite conhecer a história de uma sessão de modelagem (Figura 4.17).

Cada vez que um novo comando vai ser executado na cena, ele é inserido no final da lista de comandos. Dependendo do tipo de comando, a sua execução pode inserir e remover modelos da cena. Por exemplo, um comando “combinar” inserirá um novo modelo na cena, e removerá os modelos combinados.

Desta forma, o mecanismo de *Undo*/*Redo* funciona sobre a lista de comandos do sistema, percorrendo-a em ambas as direções e executando ou desfazendo os comandos adequadamente.

A determinação do tipo de comando a executar, em função dos traços 2D inseridos no sistema, é realizada em três passos (Figura 4.18): (1) A etapa de classificação determina o tipo do traço. (2) Com base no lugar onde o traço foi realizado e dependendo do seu tipo, a etapa de inferência cria o comando adequado e (3) dependendo da escolha do usuário, se for o caso, a etapa de resolução insere o comando na lista e o executa [2].

Uma descrição sucinta da hierarquia de classes do sistema é apresentada na Figura

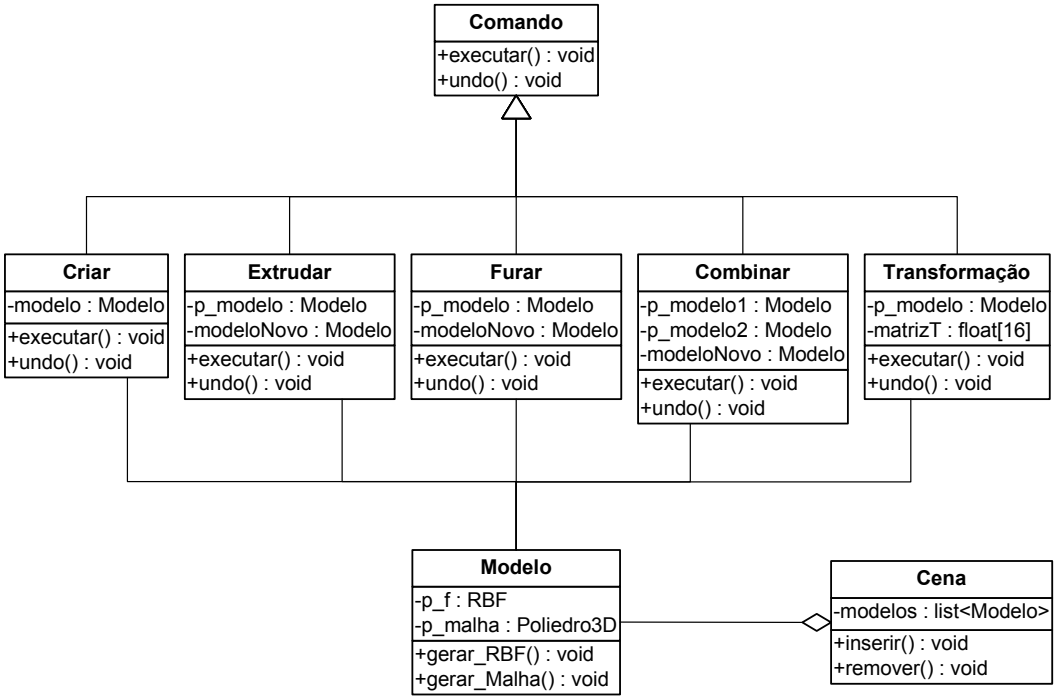


Figura 4.19: Hieraquia de classes do sistema.

4.19. Os atributos *p_modelo* nas classes são ponteiros de modelos, enquanto que a ausência do prefixo *p_* significa uma referência ao próprio modelo.

A superclasse **Comando** é uma classe abstrata com dois métodos: *executar()* e *undo()*. O método *executar()* executa as ações indicadas por um comando. O método *undo()* desfaz as ações feitas pelo método *executar()*. Por exemplo, numa operação de extrusão, *undo()* remove o modelo resultante da operação de extrusão *modelo*, e insere novamente o modelo extrudado *p_modelo*.

Criar é uma classe que implementa o processo de criação de um modelo. **Extrudar** modifica o modelo apontado por *p_modelo* gerando um modelo resultante *modeloNovo*. O mesmo acontece com a classe **Furar**. **Combinar** produz um novo modelo *modeloNovo* a partir dos modelos *p_modelo1* e *p_modelo2*. As rotações e translações são implementadas pela classe **Transformação**.

A geometria de um objeto da classe **Modelo** está especificada por duas representações: uma representação analítica *f*, que é uma função implícita baseada em RBFs e uma malha poligonal gerada uma única vez pela poligonização de *f*.

Por último, a classe **Cena** contém uma lista de modelos (possivelmente vazia) que é manipulada através dos métodos *inserir()* e *remover()*.

4.6 Resultados e limitações

A Figura 4.20 apresenta alguns modelos construídos com o sistema protótipo apresentado. Os modelos estão constituídos de superfícies suaves e de topologia arbitrária e refletem a natureza de desenho à mão *livre* com que foram construídos. O protótipo permite exportar os modelos criados em formato OFF [41].

O sistema foi especificamente desenhado para a criação rápida de modelos simples e de topologia arbitrária e não para a criação precisa e detalhada de modelos complexos. Devido ao tipo de RBFs usadas para a representação da superfície dos objetos, o sistema não suporta a modelagem de objetos que possuem pontas e vincos.

A combinação de dois objetos muito pequenos pode resultar num objeto implícito constituído de duas componentes conexas, sendo que cada componente conexa representaria cada um dos objetos iniciais (Veja Figura 4.21). Esta situação pode ser evitada fazendo uso de malhas poligonais mais finas (maior quantidade de pontos de restrição) no momento da criação dos objetos a combinar.

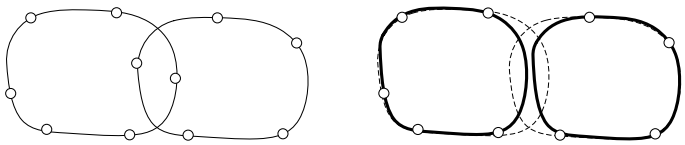


Figura 4.21: A combinação de dois modelos pequenos pode produzir duas componentes conexas.

Uma operação de furação não pode ser executada sobre objetos em qualquer posição. Se o buraco a ser feito atravessa mais de duas vezes a superfície do objeto, a operação falha (Figura 4.22(a)). O mesmo acontece se o traço não representa um polígono simples.

Em alguns casos –por exemplo, quando o buraco a ser feito é muito pequeno–, o processo de interpolação não respeita as restrições que definem o buraco que acaba sendo tampado, produzindo uma superfície com duas componentes conexas. Veja na Figura 4.22(b) uma ilustração deste problema.

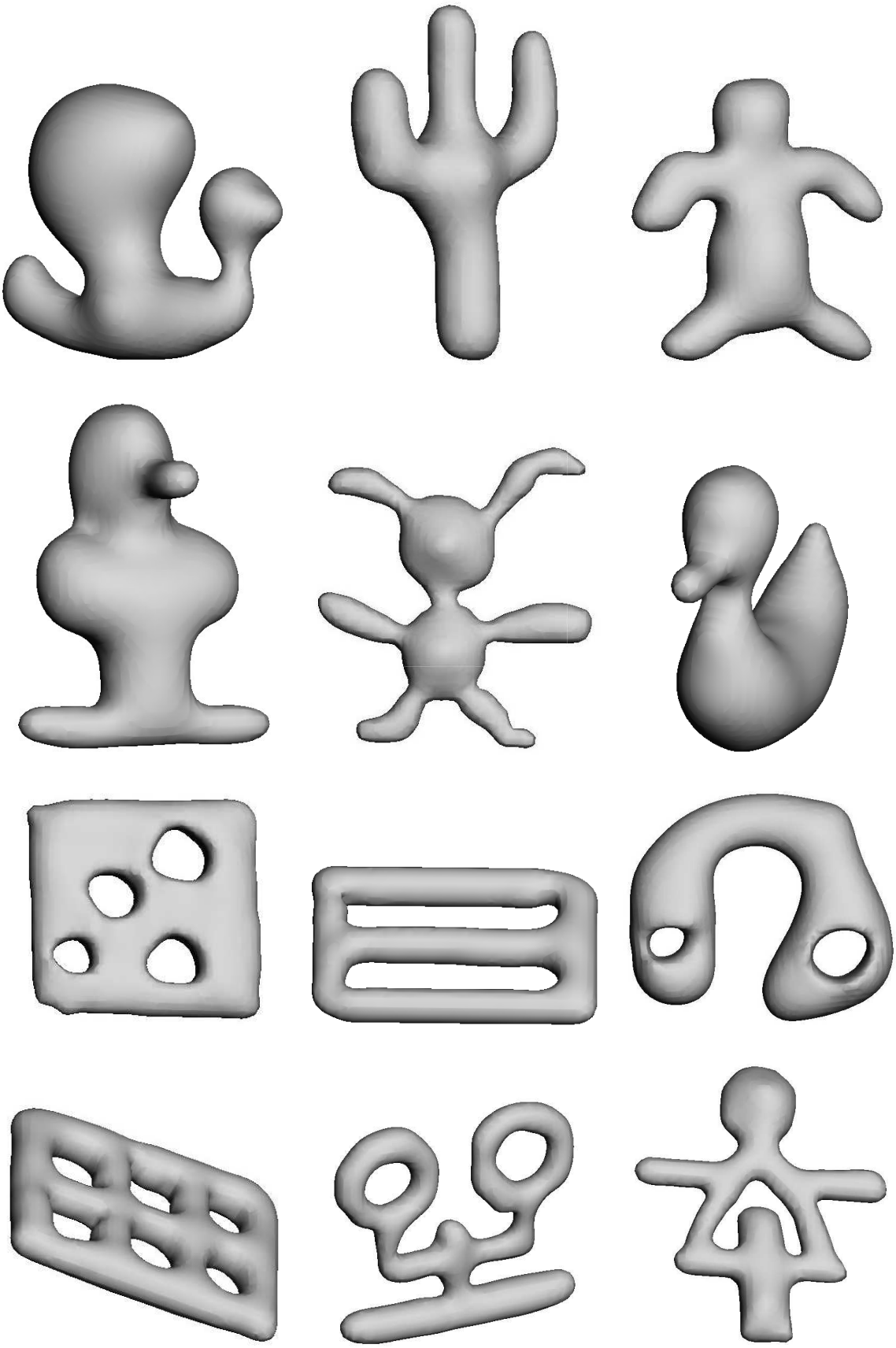


Figura 4.20: Modelos 3D construídos com o sistema protótipo.

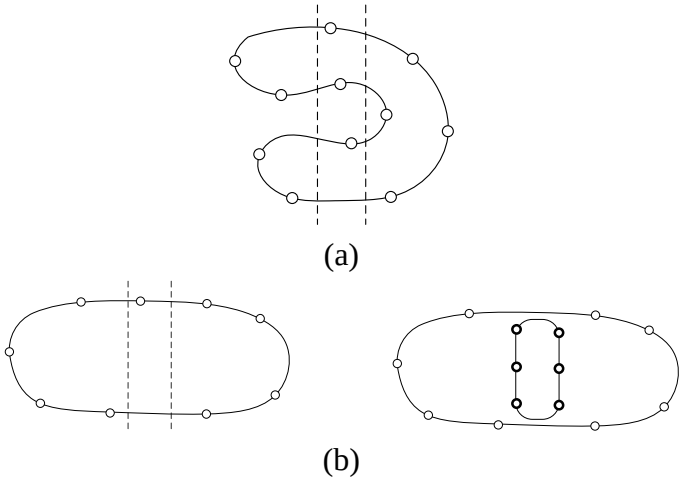


Figura 4.22: Ilustração 2D das limitações da operação de furação. (a) Se o buraco a ser feito atravessar mais de duas vezes a superfície do objeto, a operação falha. (b) Geração de uma superfície com duas componentes conexas.

Uma limitação da operação de extrusão acontece quando o perfil de extrusão é especificado numa posição onde o modelo a extrudar apresenta muitos detalhes. Possivelmente, a operação produzirá um novo objeto com características diferentes das esperadas pelo usuário. Por outro lado, como somente os pontos da curva base e do perfil de extrusão são usados como restrições para uma operação de extrusão, é possível que aconteça vazamento de superfícies. Esta situação pode ocorrer se operações de edição forem posteriormente aplicadas sobre a região do objeto gerada pela extrusão.

A inclusão de pequenos detalhes nos objetos modelados implica na utilização de uma maior quantidade de pontos de restrição, portanto um decorrente decremento na performance do sistema.

Capítulo 5

Poligonização de modelos implícitos baseados em RBFs

5.1 Introdução

O capítulo anterior apresenta algoritmos para a criação e edição de modelos de objetos 3D. Gera-se uma função implícita $f : \Re^3 \mapsto \Re$, produto da interpolação de um conjunto de n pontos, chamados pontos de restrição (*constraint points*). Esta função consiste de um somatório ponderado de funções de base radiais *(RBFs-Radial Basis Functions)* centralizadas nos pontos de restrição, onde o conjunto de nível zero de f representa a superfície do objeto modelado.

Uma visualização de alta qualidade desta iso-superfície requer que f seja avaliada numa grande quantidade de pontos do espaço. Devido à natureza global das funções base usadas, todos os termos de f devem ser usados no cálculo do valor da função em qualquer ponto do espaço*. Dessa forma, cada avaliação de f tem complexidade $O(n)$.

A visualização deste tipo de superfícies pode ser feita usando métodos diretos, tais como traçado de raios (*ray-tracing*) [24]. Além disso, Reuter et al. [40] apresentam uma técnica de renderização baseada em pontos (*point rendering*), desenvolvida especificamente para visualizar superfícies representadas por RBFs. No entanto, a visualização é mais freqüentemente executada empregando um esquema de poligonização, como o algoritmo de continuação de Bloomenthal [9] ou o algoritmo *Marching Cubes* [29].

Muitos pesquisadores da área de modelagem implícita baseada em RBFs preferem usar o algoritmo de Bloomenthal, visto que é bem eficiente e tem uma implementação muito conhecida. Também, tem a vantagem de amostrar o espaço somente na vizinhança

*Deve-se mencionar que métodos aproximados como os descritos em [12] podem ajudar a reduzir o número de termos do somatório.

da iso-superfície desejada. Sendo um método de continuação, requer um ponto semente para cada componente conexa da iso-superfície, e fornecer um conjunto de sementes, completo mas não redundante, pode ser difícil em alguns casos.

Por outro lado, os métodos que amostram o espaço regularmente (e.g., *Marching Cubes*) asseguram produzir um resultado correto. Porém, evidentemente, não são adequados para nosso problema posto que pagam um alto custo computacional devido à amostragem do espaço em posições irrelevantes. Neste contexto, um método hierárquico pode-se provar ser útil, dado que é capaz de convergir rapidamente a regiões do espaço que atravessam a iso-superfície desejada.

Neste capítulo, apresentamos uma técnica de extração de iso-superfícies baseada na amostragem hierárquica do domínio espacial da função implícita. O algoritmo minimiza o número de avaliações de f e portanto permite uma eficiente aplicação do algoritmo *Marching Cubes*. A idéia está baseada na interpolação de uma distância pseudo-Euclidiana baseada numa cuidadosa eleição e posicionamento dos pontos de restrição.

Este capítulo esta organizado da seguinte forma: A Seção 5.2 descreve os métodos de poligonização usados na visualização de funções implícitas baseadas em RBFs. A Seção 5.3 expõe o algoritmo proposto e na seção 5.4 são apresentados os resultados experimentais.

5.2 Trabalhos relacionados

Uma vez que se tem uma função implícita, tal como a produzida pelo processo de interpolação baseado em RBFs, ela pode ser visualizada produzindo uma aproximação linear por partes de seu conjunto de nível zero. Este procedimento é conhecido como poligonização. Uma descrição detalhada dos métodos de poligonização está fora do alcance desta dissertação. Sugerimos ao leitor interessado em maiores detalhes revisar os seguintes trabalhos [5, 52].

Lorensen e Cline [29] apresentaram o algoritmo *Marching Cubes*, especificamente desenhado para a construção de iso-superfícies de dados médicos 3D. O princípio básico é reduzir o problema à triangulação de um simples cubo que intersecta à superfície. Os pontos de interseção são identificados ao longo das arestas do cubo em base a uma interpolação linear dos dados amostrados em cada vértice do cubo. Estes pontos são juntados em triângulos para formar um retalho da superfície poligonizada. Portanto, toda a iso-superfície pode ser triangulada fazendo “marchar” este cubo através dos dados, criando

deste modo os retalhos da superfície nos cubos intersectados [48].

Bloomenthal [8] apresenta um algoritmo de poligonização no qual a função implícita é amostrada adaptativamente, subdividindo o espaço com um esquema de particionamento baseado em *octrees*, que pode convergir à superfície (usando subdivisão dos cubos da *octree*) ou fazer um seguimento dela (por propagação de células). No final, as células terminais da *octree* são poligonizadas em tempo constante. Para resolver problemas de ambigüidade ele também fez uso de subdivisão adaptativa.

Além desses dois métodos básicos, muitas variantes foram propostas, por exemplo [48, 31]. Em essência, estas variantes tentam reduzir a taxa de amostragem assegurando-se que a superfície resultante seja geometricamente e topologicamente correta.

Em geral, qualquer poligonizador desenvolvido para funções implícitas gerais pode ser usado para poligonizar superfícies implícitas variacionais. Turk e O'Brien [50, 51], Karpenko [25] realizam a extração de iso-superfícies usando o método de continuação de Bloomenthal. Huong Quynh Dinh et al. [18, 19] extraem iso-superfícies usando o algoritmo *Marching Cubes* [29]. Carr et al. [12] usaram um método de continuação baseado no algoritmo *Marching Tetrahedra* [48].

Poligonizadores específicos para superfícies implícitas variacionais foram propostos por [16, 27, 23]. O algoritmo do Crespin [16] realiza uma tetraedralização incremental de Delaunay em base aos pontos de restrição. Este método é muito custoso e não apresenta bons resultados visuais.

Laga et al. [27] usam um esquema baseado em *octrees* para encontrar células perto dos pontos de restrição. A principal vantagem deste procedimento de classificação de células é não requerer a avaliação da função implícita. Uma vez que as células intersectantes são encontradas, elas são poligonizadas avaliando a função implícita em seus vértices. A principal desvantagem deste método é que não pode ser usado quando a densidade dos pontos é baixa e irregular.

Xiaogang et al. [23] apresentam um método que requer como dado de entrada uma malha poligonal grosseira que aproxima a iso-superfície desejada. Esta malha de controle é recursivamente subdividida até um nível especificado usando um esquema de subdivisão poliedral. Posto que os novos vértices adicionados não pertencem à superfície, eles são mapeados sobre a superfície usando o método de iteração de Newton. O algoritmo é eficiente e produz malhas poligonais de alta qualidade. A única desvantagem é que requer uma malha triangular para prover a informação de conectividade necessária.

5.3 Algoritmo de poligonização

O algoritmo de poligonização de superfícies implícitas baseadas em RBFs é constituído de três etapas principais:

1. Especificação dos pontos de restrição, baseada numa métrica pseudo-Euclidiana.
2. Construção da função interpolante.
3. Amostragem adaptativa do espaço, identificação dos cubos que intersectam a superfície e a sua subseqüente triangulação.

5.3.1 Especificação dos pontos de restrição

Uma função implícita baseada em RBFs é modelada através da especificação de um conjunto de pontos de restrição $\{c_1, c_2, ..., c_n\}$, junto com um conjunto de valores $\{v_1, v_2, ..., v_n\}$ para cada um desses pontos [56]. Os pontos de restrição se dividem em dois subconjuntos: os pontos de restrição de fronteira e os pontos de restrição de normais. Os pontos de restrição de fronteira são posições c_i que tomarão o valor $v_i = 0$, e pelos quais a superfície implícita criada passará exatamente. Os pontos de restrição de normais são as posições que definirão a orientação da superfície.

A escolha e localização dos pontos de restrição é fortemente dependente do problema a ser resolvido. Em aplicações de reconstrução, por exemplo, [56], o dado de entrada é, freqüentemente, uma malha poligonal cujos vértices q_i são usados como pontos de restrição de fronteira. Além disso, para cada vértice q_i, é estimado um vetor normal e gera-se um ponto de restrição de normal n_i, deslocando q_i uma pequena distância d ao longo desse vetor. Em resumo, para o conjunto de pontos de restrição, definem-se: $v_i = f(q_i) = 0$, ou $v_j = f(n_j) = w$. A Figura 5.1 ilustra esta idéia.

Por razões que ficarão mais claras posteriormente, desejaríamos assegurar a seguinte propriedade para a função interpolante:

$$|f(\mathbf{x})| \leq \delta(\mathbf{x}, S), \forall \mathbf{x} \in \Re^3, \tag{5.1}$$

onde S é a iso-superfície de nível zero de f e δ denota a métrica Euclidiana. Em outras palavras é desejável que $f(\mathbf{x})$ possa ser usada como um limite inferior para distância de $\mathbf{x}$ à superfície S.

Para a verificação dessa propriedade ajustamos os valores de w e d. Uma condição necessária é que $w < d$. Isto garante que se n_i é um ponto de restrição de normal, então

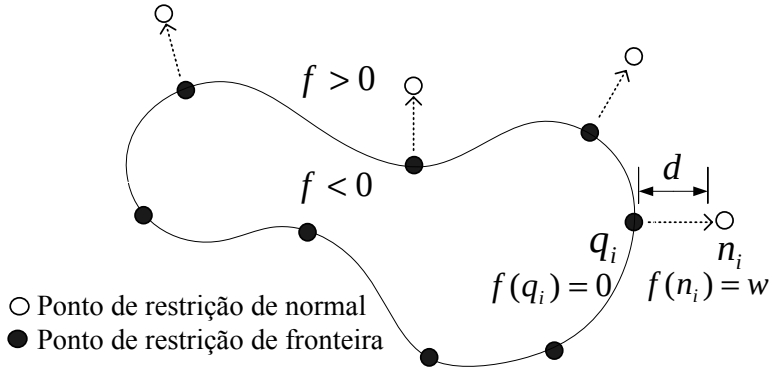


Figura 5.1: Os pontos de restrição de normais, n_i, são colocados ao longo de um vetor normal a uma distância d dos pontos de restrição de fronteira, q_i. A função f satisfaz $f(\mathbf{x}) < 0$ para todo **x** dentro da curva e $f(\mathbf{x}) > 0$ para **x** fora da curva.

$f(n_i) < d$. É razoável assumir que d é uma boa aproximação da distância Euclidiana entre n_i e S. Em nossa implementação, esta suposição é fortalecida, assegurando-nos que não existe outro vértice $q_j, j \neq i$ que está mais próximo de n_i [12]. Além disso, devemos ajustar o valor de w tal que seja significativamente menor que d, e desta forma garantir a propriedade (5.1), pelo menos numa vizinhança limitada de S.

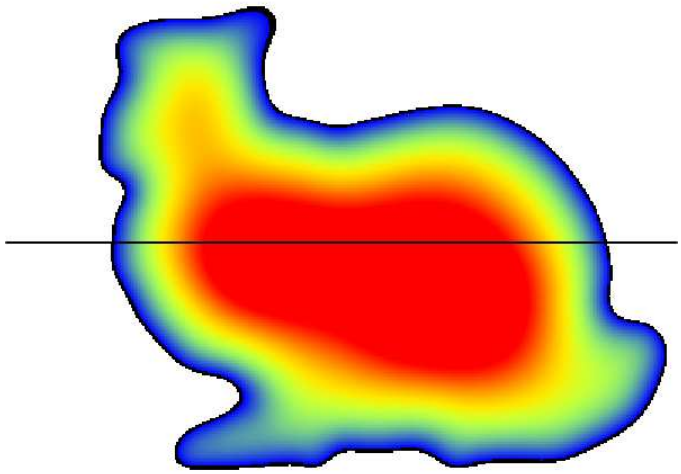
A Figura 5.2 ilustra como a função interpolante e a função da distância Euclidiana variam ao longo de uma linha que intersecta uma reconstrução do "*Stanford Bunny*". A Figura 5.2(a) representa um corte do modelo criado com base em uma malha poligonal constituída de 800 vértices, usando $d = 0.015$ e $w = 3d/4$. Devemos observar que o valor da função interpolante é menor ou igual à função de distância Euclidiana numa vizinhança do modelo (Fig. 5.2(b)). Entretanto, à medida que os pontos amostrados se afastam muito do modelo, a função interpolante tenderá a alcançar alcançará a função de distância Euclidiana (Fig. 5.2(c)).

Felizmente, nosso método de poligonização não requer que a propriedade (5.1) seja válida em qualquer lugar do espaço, mas apenas numa vizinhança limitada do modelo.

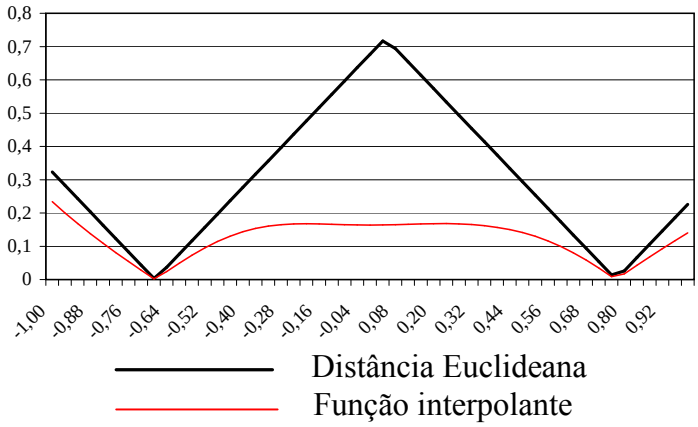
5.3.2 Construção da função interpolante

Uma vez que os pontos de restrição são determinados, o sistema linear de equações (3.11) é construído e resolvido.

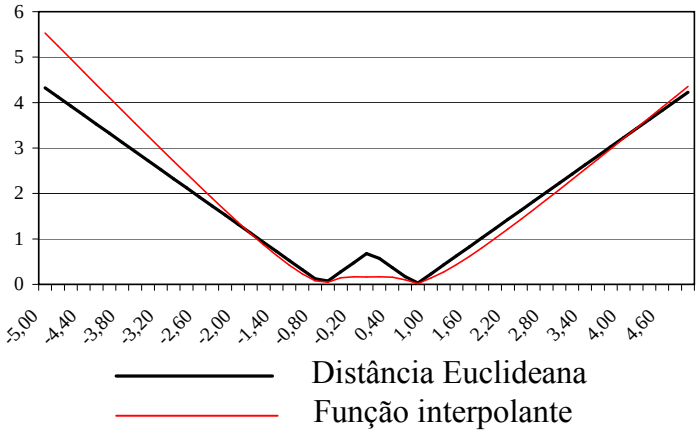
No entanto, deve-se notar que devido ao uso da *spline* triharmônica como função de base, a matriz resultante é densa e as entradas tendem a ter valores grandes em posições distantes da diagonal principal. Em particular, as entradas na diagonal principal têm valores iguais a zero. Tais matrizes mal condicionadas só podem ser resolvidas por



(a)



(b)



(c)

Figura 5.2: (a) Uma seção transversal da função interpolante do "*Stanford Bunny*". (b) Gráficos da função interpolante f e a métrica Euclidiana, avaliadas ao longo de um segmento de linha numa vizinhança do modelo. (c) *Zoom* da figura mostrada em (b).

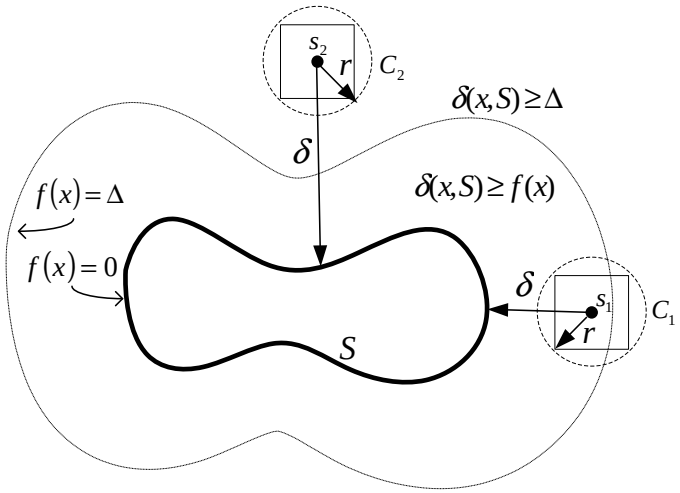


Figura 5.3: Uma célula C_1 "perto" de S pode ser rejeitada se $f(s_1) > r$ posto que neste caso $\delta(s_1, S) > r$. Uma célula C_2 "distante" de S satisfaz $f(s_2) > \Delta$, e desta forma, também satisfaz $\delta(s_2, S) > \Delta$. Portanto C_2 pode ser descartada já que $r < \Delta$.

métodos diretos e aproximados pagando um alto custo computacional. Recentemente, alguns avanços matemáticos propostos ajudam a lidar com este problema (Veja a discussão desses métodos na seção 3.1.3).

5.3.3 Amostragem adaptativa do espaço

Nesta etapa o espaço é amostrado usando uma grade de refinamento progressivo. A grade se faz mais fina para encontrar as células que intersectam a superfície. A idéia é refinar uma célula da grade somente se ela contém uma parte da superfície. Ao final, obtemos um conjunto de células de igual tamanho, não sobrepostas, que garantidamente intersectam a superfície.

Seja C uma célula cúbica da grade. Para determinar se C intersecta a superfície, avaliamos a função interpolante no centro s do cubo. Se r é o raio da menor esfera que contém C, então a iso-superfície S intersecta C somente se $\delta(s, S) \leq r$.

Consideremos válida a seguinte implicação:

$$|f(\mathbf{x})| \leq \Delta \Rightarrow \delta(\mathbf{x}, S) \geq |f(\mathbf{x})|, \tag{5.2}$$

onde Δ é uma constante e $\mathbf{x} \in \Re^3$ é um ponto da vizinhança de S.

Portanto, enquanto s se encontra dentro dessa vizinhança (isto é a célula está "perto" de S), podemos substituir $\delta(s, S)$ por $f(s)$ em nosso teste de distância (veja a Figura 5.3). Em outras palavras, uma célula C não necessita ser subdividida se $r < |f(s)| \leq \Delta$.

Agora, vamos considerar uma célula C que está relativamente "distante" de S, especificamente, uma célula cujo centro s é tal que $|f(s)| > \Delta$. Neste caso, estamos assumindo que a distância entre s e S é maior que Δ, posto que todos os pontos p que estão numa distância menor que Δ a S necessariamente satisfazem $|f(\mathbf{p})| < \Delta$. Assim, podemos assumir

$$|f(\mathbf{x})| > \Delta \Rightarrow \delta(\mathbf{x}, S) > \Delta. \tag{5.3}$$

Podemos tomar vantagem disto assegurando-nos que o raio de C nunca seja maior que Δ, posto que neste caso $|f(s)| > \Delta$, o que implicaria $\delta(\mathbf{x}, S) > r$, significando que a célula "distante" não intersecta S (veja Fig. 5.3). Esta hipótese permite-nos cobrir ambos os casos (células "perto" ou "afastadas" de S) com um simples teste de rejeição:

$$|f(s)| > r \Rightarrow C \text{ não intersecta } S. \tag{5.4}$$

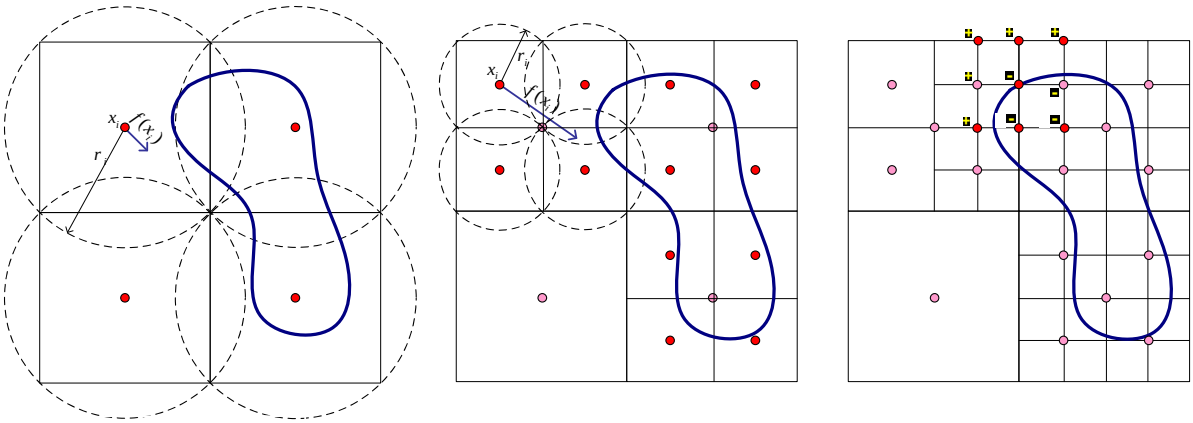
Resta ser discutida uma última questão: como é estabelecido o valor de Δ? Na prática, não há necessidade de calcular um valor fixo para Δ, basta assegurar-nos que seja convenientemente grande, isto é, que o raio do maior cubo usado no processo de poligonização nunca seja maior que Δ. Isto é obtido usando uma razão entre w e d suficientemente pequena (ver Fig. 5.6). No entanto, rações muito pequenas terão um efeito perjudicial na performance do algoritmo. Para ver isto, note que neste caso, $f(\mathbf{x})$ retornará estimativas muito pequenas de $\delta(\mathbf{x}, S)$, fazendo com que muitas células sejam subdivididas sem necessidade. Foram realizados vários experimentos que indicam (veja Seção 5.4) que $w/d = 3/4$ é uma boa proporção sem incorrer em custos que afetam a performance do algoritmo.

O algoritmo de amostragem do espaço usado na nossa implementação é resumido no seguinte pseudo-código:

procedure *SpaceSample*(f, C, *maxlevel*, *level*)

1. **If** (*level* $==$ *maxlevel*) **then**
   (a) Avaliar f naqueles vértices do cubo C que ainda não foram avaliados;
   (b) *MCcellPolygonize*(C);
2. **Else If** (*Straddle*(f, C)) **then**
   (a) Subdividir C em 8 sub-cubos C_i de igual tamanho;
   (b) **For** $i = 0..7$ **do**
   SpaceSample(f, C_i, *maxlevel*, *level*+1);

A Função *Straddle* efetua o teste de rejeição da equação (5.4), retorna verdadeiro se C intersecta a superfície e falso caso contrário. Células folha – correspondentes ao



(a) (b) (c)

Figura 5.4: (a) e (b) Subdivisão hierárquica do domínio espacial da função implícita empregando o teste de rejeição (5.4) para células afastadas. (c) Para células folha da subdivisão, o teste de rejeição é feito examinando os sinais da função nos vértices da célula.

máximo nível de subdivisão *maxlevel* –, são passadas à função *MCcellPolygonize*, que realiza o processo de triangulação [29]. O teste de rejeição nestas células é implicitamente realizado pelo teste padrão do algoritmo *Marching Cubes*, isto é, examinando os sinais da função f nos vértices dos cubos. Veja uma ilustração do processo na Figura 5.4.

É importante destacar que o algoritmo proposto efetua uma eficiente amostragem de f, sendo mais fina somente nas proximidades da superfície interpolada. A Figura 5.5 ilustra esta idéia.

Uma consideração crucial na implementação deste algoritmo é evitar reavaliar a função implícita nos vértices que já foram visitados. Por exemplo, a cláusula "if" no passo 2 avalia a função no centro do cubo C; este valor pode corresponder posteriormente a um vértice de um cubo no passo $1.(a)$. Nossa implementação emprega um *cache* para os vértices avaliados de forma a garantir que f seja calculada somente uma vez para cada ponto no espaço.

5.4 Experimentos

Os experimentos foram computados num PC equipado com um processador AMD-Duron correndo a 1.3 GHz e 256 MB de memória principal.

Todos os modelos foram simplificados a 800 vértices e colocados dentro de um espaço cúbico com (-1,-1,-1) e (1,1,1) como pontos mínimo e máximo, respectivamente.

Em nossa implementação o espaço cúbico, de tamanho $s \times s \times s$, é uniformemente

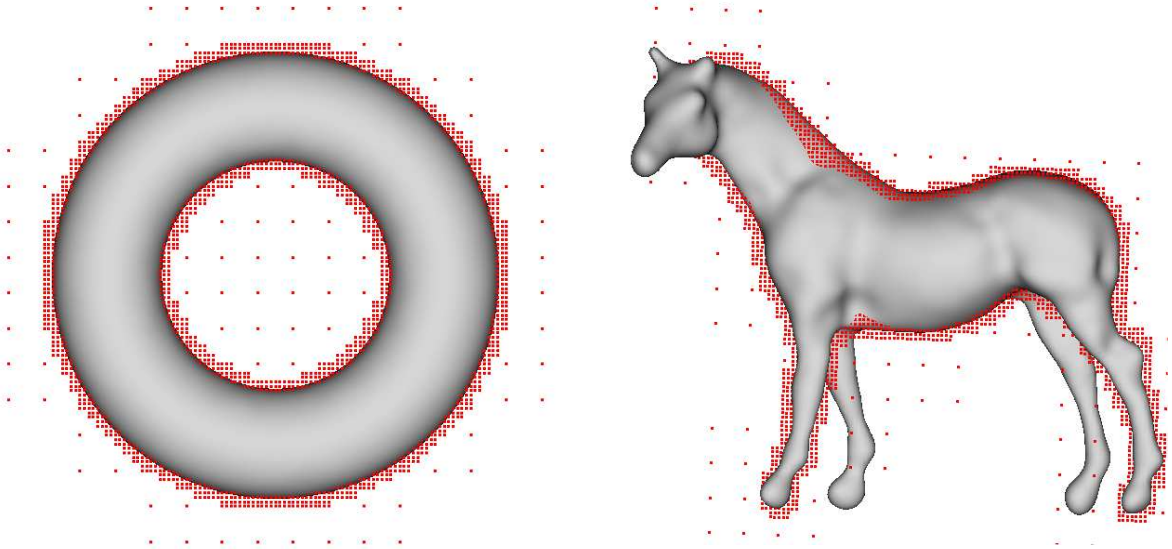


Figura 5.5: Ilustração do amostragem hierárquico da função interpolante. Os pontos vermelhos representam os pontos (sobre um mesmo plano) amostrados na poligonização dos modelos.

dividido numa grade 3D de cubos idênticos, onde s é uma potência de dois.

Primeiramente conduzimos experimentos visando encontrar valores "ótimos" para o raio máximo r e para a razão w/d. Para este propósito, realizamos a poligonização do "*Stanford Bunny*" usando seis valores de w/d (0.2, 0.5, 0.667, 0.75, 0.8 e 0.83) e seis valores para r. Posto que nós usamos um esquema de decomposição do tipo *octree*, os seis valores de r correspondem aos raios dos octantes dos consecutivos níveis de refinamento do espaço cúbico inicial, i.e., $\sqrt{3}/2$, $\sqrt{3}/4$, e assim por diante. Na Tabela 5.1 observamos a relação direta entre diferentes valores de r e w/d. Uma pequena proporção w/d implica um maior tamanho de Δ, fazendo possível usar grandes tamanhos iniciais para r com resultados corretos da poligonização. Porém, grandes valores de Δ têm um efeito prejudicial na performance do algoritmo. É importante observar que para alguns valores w e d o algoritmo rejeita algumas células que intersectavam a superfície, produzindo alguns buracos no modelo poligonizado. Exemplos deste comportamento são apresentados na Figura 5.7.

A Figura 5.6 ilustra como o valor de Δ depende da razão w/d. Lembre-se que Δ denota o "tamanho" da região onde f pode ser usada como um limite inferior para a métrica Euclidiana.

Em seguida, realizamos outros experimentos para testar a eficiência do algoritmo na poligonização de diferentes funções de reconstrução usando os valores "ótimos" w e d obtidos. Veja os resultados no lado esquerdo da tabela 5.2. A Figura 5.8 apresenta al-

	$r = \frac{\sqrt{3}}{2}$	$r = \frac{\sqrt{3}}{4}$	$r = \frac{\sqrt{3}}{8}$	$r = \frac{\sqrt{3}}{16}$	$r = \frac{\sqrt{3}}{32}$	$r = \frac{\sqrt{3}}{64}$
$\frac{w}{d}=\frac{1}{5}$	26.38% ok	26.38% ok	26.38% ok	26.36% ok	26.68% ok	32.78% ok
$\frac{w}{d}=\frac{1}{2}$	12.36% ok	12.36% ok	12.36% ok	12.41% ok	13.26% ok	21.94% ok
$\frac{w}{d}=\frac{2}{3}$	9.89% ok	9.89% ok	9.89% ok	9.97% ok	10.93% ok	20.09% ok
$\frac{w}{d}=\frac{3}{4}$	9.01% ok	9.01% ok	9.01% ok	9.10% ok	10.10% ok	19.44% ok
$\frac{w}{d}=\frac{4}{5}$	8.57% hole	8.57% hole	8.58% hole	8.67% hole	9.69% ok	19.12% ok
$\frac{w}{d}=\frac{5}{6}$	8.29% hole	8.29% hole	8.30% hole	8.39% hole	9.42% hole	18.90% ok

Tabela 5.1: Esta tabela mostra resultados do algoritmo proposto na poligonização da função de reconstrução do "*Stanford Bunny*" modelada com diferentes valores de w/d. O valor de d foi ajustado a 0.015 em todos os testes. Os valores em porcentagem correspondem ao número total de avaliações da função requeridas pelo algoritmo comparadas com o número de avaliações computadas pelo método *Marching Cubes* padrão. Um valor "*hole*" significa que a poligonização resultante é incorreta, isto é, o algoritmo rejeitou uma ou mais células que intersectam a superfície.

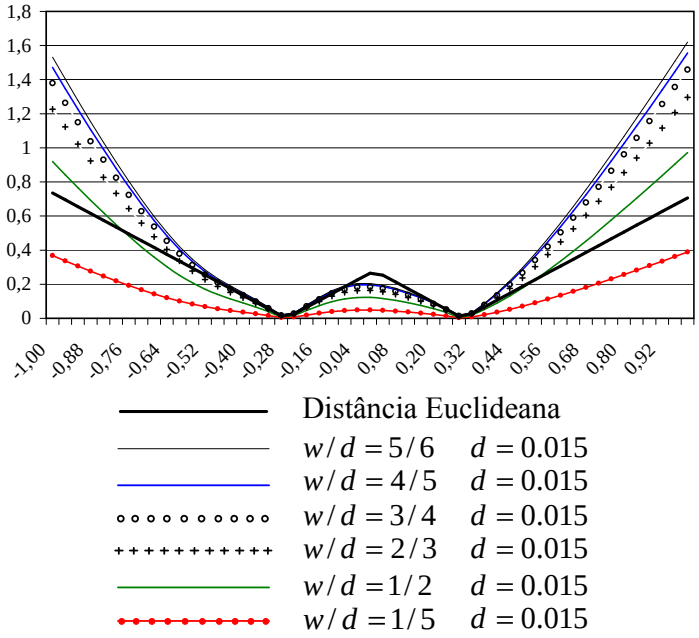


Figura 5.6: Gráfico de varias funções interpolantes usadas nos experimentos (veja Tabela 5.1), avaliadas no segmento de linha definida por (-1,-1,-1) e (1,1,1).

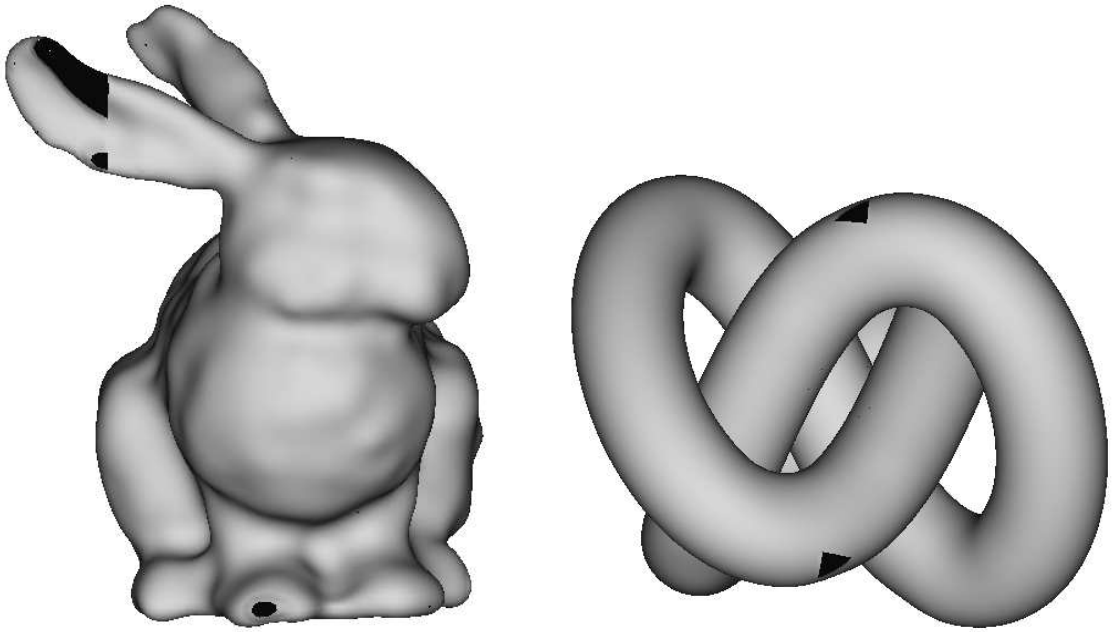


Figura 5.7: Poligonização com resultados incorretos, isto é, buracos na triangulação resultante. As interpolações para os dois modelos foi realizada usando $w = d = 0.015$.

gumas imagens obtidas nestes experimentos. Todas as funções foram construídas usando como entrada uma malha poligonal de 800 vértices, usando $d = 0.015$ e $w/d = 3/4$. A decomposição LU para resolver o sistema de equações demorou 42 segundos em todos os casos. O cubo inicial foi inicialmente subdividido em $128 \times 128 \times 128$ células.

Finalmente alguns experimentos foram realizados para comparar o algoritmo proposto com o algoritmo de poligonização de Bloomenthal [9]. Para obter uma comparação "justa", modificou-se a implementação de Bloomenthal de modo que a sua função de cálculo de vértices empregue interpolação linear ao invés de subdivisão binária. Além disso, o algoritmo foi configurado para usar decomposição cúbica no lugar da decomposição tetraedral que vem configurada por *default*. Também, o ponto "semente" foi estabelecido para ser um vértice da malha poligonal. Em ambas implementações, o tamanho dos cubos correspondem a uma decomposição de $128 \times 128 \times 128$ do espaço do mundo. Os resultados das comparações são mostrados na tabela 5.2.

Os experimentos indicam que a técnica proposta tende a funcionar mais rápido que a implementação de Bloomenthal [9]. Como podia-se esperar, o número de triângulos para ambos os métodos são quase idênticos. A ligeira variação é devida ao fato de que as decomposições cúbicas não são as mesmas, posto que o ponto "semente" no algoritmo de Bloomenthal determina a origem da decomposição. Este ganho em performance pode ser atribuído ao menor número de avaliações realizadas pela técnica proposta.

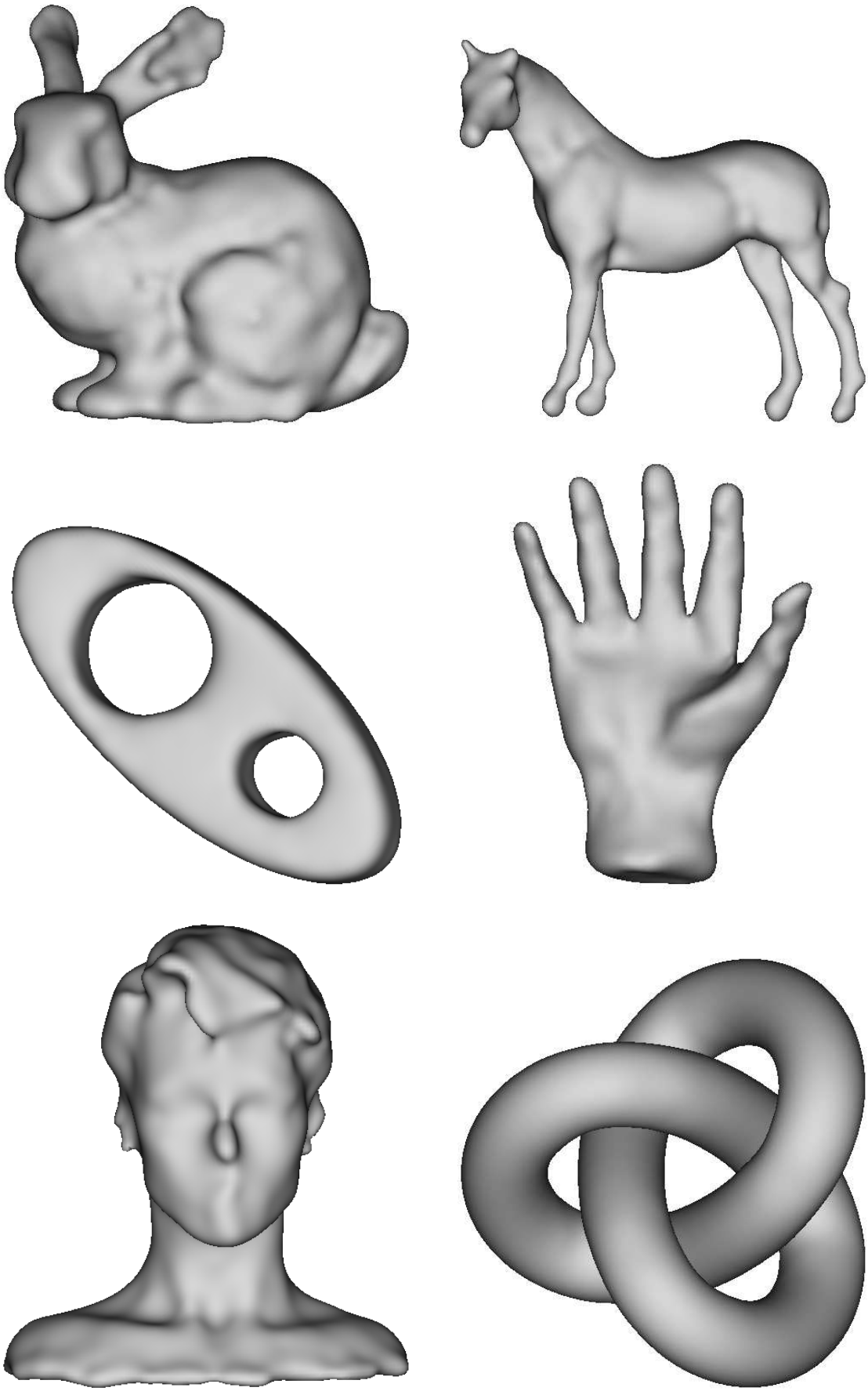


Figura 5.8: Diferentes modelos poligonizados com o algoritmo proposto.

Modelo	Pontos de restrição (número)	Algoritmo proposto				Algoritmo de Bloomenthal		
		Extração Iso-superfície	Avaliações (número)	Triângulos (número)	Avaliações (%)	Extração Iso-superfície	Avaliações (número)	Triângulos (número)
Bunny	1600	58s	193395	81340	9.01%	75s	244171	81420
Horse	1600	34s	111560	47028	5.20%	43s	141128	47088
Torus2	1600	27s	90654	33932	4.22%	31s	102867	34304
Hand	1599	26s	83202	34660	3.87%	32s	104085	34720
Head	1600	53s	176270	71528	8.21%	65s	214441	71508
Knot	1600	81s	269701	114704	12.56%	104s	343725	114592

Tabela 5.2: Comparação entre a técnica proposta e o algoritmo de Bloomenthal.

Nós também testamos consistência topológica, com resultados positivos, nas poligonizações geradas. Deve-se notar, porém, que a técnica proposta está baseada no algoritmo *Marching Cubes* [29] e portanto herda todas as suas desvantagens tais como a criação de um excessivo número de triângulos e alguns problemas de ambigüidade em baixas resoluções.

Capítulo 6

Conclusões e trabalhos futuros

Apresentamos diversos algoritmos que permitem a criação rápida de modelos de aparência *livre* 3D fazendo uso de uma interface baseada em traços. Cobrimos algumas limitações dos trabalhos anteriormente propostos [22, 25, 37, 3]. Apresentamos a operação de combinação como a principal operação de edição, sendo que a furação e a extrusão são implementados fazendo uso desta operação.

Propomos uma operação de criação de modelos a partir de uma malha poligonal grosseira construída a partir de um traço 2D e que tem uma distribuição quase regular de seus vértices. Esta técnica permite evitar problemas de vazamento de superfícies.

Devido ao uso de RBFs como esquema de representação das superfícies dos objetos modelados, são imprescindíveis o posicionamento correto dos pontos de restrição e a especificação cuidadosa dos seus valores de função após a edição dos modelos, sob pena de se obter resultados imprevisíveis.

A presente proposta ainda comporta diversas extensões:

- A operação de furação pode ser melhorada, tanto no suporte de buracos a serem feitos sobre qualquer posição dos objetos quanto na forma de especificação dos pontos de restrição que o definem.

- O suporte de vincos e pontas (produto da aplicação de operações de corte e furação, por exemplo), pode ser encarado fazendo uso das RBFs paramétricas, reportadas recentemente por Laga et al. [27].

- A grande quantidade de pontos de restrição requerida para o suporte de pequenos detalhes nos modelos implica necessariamente a inclusão de técnicas aproximadas para a construção da função interpolante [7, 6, 12].

- Uma vez que o protótipo apresentado foi concebido para a criação de cenas simples, a experimentação de uma interface sugestiva [21] como mecanismo de mediação frente a ambigüidades se mostrou de razoável utilidade. A sua aplicabilidade em sistemas de grande porte deve ser necessariamente avaliada.

- Por último, a combinação de modelos usando traços guias [25], resulta numa operação candidata a ser suportada numa futura evolução do sistema.

Por outro lado, apresentamos um algoritmo para a poligonização rápida de superfícies implícitas variacionais. Esta técnica minimiza o número das custosas avaliações da função implícita usando uma amostragem hierárquica do espaço, permitindo deste modo uma eficiente aplicação do algoritmo *Marching Cubes*.

A técnica proposta pode também ser aplicada à poligonização de outras classes de objetos implícitos, desde que suas funções se comportem como limites inferiores para a métrica Euclidiana dentro de uma dada vizinhança (veja a discussão na Seção 5.3.3). Uma observação importante é que esta condição é consideravelmente mais fraca que o bem conhecido critério de exclusão de Lipschitz [24].

Adicionalmente, deve ser possível adaptar as idéias apresentadas nesta dissertação a outras técnicas de visualização. Por exemplo, o *ray tracing* de superfícies implícitas pode ser eficientemente realizado acoplando o critério de rejeição proposto numa técnica acelerada baseada em octrees.

Para concluir, desejaríamos explorar esta técnica no contexto do método FastRBF [12], no esquema de poligonização adaptativa e o suporte de pontas e vincos.

Referências Bibliográficas

[1] An algorithm for piecewise linear approximation of implicitly defined two-dimensional surfaces. E. l. allgower and s. gnutzmann. *SIAM Journal of Numerical Analysis*, 24(2):2452–2469, 1987.

[2] C. Alvarado and R. Davis. Resolving ambiguities to create a natural computer-based sketching environment. In *Proceedings of IJCAI-2001*, pages 1365–1371, 2001.

[3] B. Araujo and J. Jorge. Blobmaker: Free-form modeling with variational implicit surfaces. In *12o Encontro Português de Computação Gráfica (EPCG)*, October 2003.

[4] E. Azevedo and A. Conci. *Computação Gráfica*. Editora Elsevier Ltda., Rio de Janeiro, 2003.

[5] C. Bajaj, J. Blinn, J. Bloomenthal, M. Cani-Gascuel, A. Rockwood, B. Wyvill, and G. Wyvill. *Introduction to Implicit Surfaces*. Morgan Kaufmann Publishers, INC., San Francisco, California, 1997.

[6] R. K. Beatson, J. B. Cherrie, and D. L. Ragozin. Fast evaluation of radial basis functions: Methods for four-dimensional polyharmonic splines. *SIAM J. Math. Anal.*, 32(6):1272–1310, 2001.

[7] R. K. Beatson and W. A. Light. Fast evaluation of radial basis functions: Methods for two-dimensional polyharmonic splines. *IMA Journal of Numerical Analysis*, (17):343–372, 1997.

[8] J. Bloomenthal. Polygonization of implicit surfaces. *Computer Aided Geometric Design*, pages 341–335, November 1988.

[9] J. Bloomenthal. An implicit surface polygonizer. *Graphics Gems IV*, pages 324–349, 1994.

[10] M. D. Buhmann. *Radial Basis Functions: Theory and Implementations*. Cambridge University Press, 2003.

[11] M.D. Buhmann. A new class of radial basis functions with compact support. *Mathematics of Computation*, 70(233):307–318, June 2001.

[12] J. C. Carr, R. K. Beatson, J. B. Cherrie, T. J. Mitchell, W. R. Fright, B. C. McCallum, and T. R. Evans. Reconstruction and representation of 3D objects with radial basis functions. In *Proceedings of SIGGRAPH 2001*, pages 67–76. ACM Press, 2001.

[13] J. C. Carr, R. K. Beatson, B. C. McCallum, W. R. Fright, T. J. McLennan, and T. J. Mitchell. Smooth surface reconstruction from noisy range data. In *Proceedings of ACM Graphite 2003*, pages 119–126. ACM Press, 2003.

[14] J. M. Cohen, F. Hughes, and R. C. Zeleznik. Harold: A world made of drawings. In *Proceedings of 1st international symposium on Non-photorealistic animation and rendering*, pages 83–90. ACM Press, 2000.

[15] J. M. Cohen, L. Markosian, R. C. Zeleznik, J. F. Hughes, and R. Barzel. An interface for sketching 3D curves. In *Proceedings of Symposium on Interactive 3D Graphics*, pages 17–21. ACM Press, 1999.

[16] B. Crespin. Dynamic triangulation of variational implicit surfaces using incremental delaunay tetrahedralization. In *Proceedings of the 2002 IEEE Symposium on Volume Visualization and Graphics*, pages 73–80, Piscataway, NJ, 28-29 2002. IEEE Computer Society.

[17] H. Q. Dinh. *Implicit Shapes: Reconstruction and Explicit Transformation*. PhD thesis, College of Computing. Georgia Institute of Technology, August 2002.

[18] H. Q. Dinh, G. Turk, and G. Slabaugh. Reconstructing surfaces using anisotropic basis functions. In *Proceedings of the Eighth International Conference On Computer Vision*, pages 606–613, Los Alamitos, CA, July 2001. IEEE Computer Society.

[19] H. Q. Dinh, G. Turk, and G. Slabaugh. Reconstructing surfaces by volumetric regularization using radial basis functions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(10):1358–1371, October 2002.

[20] T. Igarashi. Freeform user interfaces for graphical computing. In *Proceedings of 3rd International Symposium on Smart Graphics*, pages 39–48. Springer, July 2003.

[21] T. Igarashi and J.F. Hughes. A suggestive interface for 3D drawing. In *Proceedings of ACM UIST'01*, 14th Annual Symposium on User Interface Software and Technology, pages 173–181. ACM Press, 2001.

[22] T. Igarashi, S. Matsuoka, and H. Tanaka. Teddy: A sketching interface for 3D freeform design. In *Proceedings of SIGGRAPH 99*, Annual Conference Series, pages 409–416. ACM Press, 1999.

[23] X. Jim, H. Sun, and Q. Peng. Subdivision interpolating implicit surfaces. *Computer Graphics*, pages 763–772, October 2003.

[24] D. Kalra and A. Barr. Guaranteed ray intersections with implicit surfaces. *Computer Graphics*, pages 297–306, July 1989.

[25] O. Karpenko, J. F. Hughes, and R. Raskar. Free-form sketching with variational implicit surfaces. *Computer Graphics Forum*, September 2002.

[26] N. Kojekine. *Computer Graphics and Computer Aided Geometric Design by means of Compactly Supported Radial Basis Functions*. PhD thesis, Tokyo Institute of Technology, 2003.

[27] H. Laga, R. Piperakis, H. Takahashi, and M. Nakajima. A radial basis function based approach for 3D object modeling and reconstruction. In *Proceedings of IWAIT2003*, pages 139–144, 2003.

[28] S. K. Lodha and R. Franke. Scattered data techniques for surfaces. In *Proceedings of Dagstuhl Conference on Scientific Visualization*, pages 182–222, Dagstuhl-Germany, June 1999. IEEE Computer Society Press.

[29] W. Lorensen and H. Cline. Marching cubes: a high resolution 3D surface construction algorithm. *Computer Graphics*, 21(4):163–169, July 1987.

[30] L. Markosian, J. M. Cohen, T. Crulli, and J. Hughes. Skin: a constructive approach to modeling free-form shapes. In *Proceedings of SIGGRAPH 99*, Annual Conference Series, pages 393–400. ACM Press, 1999.

[31] S. V. Matveyev. Approximation of isosurface in the marching cube: Ambiguity problem. In *Proceedings IEEE Visualization*, pages 288–292. IEEE Computer Society, October 1994.

[32] T. Metin, T. Stahovich, and R. Davis. Sketch based interfaces: Early processing for sketch understanding. In *Proceedings of Perceptive User Interfaces Workshop 2001*, 2001.

[33] B. S. Morse, T. S. Yoo, P. Rheingans, D. T. Chen, and K. R. Subramanian. Interpolating implicit surfaces from scattered surface data using compactly supported radial basis functions. In *Proceedings of International Conference on Shape Modeling International*, pages 89–98, Genova, Italy, May 2001.

[34] G. M. Nielson. Scattered data modeling. *Computer Graphics and Applications, IEEE*, 13(1):60–70, January 1993.

[35] Y. Ohtake, A. Belyaev, M. Alexa, G. Turk, and H.P. Seidel. Multi-level partition of unity implicits. *ACM Transactions on Graphics*, 22(3):463–470, July 2003.

[36] Y. Ohtake, A. G. Belyaev, and H. P. Seidel. A multi-scale approach to 3D scattered data interpolation with compactly supported basis function. In *Shape Modeling International*, pages 153–164. IEEE Computer Society, 2003.

[37] S. Owada, F. Nielsen, K. Nakazawa, and T. Igarashi. A sketching interface for modeling the internal structures of 3D shapes. In *Proceedings of 3rd International Symposium on Smart Graphics*, pages 49–57. Springer, July 2003.

[38] A. Pasko, V. Adzhiev, A. Sourin, and V. Savchenko. Function representation in geometric modeling: concepts, implementations and applications. *The Visual Computer*, 11(8):429–446, 1995.

[39] A. Pasko and V. Savchenko. Constructing functionnally-defined surfaces. In *Implicit Surfaces '95*, 1995.

[40] P. Reuter, I. Tobor, C. Schlick, and S. Dedieu. Point-based modelling and rendering using radial basis functions. In *Proceedings of ACM Graphite*, pages 111–118, Held in Melbourne, Australia, 2003. ACM Press.

[41] R. J. Rost. A 3D object file format. http://www.dcs.ed.ac.uk/home/mxr/gfx/3d/off.spec.

[42] V. Savchenko, A. Pasko, O. Okunev, and T. Kunii. Function representation of solids reconstructed from scattered surface points and contours. *Computer Graphics Forum*, 14(4):181–188, 1995.

[43] C. Tai, H. Zhang, and C. Fong. Prototype modeling from sketched silhouettes based on convolution surfaces. *Computer Graphics Forum*, 23(1):71–83, 2004.

[44] FarField Technology. http://www.farfieldtechnology.com.

[45] I. Tobor, P. Reuter, and C. Schlick. Efficient reconstruction of large scattered geometric datasets using the partition of unity and radial basis functions. *Journal of WSCG*, 12(1-3):467–474, February 2004.

[46] O. Tolba, J. Dorsey, and L. McMillan. Sketching with projective 2D strokes. In *Proceedings of the 12th annual ACM symposium on User interface software and technology*, pages 149–157. ACM Press, 1999.

[47] O. Tolba, J. Dorsey, and L. McMillan. A projective drawing system. In *Proceedings of 2001 ACM Symposium on Interactive 3D Graphics*, pages 25–34. ACM Press, 2001.

[48] G. M. Treece, R. W. Prager, and A. H. Gee. Regularised marching tetrahedra: improved iso-surface extraction. *Computers and Graphics*, 23(4):583–598, 1999.

[49] G. Turk and J. O'Brien. Shape transformation using variational implicit functions. In Alyn Rockwood, editor, *Proceedings of SIGGRAPH 99*, pages 335–342. Addison Wesley Longman, August 1999. ISBN 0-20148-560-5. Held in Los Angeles, California.

[50] G. Turk and J. O'Brien. Variational implicit surfaces. Technical report, Georgia Institute of Technology, May 1999.

[51] G. Turk and J. O'Brien. Modelling with implicit surfaces that interpolate. *ACM Transactions on Graphics*, pages 855 – 873, October 2002.

[52] L. Velho, J. Gomes, and L. H. Figueiredo. *Implicit Objects in Computer Graphics*. Springer Verlag, New York, 2002.

[53] H. Wendland. Piecewise polynomial, positive definite and compactly supported radial basis functions of minimal degree. *Advances in Computational Mathematics*, (4):389–396, 1995.

[54] A. P. Witkin and P. S. Heckbert. Using particles to sample and control implicit surfaces. In *Proceedings of SIGGRAPH 94*, Annual Conference Series, pages 269–277, 1994.

[55] B. Wyvill, E. Galin, and A. Guy. Extending The CSG Tree. Warping, Blending and Boolean Operations in an Implicit Surface Modeling System. *Computer Graphics Forum*, 18(2):149–158, June 1999.

[56] G. Yngve and G. Turk. Robust creation of implicit surfaces from polygonal meshes. *IEEE Transactions on Visualization and Computer Graphics*, pages 335–342, December 2002. ISBN 0-20148-560-5. Held in Los Angeles, California.

[57] R. C. Zeleznik, K. P. Herndon, and J. F. Hughes. SKETCH: An interface for sketching 3D scenes. In *SIGGRAPH 96 Conference Proceedings*, pages 163–170. Addison Wesley, 1996.