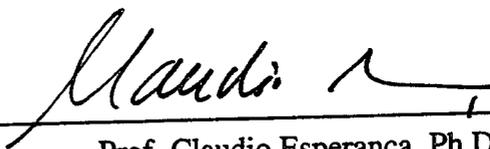


MODELAGEM E DEFORMAÇÃO DE SUPERFÍCIES BASEADAS EM TRAÇOS

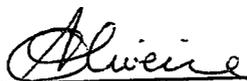
Disney Douglas de Lima Oliveira

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 DOUTOR EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

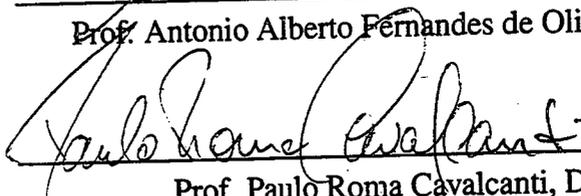
Aprovada por:



Prof. Claudio Esperança, Ph.D.



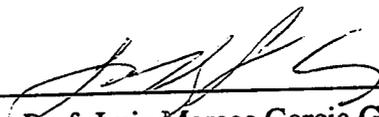
Prof. Antonio Alberto Fernandes de Oliveira, Dsc.



Prof. Paulo Roma Cavalcanti, Dsc.



Prof. André Luiz Pires Guedes, Dsc.



Prof. Luiz Marcos Garcia Gonçalves, Dsc.



Prof. Cícero Augusto Mota Cavalcante, Dsc.

RIO DE JANEIRO, RJ - BRASIL

MARÇO DE 2008

OLIVEIRA, DISNEY DOUGLAS DE LIMA

Modelagem e deformação de superfícies baseadas em traços [Rio de Janeiro] 2008

XVIII, 140 p., 29,7 cm, (COPPE/UFRJ, D.Sc., Engenharia de Sistemas e Computação, 2008)

Tese – Universidade Federal do Rio de Janeiro, COPPE

1. Modelagem a mão livre
 2. Geometria diferencial discreta
 3. Funções de bases radiais
- I. COPPE/UFRJ II. Título (série)

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

MODELAGEM E DEFORMAÇÃO DE SUPERFÍCIES BASEADAS EM TRAÇOS

Disney Douglas de Lima Oliveira

Março/2008

Orientador: Claudio Esperança

Programa: Engenharia de Sistemas e Computação

Este trabalho aborda técnicas de modelagem e edição de malhas triangulares para representação de modelos tridimensionais. Estas incluem técnicas de deformação do espaço usando funções de base radiais (RBFs), e processamento de malhas baseado no operador Laplaciano, realizando comparações entre os métodos através da manipulação à mão livre da malha poligonal.

Apresentamos uma ferramenta para criação e/ou deformação de modelos 3D à mão livre através de traços. O usuário pode explorar diferentes opções de deformação alterando a suavidade continuamente entre C^0 e C^2 , onde a variação da suavidade na deformação do espaço é obtida por meio da mistura de splines poli-harmônicas nas RBFs, sendo possível a comparação visual dos resultados obtidos por transformações realizadas segundo os dois métodos. Deste modo, a criação e deformação de modelos tridimensionais pode ser realizada de forma fácil, rápida e intuitiva. O usuário pode criar um modelo simples a partir de um traço e depois realizar operações de edição tais como a criação de características afiadas e características suaves na superfície do modelo.

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

STROKE BASED SURFACE MODELLING AND DEFORMATION

Disney Douglas de Lima Oliveira

Março/2008

Advisor: Claudio Esperança

Department: Computing and Systems Engineering

This work discusses techniques for approaches modeling and editing triangular meshes for three-dimensional representation of models. We apply two spatial deformation techniques: one using Radial Basis Functions (RBFs) and another based on the discretized Laplacian operator. Comparisons between the approaches are carried out with free hand manipulation of the polygonal mesh.

We present a tool for creation and/or deformation of 3D models based on free hand strokes. The user can explore different deformation options altering the smoothness continuously between C^0 and C^2 . The smoothness variation of the spatial deformation is obtained by blending poly-harmonic splines in the RBFs, allowing a visual comparison of the results using the two approaches. In this way, the creation and deformation of three-dimensional models can be made intuitively, quickly, and easily. The user creates a simple model from a stroke and then carries out edition operations such as the creation of sharp and smooth features on the model's surface.

*À minha esposa Genilce,
pelo apoio, paciência e dedicação.*

Agradecimentos

Primeiramente gostaria de agradecer ao meu orientador Claudio Esperança pela confiança e apoio a mim depositados, pelas críticas construtivas, principalmente na definição e implementação do tema de tese.

Ao professor Antônio Oliveira por seus conselhos valiosos e seu vasto conhecimento tanto em computação gráfica quanto em matemática, sendo o nosso guru do LCG. Ao professor Paulo Roma pelas discussões e suporte na instalação das bibliotecas que precisei no Fedora e por fazer meu aplicativo funcionar em todas as máquinas do LCG. Ao professor Cícero Cavalcante por ter me incentivado e orientado no período em que fiquei em Manaus, aos professores André Guedes e Luiz Gonçalves membros externos da banca, por terem aceito o convite e pelo tempo dedicado.

Aos amigos que conquistei nesses anos em que fiquei no LCG, em especial ao Álvaro Cuno que foi fundamental na conclusão deste trabalho, principalmente com relação a fase de implementação, na qual dedicou várias horas de seu precioso tempo me ajudando a debugar e programar. Ao Ricardo Marroquim que além de me ajudar com dúvidas e sugestões, também me hospedou em sua casa nessa fase final. A todos aqueles que também me ajudaram de forma direta ou indireta como o Saulo Ribeiro, Yalmar Atencio, Guina Alzamora, André Máximo, Antonio Lopes, Cesar Xavier, Margareth Catoia, Ricardo Farias, Carlos Henrique e Karl.

Ao PESC, professores e técnico-administrativos, em especial, à Sonia, Mercedes, Cláudia, Lúcia, Solange, Sueli, Loudes e Dona Deda, por serem sempre solícitas comigo durante todo este tempo.

À Universidade Federal do Amazonas por ter me liberado de minhas atividades docentes pelo período de quatro anos, à CAPES pela bolsa PICDT, e aos professores do Departamento de Matemática da UFAM, em especial aos professores Roberto Silva, Flávia Jacinto, Nilomar Oliveira e Valtemir Cabral com quem tive a oportunidade de conviver aqui na COPPE, e ao professor Domingos Anselmo por ter me substituído nas disciplinas do DM em que fiquei impossibilitado de ministrar.

Aos amigos que convivi na COPPE, em especial ao Roberto Prata, Paulo Sérgio, Sissy Sousa, Erik, Kelly e Felipe.

Aos meus irmãos Dênis, Dionne e Estephânia pelo apoio, à minha mãe Rivanda, minha avó Rita, minha tia Rosemeire e meu pai Gustavo pelo apoio e orações.

Ao meu filhos André e Vilany pela paciência e compreensão durante todo este tempo em que fiquei de certa forma ausente, especialmente nos momentos de stress. A minha esposa Genilce que abdicou inclusive de sua carreira profissional para me acompanhar nesta fase de minha vida, agradeço principalmente sua dedicação, compreensão e apoio.

Sumário

| | | |
|----------|---|----------|
| 1 | Introdução | 1 |
| 2 | Sistemas de Modelagem Geométrica | 5 |
| 2.1 | Esquemas de representação | 5 |
| 2.1.1 | Representação discreta | 5 |
| 2.1.2 | Representação paramétrica | 6 |
| 2.1.3 | Representação implícita | 6 |
| 2.2 | Primitivas para especificação de objetos implícitos | 7 |
| 2.2.1 | Analíticas | 7 |
| 2.2.1.1 | O Plano | 7 |
| 2.2.1.2 | As Quádricas | 8 |
| 2.2.1.3 | O Toro | 8 |
| 2.2.1.4 | As Super-quádricas | 8 |
| 2.2.2 | Procedurais | 9 |
| 2.2.2.1 | Fractais | 9 |
| 2.2.2.2 | Hipertextura | 10 |
| 2.2.3 | Baseadas em amostras | 10 |
| 2.2.3.1 | Amostragem irregular | 11 |
| 2.2.3.2 | Amostragem regular | 12 |
| 2.2.4 | Baseadas em esqueletos | 13 |
| 2.2.4.1 | Pontos | 13 |

| | | |
|---------|---|----|
| 2.2.4.2 | Curvas | 13 |
| 2.2.4.3 | Superfícies | 13 |
| 2.3 | Construção de modelos 3D | 14 |
| 2.3.1 | Técnicas construtivas | 14 |
| 2.3.2 | Técnicas de “forma livre” | 16 |
| 2.4 | Visualização de objetos implícitos | 17 |
| 2.4.1 | Pontos | 17 |
| 2.4.2 | Curvas | 18 |
| 2.4.2.1 | Curvas de silhueta | 18 |
| 2.4.2.2 | Curvas de nível | 18 |
| 2.4.2.3 | Superfícies | 19 |
| 2.4.2.4 | Volumes | 20 |
| 2.4.3 | Poligonalização | 21 |
| 2.4.3.1 | A classe de complexo celular usado na decomposição do espaço | 22 |
| 2.4.3.2 | A estratégia de <i>tracking</i> adotada para visitar as células intersectantes | 23 |
| 2.4.3.3 | O tipo de subdivisão usada | 23 |
| 2.5 | Interfaces de usuário baseadas em traços | 24 |
| 2.6 | Problema de Reconstrução de Superfícies | 27 |
| 2.7 | Métodos de Interpolação | 28 |
| 2.7.1 | Solução paramétrica polinomial ou polinomial contínua por partes | 28 |
| 2.7.2 | Soluções algébricas | 28 |
| 2.7.3 | Funções de base radiais | 29 |
| 2.7.4 | Métodos de Shepard | 29 |
| 2.7.5 | Métodos de subdivisão | 30 |

| | | |
|----------|--|-----------|
| 3 | Funções de Base Radiais | 31 |
| 3.1 | Existência | 34 |
| 3.2 | Escolha de parâmetros | 35 |
| 3.3 | Interpolação <i>Thin-Plate Spline</i> | 35 |
| 3.4 | Seleção dos centros | 36 |
| 3.5 | Funções de base radiais na computação gráfica | 39 |
| 3.6 | Modelagem Implícita usando RBFs | 40 |
| 3.7 | Funções de base radiais com suporte compacto | 41 |
| 3.8 | Métodos iterativos | 43 |
| 3.9 | Método da Partição da Unidade | 46 |
| 3.9.1 | Funções de peso | 47 |
| 4 | Deformações de Superfícies | 49 |
| 4.1 | Representação Baseada no Gradiente | 50 |
| 4.2 | Representação baseada no laplaciano | 52 |
| 4.3 | Deformação do espaço | 53 |
| 4.3.1 | Deformação de forma livre | 54 |
| 4.3.2 | Funções de base radiais | 54 |
| 4.4 | Trabalhos relacionados | 55 |
| 5 | Sistema para Modelagem e Deformação Baseado em Traços | 61 |
| 5.1 | Criação | 61 |
| 5.1.1 | Geração da superfície linear por partes usando o eixo cordal | 63 |
| 5.1.2 | Obtenção da superfície suave | 65 |
| 5.2 | Transformações geométricas | 65 |
| 6 | Detalhes de Implementação | 74 |
| 6.1 | Determinação das regiões de deformação | 74 |

| | | |
|----------|---|------------|
| 6.2 | Algoritmos | 75 |
| 6.2.1 | Determinação das regiões através de traços | 75 |
| 6.2.2 | Algoritmo de projeção de curvas | 77 |
| 6.2.3 | Seleção das regiões para inserção de características afiadas e suaves | 80 |
| 6.2.4 | Algoritmo de Deformação | 83 |
| 6.2.4.1 | Determinando uma transformação rígida a partir de um traço | 88 |
| 6.2.5 | Características suaves e afiadas | 89 |
| 7 | Resultados | 91 |
| 7.1 | Poder de expressão | 91 |
| 7.2 | Testes de desempenho | 94 |
| 7.3 | Propriedades diferenciais | 97 |
| 8 | Conclusão | 101 |
| A | Noções de Geometria Diferencial | 104 |
| A.1 | Curvas | 104 |
| A.2 | Curvaturas de curvas planas | 105 |
| A.3 | Superfícies | 107 |
| A.4 | Curvatura Gaussiana | 111 |
| A.5 | Curvatura média | 115 |
| A.6 | Curvatura média e divergente | 117 |
| B | Geometria diferencial discreta | 120 |
| B.1 | Curvatura média normal discreta | 120 |
| B.2 | Área de Voronoi | 124 |
| B.3 | Extensão para malhas arbitrárias | 125 |

Lista de Figuras

| | | |
|------|--|----|
| 2.1 | Representação discreta de modelos: da esquerda para a direita, o refinamento da malha triangular da cabeça de David de Michelangelo. | 6 |
| 2.2 | Exemplos de superfícies quádricas: esfera, cilindro e cone. | 8 |
| 2.3 | Alguns exemplos de superfícies super-quádricas. | 9 |
| 2.4 | Estrutura fractal que descreve um conjunto de Julia. | 10 |
| 2.5 | Exemplos de modelos baseados em hipertextura. | 10 |
| 2.6 | (a) Modelagem pela justaposição de paralelepípedos. (b) Alteração dos parâmetros de primitivas instanciadas. | 15 |
| 2.7 | (a) Justaposição de formas (b) Resultados da aplicação de operações booleanas entre cubo, esfera e cilindros. | 16 |
| 2.8 | Visualização por sistemas de partículas [7]. | 17 |
| 2.9 | Visualização de objetos implícitos usando curvas. (a) Visualização por curvas de silhueta. (b) Visualização por curvas de contorno [7]. | 19 |
| 2.10 | Visualização da superfície de objetos implícitos. (a) Modelo poligonal iluminado [103]. (b) Traçado de raios (<i>Ray Tracing</i>) [7]. | 20 |
| 2.11 | Visualização Volumétrica, lançamento de raios (<i>Ray casting</i>). | 20 |
| 2.12 | poligonalização. | 22 |
| 2.13 | Ambigüidade na poligonalização. | 23 |
| 2.14 | poligonalização adaptativa [7]. | 24 |
| 2.15 | Diferença entre o traçado e a clássica operação de arrastar. | 26 |

| | | |
|-----|---|----|
| 3.1 | Modelos do coelho (“ <i>bunny</i> ”) de Stanford. | 38 |
| 3.2 | Reconstrução de modelos obtidos usando o método <i>FastRBF</i> . A nuvem de pontos para o Buda e o dragão é constituída de 543.652 e 437.645 pontos respectivamente. [19] | 40 |
| 3.3 | Reparação automática dos buracos do modelo. [19] | 41 |
| 4.1 | Deformação do espaço do objeto | 53 |
| 5.1 | Criação de um modelo. (a) O usuário desenha a silhueta do objeto através de um traço. (b) O objeto é inflado. (c) O objeto visto de outro ângulo no qual foram acrescentados características localizadas dando ao modelo um perfil mais expressivo. | 62 |
| 5.2 | Três círculos máximos (vermelho) na região e seus respectivos MCTs (azul), e o eixo cordal (verde). | 64 |
| 5.3 | (a) Polígono de entrada. (b) Triangulação de Delaunay restrita e classificação dos triângulos. (c) Cálculo do eixo cordal. (d) Determinação da espinha. (e) Remoção de arestas pequenas. (f) Triangulação com elevação da espinha. | 66 |
| 5.4 | (a) Seleção das regiões: pontos fixos (vermelho), pontos de controle (verde) e pontos livres (azul). (b) a região de controle foi transladada para cima, e os pontos livres foram realocados segundo o funcional de energia (5.3) com $k = 2$ | 67 |
| 5.5 | Inclusão de características afiadas e suaves: (acima-esquerda) o traço inicial; (acima-direita) as regiões fixa, livre e rígidas são automaticamente determinadas pelo sistema; (abaixo-esquerda) característica afiada adicionada ao modelo; (abaixo-direita) característica suave adicionada ao modelo. | 72 |

| | | |
|-----|--|----|
| 6.1 | Deformação. (a) O usuário seleciona uma região de interesse através de um traço. (b) As três regiões: Fixa (Vermelho), Suporte (Azul) e Handle (Verde) (c) Uma segunda curva induz o perfil da deformação. (d) O modelo deformado. | 76 |
| 6.2 | Esquema de projeção do traço na superfície | 77 |
| 6.3 | interseção entre um <i>swath</i> e uma aresta da superfície | 78 |
| 6.4 | Triangulação em função do traço projetado. (a) A malha inicial (b) Um traço é projetado na malha. Uma lista de triângulos intersectados é criada. (c) Os vértices do traço projetado são realocados. (d) A malha é retriangulada. | 82 |
| 6.5 | Tipos de interseção do traço com os triângulos da superfície e os respectivos splits [75]. | 83 |
| 6.6 | Determinação das regiões de deformação por meio de traços. (a) O traço inicial. (b) O sistema detecta vários anéis de triângulos no modelo. (c) Apenas um par de anéis é escolhido. (d) Um algoritmo <i>flood fill</i> determina as regiões para a deformação. (e) Um segundo traço define o perfil de deformação. (f) O modelo deformado. | 89 |
| 7.1 | Deformação no nariz do camelo: (a) Modelo original. (b) Características afiadas. (c) Características suaves. | 92 |
| 7.2 | (a) Manequim original. (b) Manequim com várias deformações. | 93 |
| 7.3 | Deformação. (a) Modelo do tigre original. (b) Tigre deformado com a ferramenta apresentada. | 93 |
| 7.4 | (a) Cabeça do tigre original. (b) A boca foi aberta e foram inseridos dentes | 94 |
| 7.5 | Gráficos baseados nas informações das Tabelas 7.1 (a), (b) e (c) destacando o desempenho das técnicas de deformação por laplaciano e por RBFs usando decomposição LU para as suavidades C^0 (a), C^1 (b) e C^2 (c). | 96 |

| | | |
|------|---|-----|
| 7.6 | Na esquerda deformação por laplaciano e na direita deformação por RBF destacando a curvatura gaussiana com suavidades C^0 (a), C^1 (b) e C^2 (c). | 98 |
| 7.7 | Na esquerda deformação por laplaciano e na direita deformação por RBF destacando a curvatura média com suavidades C^0 (a), C^1 (b) e C^2 (c). | 100 |
| A.1 | Em (c), o limite da reta p_1, p_2 depende de como p_1 e p_2 se aproximam de p . | 105 |
| A.2 | A aplicação tangente. | 105 |
| A.3 | A curvatura mede a velocidade de variação da tangente | 106 |
| A.4 | O sinal da curvatura | 107 |
| A.5 | O terno f_1, f_2, f_3 está na orientação de e_1, e_2, e_3 em (b), e não está na orientação de e_1, e_2, e_3 em (c) | 108 |
| A.6 | A faixa de Möbius é não orientável: Percorrendo uma curva fechada na superfície, o vetor unitário normal volta na posição oposta, o que mostra que não é possível defini-lo continuamente | 108 |
| A.7 | (a) esfera. (b) elipsóide. (c) cilindro. (d) parabolóide. (e) parte de um cilindro. (f) parte de um parabolóide. | 109 |
| A.8 | A orientação do plano tangente, juntamente com N , coincide com a orientação dada de E^3 . | 110 |
| A.9 | As curvaturas de Euler | 110 |
| A.10 | A aplicação normal de Gauss | 112 |
| A.11 | O domínio regular é transformado em uma curva em forma de 8 | 112 |
| A.12 | A Aplicação Normal de Gauss recobre parte já coberta do domínio | 113 |
| A.13 | A Aplicação Normal de Gauss transforma a fronteira em uma curva que dá várias voltas antes de fechar | 113 |
| A.14 | O sinal da área da imagem esférica quando a aplicação normal é biunívoca | 114 |
| A.15 | Curvas paralelas | 115 |
| A.16 | Uma variação normal | 116 |

| | | |
|------|---|-----|
| A.17 | Representação bidimensional do domínio cilíndrico | 118 |
| B.1 | 1-anel do vértice x_0 | 121 |
| B.2 | (a) uma região de volume finito na malha usando <i>célula de Voronoi</i> , ou usando (b) <i>célula Baricêntrica</i> | 122 |
| B.3 | A área na vizinhança do vértice x_i não muda se o vértice movimentar-se no mesmo plano do 1-anel de x_i , podendo apenas aumentar no caso contrário. Portanto temos um mínimo local, o que mostra que a derivada da área com relação a posição de x_i é zero para regiões planas. | 123 |
| B.4 | Região de Voronoi em um triângulo não obtuso. | 124 |
| B.5 | Área mista entorno do vértice x_i : os triângulos 1 e 2 são obtusos, então o ponto que substitui o circuncentro é o ponto médio da aresta oposta ao ângulo obtuso | 126 |

Lista de Tabelas

| | | |
|-----|---|----|
| 3.1 | Principais funções radiais usadas para resolver o problema de interpolação. | 33 |
| 3.2 | Funções radiais com suporte compacto com a respectiva ordem de suavidade do interpolante. | 42 |
| 7.1 | Tempo de execução em segundos para montar a matriz L , computar sua inversa L^{-1} , computar a matriz B , montar a matriz Φ , computar sua inversa Φ^{-1} , computar a matriz P_ϕ , computar a matriz B , o tempo total da deformação por meio do operador laplaciano e o tempo total da deformação por meio de RBFs de acordo com o número de pontos usando decomposição LU para suavidades C^0 (a) C^1 (b) e C^2 (c). | 95 |

Capítulo 1

Introdução

A queda de preços dos computadores – além de características tais como redução das suas dimensões, alto desempenho e ubiqüidade – passou a permitir a exploração de novos tipos de aplicações, com uma tendência a suportar a cada um estilo de interação mais “natural”. Entre estes paradigmas de interação homem-computador, pós-WIMP (*Windows, Icons, Menus, Pointers*), 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 do computador, o qual gera ações apropriadas por meio de análise das gestuais 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 computadores sem fazer uso de seqüências de comandos cansativos e sem ter que lidar com um grande número de ícones, menus e ferramentas de seleção. A sua principal característica – a manipulação direta – tenta fazer do desenho uma atividade sem restrições artificiais. Encontrar uma técnica de modelagem de superfícies que seja interativa, de manipulação simples, que modele a rigidez ou suavidade da deformação de um objeto real e que não necessite de treinamento prévio para usuários casuais ainda é um grande desafio.

Quando falamos de modelos tridimensionais de aparência livre, nos referimos a objetos com 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 [109, 7, 105] – coletivamente denominados esquemas implícitos ou representações implícitas – vêm sendo sugeridas, particularmente aquelas que empregam Funções de Base Radiais (*Radial Basis Functions – RBFs*). Turk e O’Brien [102] [104], foram os primeiros a utilizar RBFs em modelagem, trabalhando com uma reconstrução *thin-plate* combinada com a técnica de amostra por partículas desenvolvida por Witkin e Heckbert [109] para visualização interativa.

Para obter uma renderização precisa, Turk e O’Brien convertem a superfície implícita em uma malha poligonal usando o algoritmo *marching cubes* [67], por exemplo, ou geram uma renderização direta da superfície por *ray-tracing*. Mas as duas técnicas são bastante lentas, não permitindo uma taxa de atualização de tela em tempo real após mudanças na superfície. Eles também descrevem algumas operações básicas como mover um ponto, adicionar um ponto, mudar a normal, de forma a permitir modelagem de objetos tridimensionais de aparência *livre*.

Botsch and Kobbelt [11], apresentam uma técnica de deformação de superfícies através do operador Laplaciano e posteriormente usam RBFs para realizar tais deformações [12]. Algumas técnicas com operações baseadas em traços podem preservar os detalhes da geometria do objeto, mas não permitem inserir uma linha característica na malha, como por exemplo um sulco [30, 60]. As técnicas de manipulação de *handles*, isto é, regiões

desenhadas sobre a superfície com comportamento rígido, [11, 98, 114, 12, 9] são intuitivas e produzem deformações suaves. No entanto, realizar deformações por meio da manipulação de um *handle* pode se tornar uma tarefa bastante trabalhosa e pouco intuitiva. Recentemente, Nealen et al. [78] apresentaram o FiberMesh: uma ferramenta de modelagem a mão livre que pode ser vista como uma extensão do Teddy [55].

Em nosso trabalho, fazemos um estudo das técnicas de deformação do espaço por meio de funções de base radiais e técnicas de deformação através do operador laplaciano, apresentando uma ferramenta que permite a modelagem e deformação de superfícies através de traços. O usuário faz um traço em uma região da superfície de seu interesse e, então, pode realizar deformações na parte da superfície selecionada.

Nossa idéia é usar a técnica de deformação do espaço por meio de RBFs e técnicas de discretização do laplaciano para realizar deformações nos objetos e inserir linhas características nas superfícies dos mesmos, de modo que se possa comparar os resultados obtidos pelos dois métodos.

Assim, fazendo traços no plano da tela, podemos criar um objeto grosseiro inicial, que não contém muitos detalhes, e por meio de deformações inserir detalhes na superfície do objeto, ou efetuar deformações mais complexas. Nosso protótipo também aceita a entrada de malhas triangulares prontas, eliminando a etapa de criação. A realização de algumas seqüências de deformações possibilita a obtenção de quadros simples de animação dos modelos.

Contribuição: As principais contribuições deste trabalho são a comparação entre os métodos de deformação por meio do operador laplaciano e deformação por RBFs, sendo feito um estudo com relação ao desempenho das técnicas, bem como a comparação das propriedades diferenciais das deformações obtidas pelos dois métodos. Adicionalmente foi implementado um protótipo para deformação e modelagem à mão livre que permite a comparação visual entre as duas técnicas abordadas.

Este trabalho está organizado como se segue. No Capítulo 2 apresentamos os sistemas de modelagem 3D mais utilizados em computação gráfica, seus esquemas de representação e algumas técnicas de construção e visualização, e uma visão geral dos métodos empregados nos problemas de reconstrução de superfície. O Capítulo 3 é dedicado à definição e às aplicações das funções de bases radiais na computação gráfica. No Capítulo 4 discutimos as técnicas baseadas na superfície do modelo destacando as representações baseadas no gradiente e representações baseadas no laplaciano e fazemos um resumo das técnicas de deformação linear e não linear do espaço. No Capítulo 5 apresentamos um sistema para modelagem e deformação baseado em traços, sendo que os detalhes de implementação são enfatizados no Capítulo 6. No Capítulo 7 são apresentados alguns resultados obtidos utilizando o sistema implementado e no Capítulo 8 apresentamos as conclusões e sugestões para trabalhos futuros.

Capítulo 2

Sistemas de Modelagem Geométrica

Neste capítulo apresentamos os sistemas de modelagem computacionais mais utilizados em computação gráfica, seus esquemas de representação, algumas técnicas de construção e visualização, e uma visão geral dos métodos empregados nos problemas de reconstrução de superfície.

No desenvolvimento de sistemas de modelagem existem duas questões principais 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 está relacionada com as técnicas usadas para construir essas estruturas e a interface do sistema com o usuário [105].

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 [25].

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 precisão também arbitrária. O uso comum de representações poligonais, em particular de tri-

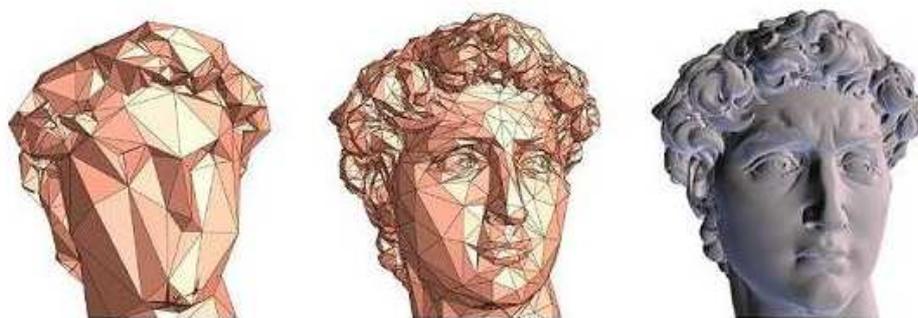


Figura 2.1: Representação discreta de modelos: da esquerda para a direita, o refinamento da malha triangular da cabeça de David de Michelangelo.

ângulos, 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.

2.1.2 Representação paramétrica

A superfície é representada por retalhos (*patches*), descritos por equações na forma paramétrica. 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 usa 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 para primitivas algébricas é dada pela equação

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

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

2.2.1.1 O Plano

É a primitiva algébrica mais simples, dado por um polinômio afim

$$Ax + By + Cz + D = 0, \quad (2.2)$$

*O material das seções 2.2, 2.3 e 2.4 foi baseado em [105], que contém uma excelente introdução à modelagem e visualização de objetos implícitos.

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

2.2.1.2 As Quádricas

São dadas por equações polinomiais de segundo grau

$$Ax^2 + By^2 + Cz^2 + 2Dxy + 2Exz + 2Fyz + 2Gx + 2Hy + 2Jz + K = 0. \quad (2.3)$$

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).



Figura 2.2: Exemplos de superfícies quádricas: esfera, cilindro e cone.

2.2.1.3 O Toro

É um produto cartesiano de dois círculos de raio A e B . É definido por um polinômio de quarto grau

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

2.2.1.4 As Super-quádricas

Não são algébricas, pois são definidas por equações polinomiais que envolvem potências de racionais. Elas generalizam as quádricas dando parâmetros de controle para manipular a forma destas primitivas. As super-quádricas são uma classe de superfícies que possuem descrição implícita e paramétrica de forma natural. São exemplos deste tipo os super-elipsóides, super-hiperbolóides, super-toróides, etc.



Figura 2.3: Alguns exemplos de superfícies super-quádricas.

2.2.2 Procedurais

Primitivas procedurais são objetos implícitos cuja função característica f é definida algorítmicamente, 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.

2.2.2.1 Fractais

Têm uma descrição algorítmica natural, podendo 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*) mostrado na Figura 2.4.

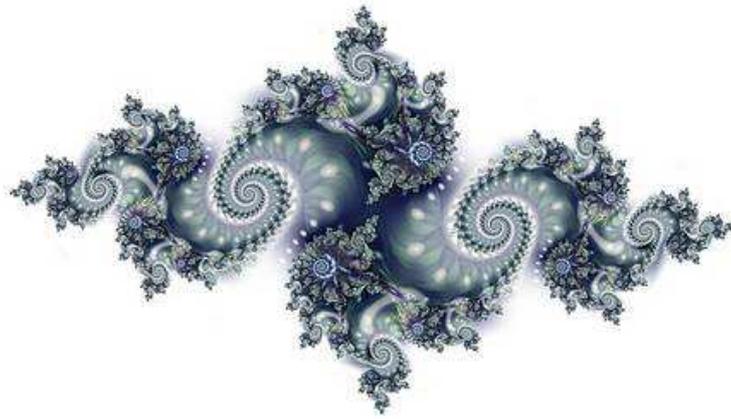


Figura 2.4: Estrutura fractal que descreve um conjunto de Julia.

2.2.2.2 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*). As hipertexturas podem representar modelos de objetos que possuem uma geometria mais complexa como fumaça, fogo, explosão e pelo de animais, conforme mostra a Figura 2.5.

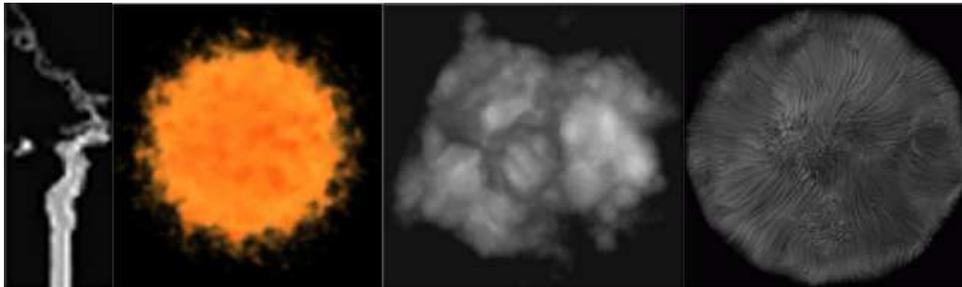


Figura 2.5: Exemplos de modelos baseados em hipertextura.

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 $x_i, i = 1, \dots, n$. Para gerar 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 .

2.2.3.1 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 $\{v_1, v_2, v_3, v_4\}$, $v_i \in \mathbb{R}^3$ e correspondentes valores $\{f_1, f_2, f_3, f_4\}$, realiza-se uma interpolação linear baricêntrica. Isto é,

$$f(x)|_T = \sum_{i=1}^4 \alpha_i f_i, \quad (2.5)$$

onde $\alpha_1, \alpha_2, \alpha_3$ e α_4 são as coordenadas baricêntricas do ponto x , 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.

2.2.3.2 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. Frequentemente 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 [105]. 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 $f(x, y, z) - c$.

2.2.4.1 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 de distância flexível. As propriedades da função de 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.

2.2.4.2 Curvas

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

2.2.4.3 Superfícies

Esta primitiva é definida como um deslocamento da superfície, restrita a posicionar-se a uma distância fixa e normal à 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 linguagens 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 construtiva 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 objetos sólidos a partir de operações sobre conjuntos (união, interseção e diferença).

Entre outras técnicas do enfoque construtivo, podemos mencionar o instanciamento

de primitivas e a combinação de objetos [5].

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 2.6(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 2.6(b)), como mudança de alturas e diâmetros.

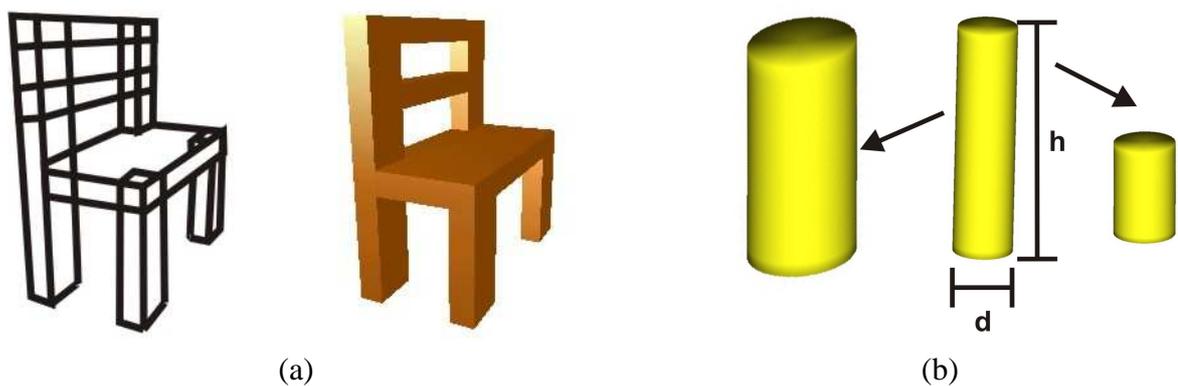


Figura 2.6: (a) Modelagem pela justaposição de paralelepípedos. (b) Alteração dos parâmetros de primitivas instanciadas.

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 2.7(a). Outra forma simples de combinar objetos é fazendo uso de operações sobre conjuntos, isto é, união, interseção e diferença (Figura 2.7(b)).

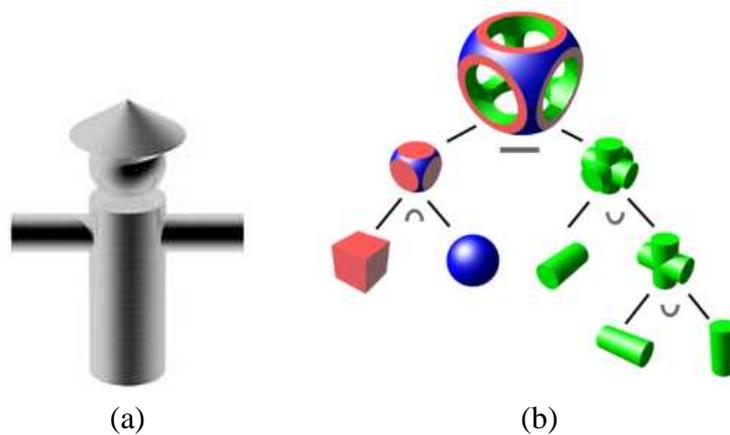


Figura 2.7: (a) Justaposição de formas (b) Resultados da aplicação de operações booleanas entre cubo, esfera e cilindros.

Finalmente, exemplos de sistemas implícitos baseados em técnicas construtivas são os sistemas *F-Rep* [82, 83] e o *Blob* [112]. 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. 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 [19, 104, 113, 26, 59], e serviu como ponto de partida para a presente tese.

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 e Heckbert [109] 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 objeto podem ser classificadas segundo a primitiva de desenho empregada [105]: pontos, curvas, superfícies ou volumes.

2.4.1 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 sistemas 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 2.8 ilustra esta técnica.

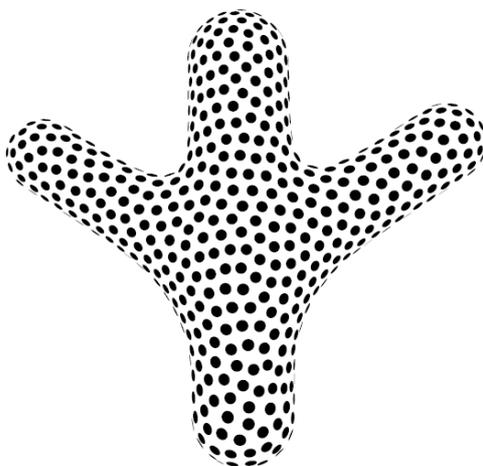


Figura 2.8: Visualização por sistemas de partículas [7].

2.4.2 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:

2.4.2.1 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 uma malha 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 2.9(a)).

2.4.2.2 Curvas de nível

São obtidas como produto da interseção 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 2.9(b)).

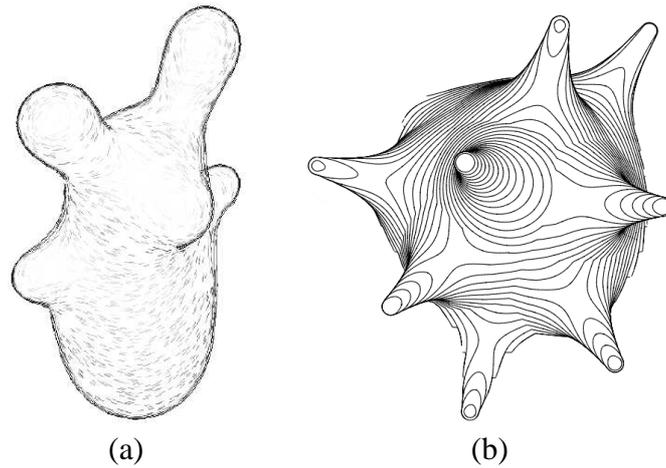


Figura 2.9: Visualização de objetos implícitos usando curvas. (a) Visualização por curvas de silhueta. (b) Visualização por curvas de contorno [7].

2.4.2.3 Superfícies

Uma renderização realística de um modelo de objeto definido implicitamente é alcançada 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 2.4.3.
- 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

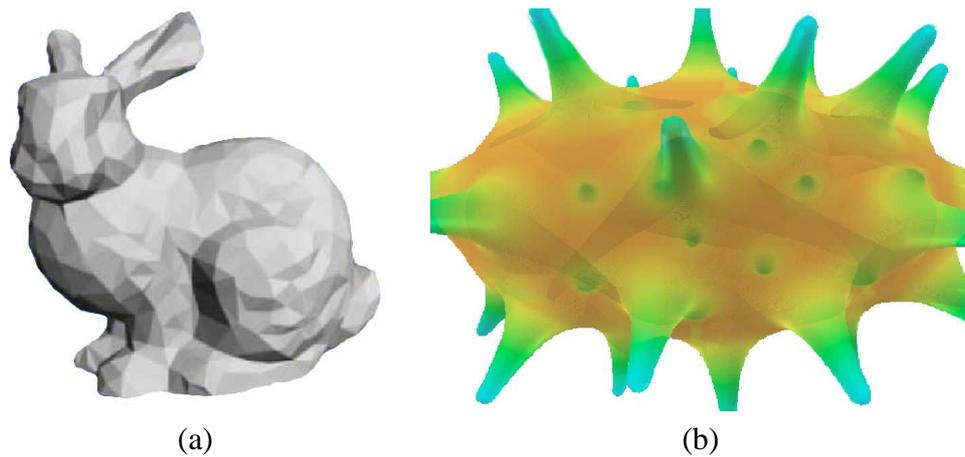


Figura 2.10: Visualização da superfície de objetos implícitos. (a) Modelo poligonal iluminado [103]. (b) Traçado de raios (*Ray Tracing*) [7].

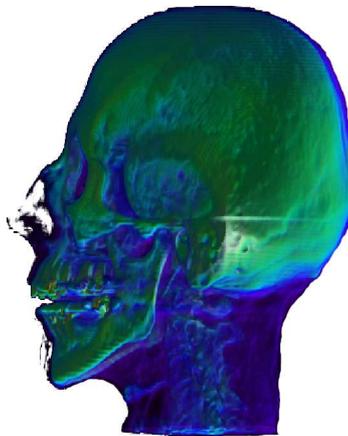


Figura 2.11: Visualização Volumétrica, lançamento de raios (*Ray casting*).

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.

2.4.2.4 Volumes

Existem duas situações em que as técnicas de visualização descritas anteriormente não são adequadas. A primeira acontece quando o objeto implícito é uma superfície delimitada clara. 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 2.11).

2.4.3 Poligonalização

Para capturar a geometria de um objeto implícito, é necessário amostrar a função implícita f que a define. Como esses objetos estão definidos indiretamente por f , não há uma maneira direta de gerar o conjunto de pontos f^{-1} que o definem, onde aqui f^{-1} denota a imagem inversa de f . 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 uma parametrização linear por partes associada. Como esta última é uma aproximação poligonal da borda do objeto, é conhecida também como um método de poligonalização.

Métodos de poligonalização 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 poligonalização extrínsecos podem ser sub-classificados de acordo com [105]:

2.4.3.1 A classe de complexo celular usado na decomposição do espaço

- Métodos não-simpliciais: subdividem o domínio de f construindo um complexo celular. O método de poligonalização não-simplicial mais conhecido é o algoritmo *Marching Cubes* [67]. Este subdivide o espaço regularmente em células cúbicas. O método consiste de três etapas principais (Figura 2.12). 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.

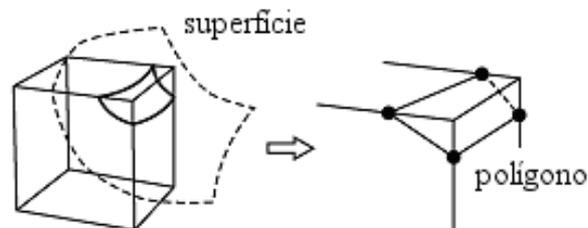


Figura 2.12: poligonalização.

Estes métodos são rápidos e simples de implementar. A principal desvantagem é que eles não são robustos. Por exemplo, veja na Figura 2.13 um caso de ambigüidade onde a interseção entre uma célula cúbica e a iso-superfície não pode ser determinada de uma única forma.

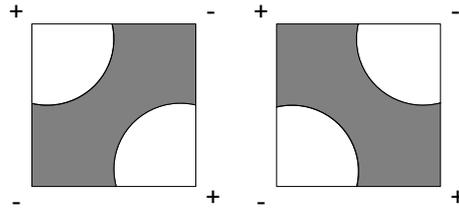


Figura 2.13: Ambigüidade na polygonalização.

- 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 [3].

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

2.4.3.2 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.

2.4.3.3 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 2.14 apresenta o resultado de um processo de poligonalização 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 poligonalização.

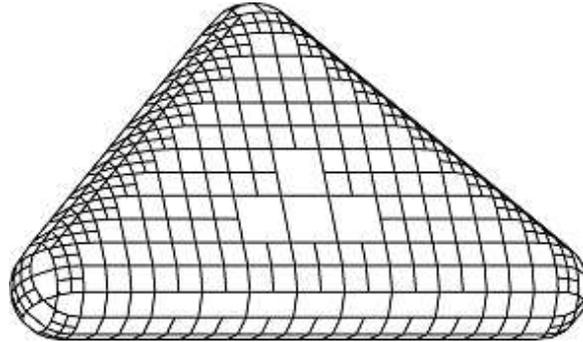


Figura 2.14: poligonalização adaptativa [7].

2.5 Interfaces de usuário baseadas em traços

Sistemas de modelagem 3D são normalmente projetados para a criação precisa e detalhada de modelos complexos [55]. 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” com que estes sistemas são implementados costumam representar uma dificuldade adicional para usuários sem experiência. 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. Tais modelos podem ser gerados com o auxílio de um sistema gráfico baseado em traços.

Metin et al. [92], 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” [92].

Por outro lado, Igarashi et.al. [54] 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 2.15 ilustra esta diferença. O traçado é uma maneira rápida, intuitiva e eficiente de expressar idéias gráficas num ambiente computado-rizado. 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 irris-tritos inspirado pela percepção humana. O processamento perceptual permite ao usuário realizar complicadas tarefas interagindo diretamente na cena, não reque-rendo que a tarefa seja decomposta numa seqüência de comandos complicados.

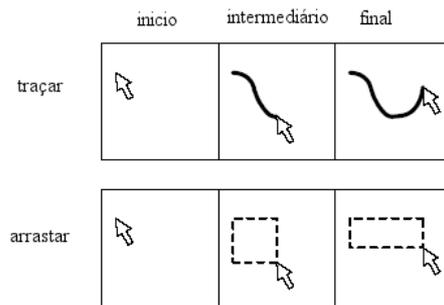


Figura 2.15: Diferença entre o traçado e a clássica operação de arrastar.

- 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 foto realista (*NPR-Non-Photo Realistic 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 esses 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 [92]:

- 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.

2.6 Problema de Reconstrução de Superfícies

O problema básico consiste em encontrar uma superfície que interpole um conjunto finito de n pontos do \mathbb{R}^3 $\{x_1, \dots, x_n\}$. Nos referimos a esse problema como um problema de *reconstrução de superfície*. Problemas deste tipo aparecem em várias áreas de aplicações como modelagem geométrica e visualização científica. Além do problema de reconstrução de superfícies descrito acima, existe uma versão mais simples e clássica onde os dados estão no \mathbb{R}^2 , mais precisamente, procuramos uma função $f(x)$ satisfazendo a alguns valores pré-determinados, ou seja, $f(x_k) = z_k$ para $k = 1, \dots, n$. Este tipo de problema é chamado *problema de reconstrução de função*.

Algumas variações importantes dessas versões básicas do problema incluem:

- quando os pontos são fornecidos juntamente com uma triangulação ou uma malha,
- quando os pontos são dados numa grade regular, tipicamente uma grade retangular,
- quando além dos seus valores, as normais dos pontos são especificadas, e
- quando os pontos são aproximados antes de serem interpolados.

Sem quaisquer restrições adicionais, podemos observar que existem várias soluções para o problema de interpolação de pontos espalhados. Por exemplo, quando os pontos

são dados com uma triangulação, a superfície linear por partes associada à triangulação é uma solução. Portanto, diferentes triangulações levam a soluções diferentes. Quando uma triangulação é fornecida, o foco da maioria das soluções é construir soluções suaves que sejam tangentes à triangulação ou que possuam curvatura contínua. Entretanto, os pontos muitas vezes são fornecidos sem qualquer triangulação ou informação de conectividade. Em tais casos, pode-se tentar triangular o conjunto de pontos e construir uma interpolação baseada na triangulação ou simplesmente usar uma interpolação baseada nos pontos que não requeira qualquer triangulação dos dados.

2.7 Métodos de Interpolação

Existem vários modos de classificar as soluções de uma interpolação de pontos espalhados. Destacamos a seguir cinco categorias baseadas na forma de resolução do problema.

2.7.1 Solução paramétrica polinomial ou polinomial contínua por partes

Talvez a representação mais popular no meio acadêmico e industrial para resolução de problemas de interpolação de dados espalhados seja a polinomial contínua por partes ou solução paramétrica racional. As soluções polinomiais são representadas como $x = x(u, v)$, $y = y(u, v)$ e $z = z(u, v)$, onde $x(u, v)$, $y(u, v)$ e $z(u, v)$ são funções polinomiais nos parâmetros u e v . O grau destes polinômios é chamado o grau paramétrico das soluções. Uma leve generalização é considerar soluções racionais que podem ser expressas na forma $x = \frac{x(u,v)}{w(u,v)}$, $y = \frac{y(u,v)}{w(u,v)}$, $z = \frac{z(u,v)}{w(u,v)}$, onde $w(u, v)$ também é uma função polinomial.

2.7.2 Soluções algébricas

Uma alternativa para as soluções paramétricas é construir funções ou superfícies de forma implícita, ou seja, a função ou superfície pode ser representada como o conjunto-zero de

alguma função do \mathbb{R}^3 . Mais formalmente, a solução é representada como algum subconjunto satisfazendo $f(x, y, z) = 0$. Além disso, a função f é geralmente escolhida como sendo um polinômio, e tais soluções são denominadas soluções algébricas. O grau do polinômio f é chamado o grau algébrico da solução. Uma forma implícita pode ser alcançada facilmente através de funções definidas por $z = g(x, y)$ reescrevendo-a como $f(x, y, z) = z - g(x, y) = 0$.

2.7.3 Funções de base radiais

O método de funções de base radiais é aquele onde as soluções são combinações lineares de funções radialmente simétricas. São os métodos de interpolação com funções de base radiais que utilizaremos neste trabalho, e que serão tratadas com mais detalhe no próximo capítulo.

2.7.4 Métodos de Shepard

A idéia básica dos Métodos de Shepard é definir uma interpolação dos dados espalhados $f(x, y)$ como uma média ponderada dos valores $z_i = f_i$ escolhendo algumas funções de mistura ou funções de peso. Mais precisamente, $f(x, y) = \sum_{i=1}^n w_i(x, y) f_i$ onde as funções de peso $w_i(x, y)$ satisfazem as seguintes propriedades:

- $w_i(x_j, y_j) = \delta_{ij}$;
- Propriedade da partição da unidade, ou seja, $\sum_{i=1}^n w_i(x, y) = 1$;
- $w_i(x, y)$ é não-negativa e pelo menos contínua.

Uma interpolação de dados espalhados que usa o método de Shepard pode ser radialmente simétrica, polinomial, uma função racional ou uma mistura delas. Este método para solução do problema de interpolação de dados espalhados tem sido usado para resolver problemas de reconstrução de função.

Shepard sugeriu que as funções de peso $w_i(x, y)$ fossem $\frac{\sigma_i(x, y)}{\sum_{i=1}^n \sigma_i(x, y)}$ onde $\sigma_i(x, y) = \frac{1}{r_i}$. Em outras palavras, as funções de peso são obtidas normalizando-se o inverso das funções distância. Estas funções de peso são radialmente simétricas, positivas e contínuas. Em contraste com as aproximação de funções de base radiais, esta aproximação não requer a resolução de um sistema de equações.

2.7.5 Métodos de subdivisão

Os métodos de subdivisão aplicam um processo de corte dos cantos de um modelo poliedral, ou seja, o poliedro é refinado ou subdividido através da inclusão de vértices, arestas e faces utilizando uma heurística adequada. As técnicas baseadas nesse tipo de solução dependem fortemente da estruturação dos dados, ou seja, da malha na qual os pontos estão dispostos e, geralmente, não apresentam uma forma analítica fechada. Entretanto, em alguns casos particulares, como por exemplo quando os pontos estão dispostos em uma grade regular, essas soluções reduzem-se a superfícies interpolantes bastante conhecidas, como o produto tensorial de *B-splines*. Talvez a mais conhecida técnica de subdivisão seja a introduzida por Chaikin em 1974 para curvas [21]. Ele gerou uma curva suave a partir de um polígono refinando-o através de cortes sucessivos de seus cantos. A curva suave obtida por este esquema se mostra equivalente a uma curva *B-spline* quadrática obtida como se os vértices originais fossem os pontos de controle. Esta idéia foi generalizada para superfícies por Catmull-Clark [20] e Doo e Sabin [29] em 1978. Nestes esquemas, uma malha inicial 3D de controle é determinada. Embora as faces desta configuração não sejam necessariamente planares, alguns casos particulares conhecidos surgem quando cada face é um triângulo ou um retângulo. No limite, com o número de passos de subdivisão finito, o poliedro converge a uma superfície. Com uma escolha cuidadosa da heurística para o corte dos cantos, é possível mostrar que a superfície limitada existe, é contínua e possui um plano tangente contínuo.

Capítulo 3

Funções de Base Radiais

Como a criação e as deformações de superfícies que realizamos em nosso trabalho usam técnicas de deformação do espaço por meio de funções de bases radiais (RBFs), destinamos este capítulo para definir e contextualizar o uso de tais funções na computação gráfica.

Uma função $\phi(\|x - x_i\|)$ onde $\|\cdot\|$ denota a norma Euclidiana é chamada uma *função de base radial*, porque depende apenas da distância Euclidiana entre os pontos x e x_i . Os pontos x_i são chamados de *centros* ou *nós*, e denotaremos por X_i o conjunto dos centros. Em particular, a função $\phi(\|x - x_i\|)$ é radialmente simétrica na vizinhança do centro x_i . Note que estamos denotando por $x = (x, y, z)$ um ponto do \mathbb{R}^3 .

A solução do problema de interpolação de pontos espalhados é obtida através de uma combinação linear de translações de uma RBF convenientemente escolhida. Geralmente um termo polinomial é adicionado à solução quando ϕ é condicionalmente positiva definida, ou para alcançar precisão polinomial. De modo mais formal, a solução procurada do problema de interpolação tem a seguinte forma:

$$s(x) = p(x) + \sum_{i=1}^n \lambda_i \phi(\|x - x_i\|), \quad (3.1)$$

onde:

- s é a função de interpolação,

- p é um polinômio, geralmente linear ou quadrático,
- os λ_i são os pesos da função de interpolação,
- ϕ é a função de base radial.

A função interpoladora consiste de uma soma ponderada de funções de base radiais ϕ radialmente simétricas ao centro x_i , adicionadas a um polinômio p de baixo grau. Dado um conjunto de n pontos x_i e valores w_i , o processo de encontrar uma interpolação RBF s , tal que

$$s(x_i) = w_i, \quad i = 1, 2, \dots, n \quad (3.2)$$

é chamado de *ajuste*. O ajuste RBF é definido pelos coeficientes λ_i da função de interpolação no somatório, juntamente com os coeficientes do termo polinomial $p(x)$. Denotaremos por \mathcal{P}_m , o espaço dos polinômios de grau menor que m . Note que $m = 0, 1, 2$ corresponde respectivamente aos casos quando não temos nenhum polinômio, quando uma função constante é adicionada, ou um polinômio linear é adicionado à interpolação.

Se tomamos $\{p_1, \dots, p_l\}$ uma base de monômios do polinômio p e $\mathbf{c} = (c_1, \dots, c_l)$ seus coeficientes dados em termos desta base, então a condição de interpolação (3.2) pode ser escrita como um sistema linear na forma matricial por

$$\begin{pmatrix} A & P \\ P^\top & \mathbf{0} \end{pmatrix} \begin{pmatrix} \lambda \\ \mathbf{c} \end{pmatrix} = \begin{pmatrix} \mathbf{w} \\ \mathbf{0} \end{pmatrix} \quad (3.3)$$

onde

$$A_{i,j} = \phi(r_{ij}), \quad r_{ij} = \|x_i - x_j\| \quad i, j = 1, \dots, n$$

$$P_{i,j} = p_j(x_i), \quad i = 1, \dots, n, \quad j = 1, \dots, l$$

$$\mathbf{w} = (w_1, \dots, w_n)^\top$$

$$\lambda = (\lambda_1, \dots, \lambda_n)^\top$$

Resolvendo o sistema linear (3.3) determinamos λ e c , e conseqüentemente $s(x)$. Além disso, assumiremos que os pontos satisfazem a seguinte condição geométrica :

$$p(x) \in \mathcal{P}_m, p(x_i) = 0, i = 1, \dots, n \Rightarrow p \equiv 0 \quad (3.4)$$

Há mais de 30 anos, Hardy [46] introduziu uma das mais populares funções de base radial chamada de *multi-quádrica de Hardy*. As multi quádricas são definidas por $\phi(r) = \sqrt{r^2 + h^2}$ onde h é um parâmetro escolhido convenientemente. Note que a multi-quádrica *cresce* quando a distância entre os centros aumenta. Para evitar isto Hardy usou a *multi-quádrica inversa* $\phi(r) = \frac{1}{\sqrt{r^2 + h^2}}$ que diminui à medida em que a distância dos centros aumenta. As multi-quádricas de Hardy e multi-quádricas inversas eram usadas com sucesso em várias aplicações sem muita justificativa teórica, quando Harder e Desmarais [45] introduziram as *thin-plate splines* (TPS) que teve um estudo teórico bastante amplo feito por Duchon [32] que desenvolveu uma teoria baseada em princípios variacionais. Posteriormente as TPS foram estudadas por Meinguet [71], Madych e Nelson [68]. As TPS em 2D são definidas por $\phi(r) = r^2 \log r$ e também são RBFs que crescem com o aumento da distância aos centros. O sistema (3.3) sempre tem solução para $m \geq 2$. Além dessas três RBFs, multi-quádricas, multi-quádricas inversas e TPS, existem várias outras RBFs bastante pesquisadas, e entre as diferentes funções radiais que têm sido usadas na solução do problema de interpolação destacamos as da Tabela 3.1:

| Função base | |
|-----------------------------|---|
| TPS bi-harmônica 2D | $\phi(r) = r^2 \log r$ |
| TPS bi-harmônica 3D | $\phi(r) = r$ |
| TPS tri-harmônica 2D | $\phi(r) = r^4 \log r$ |
| TPS tri-harmônica 3D | $\phi(r) = r^3$ |
| TPS transladada | $\phi(r) = (r^2 + h^2) \log(r^2 + h^2)^{1/2}$ |
| Multi-quádrica de Hardy | $\phi(r) = \sqrt{r^2 + h^2}$ |
| Multi-quádrica inversa | $\phi(r) = \frac{1}{\sqrt{r^2 + h^2}}$ |
| Multi-quádrica generalizada | $\phi(r) = (r^2 + h^2)^{k/2}$ |
| Gaussiana | $\phi(r) = e^{h^2 r^2}$ |

Tabela 3.1: Principais funções radiais usadas para resolver o problema de interpolação.

3.1 Existência

Dada uma RBF $\phi(r)$ e n pontos espalhados, consideremos a matriz A $n \times n$ quadrada simétrica como na equação (3.3). A RBF $\phi(r)$ é dita positiva definida se, e somente se, $v^t Av \geq 0$ para todo $v \in \mathbb{R}^n$. A RBF $\phi(r)$ é dita condicionalmente positivo definida se $v^t Av > 0$ sempre que $v \neq 0$. Se a função de base radial é estritamente positiva definida, então a matriz A é inversível, condição necessária e suficiente para que o sistema de equações lineares (3.3) tenha solução.

Consideremos agora \mathcal{P}_m o espaço dos polinômios de grau menor que m , e seja $V = (v_1, \dots, v_n)$ um vetor do \mathbb{R}^n que satisfaz $\sum_{i=1}^n v_i q(x_i) = 0$ para todo $q \in \mathcal{P}_m$ e todo centro x_i da RBF $\phi(r)$. A RBF $\phi(r)$ é dita condicionalmente positivo definida (CPD) de ordem m se, e somente se, $v^t Av \geq 0$ para todo $v \in V$. A RBF $\phi(r)$ é dita condicionalmente estritamente positivo definida (CEPD) de ordem m se $v^t Av > 0$ para $v \neq 0$. Pode ser provado que o sistema (3.3) tem solução única se a RBF é condicionalmente estritamente positiva definida de ordem m e os pontos espalhados satisfazem à condição geométrica (3.4).

Foi provado por Micchelli [73] a seguinte caracterização para funções condicionalmente positiva definidas, de onde deriva o seguinte resultado:

Teorema 3.1.1. *A função $f(t)$ é condicionalmente positiva definida de ordem m em \mathbb{R}^k para $k > 1$ se, e somente se, $(-1)^j \frac{d^j}{dt^j} f(\sqrt{t}) \geq 0, t > 0, j \geq m$. Se além disso $\frac{d^m}{dt^m} f(\sqrt{t}) \neq \text{constante}$, então $f(t)$ é condicionalmente estritamente positiva definida de ordem m .*

Agora fica fácil verificar que *thin-plate splines* ($m \geq 2$), multi-quádricas ($m \geq 1$) e multi-quádricas inversas ($m \geq 0$) são condicionalmente estritamente positiva definidas. Podemos obter alguns resultados derivados do teorema acima. Em particular o problema de interpolação de pontos espalhados tem solução para $m = 0$ e $f(t)$ CEPD de ordem 1,

sempre que $f(t) < 0$ para $t > 0$. Tal resultado garante a solução para o problema de interpolação para multi-quádricas sem a adição de qualquer constante ou termo polinomial.

3.2 Escolha de parâmetros

Algumas RBFs como multi-quádrica, multi-quádrica inversa e *thin-plate splines* transladadas como na tabela 3.1 dependem de um parâmetro h . Fora alguns resultados teóricos isolados, a maioria dos resultados são experimentais. Estes resultados indicam que o desempenho da interpolação radial depende de forma crítica da escolha deste parâmetro [17]. Nos primeiros experimentos, este parâmetro era escolhido como sendo a distância média entre os pontos e funcionava de forma satisfatória [38, 46]. Subseqüentemente, a exatidão de tais interpolações foram melhoradas pela escolha de um parâmetro que reduziu a diferença entre a interpolação multi-quádrica e multi-quádrica inversa [37]. Parâmetros com valores diferentes e diferentes pontos também foram usados para melhorar a exatidão destas interpolações [58]. Entretanto, observa-se que é fácil construir conjuntos de pontos no qual a matriz da Equação (3.3) pode ser singular quando valores diferentes de parâmetros são usados em diferentes pontos. As melhores interpolações multi-quádricas usam uma combinação linear de funções bases centradas no mesmo ponto mas com valores diferentes para o parâmetro h [16]. Floater e Iske [36] também construíram interpolações em vários passos usando funções de base radiais com suporte compacto. Discutiremos funções de base radiais com suporte compacto na seção 3.7.

3.3 Interpolação *Thin-Plate Spline*

Thin Plate Spline (TPS) em 3D é um método de interpolação que encontra uma superfície suave com energia de curvatura* mínima, e que passa por todos os pontos dados. A TPS para três pontos de controle é um plano, para mais de três é geralmente uma superfície curva e para menos de três é indefinida. O nome “*Thin Plate*” advém do fato de que

*A definição formal de curvatura pode ser encontrada no Apêndice.

a TPS simula o comportamento aproximado de uma fina folha de material flexível com elasticidade infinita, quando forçada em determinados pontos de controle. A folha resiste a mudanças suaves nas posições entre os pontos. Esta resistência funciona como uma função de energia E .

As TPS são particularmente populares na representação de transformações de formas, por exemplo, *morphing* ou detecção de formas e é freqüentemente usada em visão computacional.

Existem várias soluções distintas para o problema de interpolação de pontos espalhados da Equação (3.2). Mas se definirmos “suavidade” de um modo particular, existe uma única solução para tal problema, e esta solução é a interpolação *thin-plate spline* dos pontos.

Consideremos a função energia

$$\begin{aligned}
 E(s) = & \int_{\mathbb{R}^3} \left(\frac{\partial^2 s(x)}{\partial x^2} \right)^2 + \left(\frac{\partial^2 s(x)}{\partial y^2} \right)^2 + \left(\frac{\partial^2 s(x)}{\partial z^2} \right)^2 \\
 & + 2 \left(\frac{\partial^2 s(x)}{\partial x \partial y} \right)^2 + 2 \left(\frac{\partial^2 s(x)}{\partial x \partial z} \right)^2 + 2 \left(\frac{\partial^2 s(x)}{\partial y \partial z} \right)^2 dx dy dz
 \end{aligned} \tag{3.5}$$

Esta função é uma medida de “suavidade”, ou seja, é uma medida da curvatura de $s(x)$ em uma região do \mathbb{R}^3 . Qualquer saliência ou protuberância na superfície resultará em um aumento no valor de E . Uma superfície suave que não tem regiões com curvatura alta, terá baixos valores para E . Como (3.5) possui todos os termos elevados ao quadrado, não podemos ter valores negativos para E .

A Interpolação *thin-plate spline* para o problema de interpolação é uma função $s(x)$ que satisfaz (3.2) e que tem o menor valor possível para $E(s)$.

3.4 Seleção dos centros

Até agora consideramos apenas o caso quando os centros para interpolação de base radial estão localizados nos pontos dados. Entretanto, isto não necessariamente é verdadeiro. A

pergunta é: O que acontece se os centros são outros pontos? Há uma base teórica a qual garante que a interpolação é ideal se os centros estão localizados nos pontos dados [90]. As experiências com posicionamento de centros focalizaram-se principalmente em métodos de aproximação [70]. Embora a aproximação por mínimos quadrados seja a mais popular, outras funções objetivas também foram consideradas para minimização [88, 43]. O objetivo é alcançar uma boa aproximação com poucos centros. Isto pode reduzir o tempo de computação em construir e avaliar a interpolação ou uma aproximação. Os resultados indicam que esta economia significativa resulta em introduzir erros muito pequenos [18]. Entretanto, o comportamento do número de condições e portanto os problemas de estabilidade da computação (dependendo do espaço entre os pontos relativos ao parâmetro da multi-quádrica) para interpolação por funções multi-quádricas leva a uma interpolação de mínimos quadrados. Sivakumar e Ward [94] deram limites superiores para as normas da inversa das equações normais associadas ao problema de aproximação de mínimos quadrados. Franke, Hagen, e Nielson [39] descreveram um método guloso para a seleção de centros (nós) para aproximação de mínimos quadrados. Portanto, um algoritmo guloso pode ser usado para ajustar iterativamente uma RBF com a *exatidão desejada*.

Um algoritmo guloso simples consiste dos seguintes passos:

Algoritmo 3.1

1. Escolha um subconjunto de nós de interpolação e ajuste uma RBF usando apenas estes nós.
 2. Avalie o resíduo, $\epsilon_i = w_i - s(x_i)$, em todos os nós.
 3. Se $\max\{|\epsilon_i|\} < \textit{exatidão desejada}$ pare.
 4. Senão acrescente novos centros onde ϵ_i é grande.
 5. Reajuste a RBF e volte para o passo 2.
-

Se uma exatidão diferente δ_i é especificada em cada ponto, então a condição no passo 3 pode ser substituída por $|\epsilon_i| < \delta_i$.

A redução do número de centros não é essencial quando são usados métodos rápidos. Entretanto, reduzir o número de centros da RBF resulta em menor espaço de memória e menor tempo de avaliação, sem perda de exatidão.



1000 pontos



2000 pontos



4000 pontos



8000 pontos



16000 pontos



32000 pontos

Figura 3.1: Modelos do coelho (“*bunny*”) de Stanford.

A Fig. 3.1 ilustra o processo de ajuste com redução de centros. A medida que mais

centros são adicionados à RBF, o conjunto-zero aproxima-se cada vez mais do conjunto inteiro de pontos dados. O algoritmo guloso frequentemente resulta num tempo menor ao fazer o ajuste, mesmo com uma redução moderada no número de centros. Isto deve-se à eficiência associada à resolução e avaliação de um sistema semelhante em cada iteração e ao fato que iterações iniciais resolvem problemas muito menores.

3.5 Funções de base radiais na computação gráfica

A idéia de usar RBFs para modelagem de superfícies implícitas foi introduzida por Savchenko [89] e Turk and O'Brien [103]. O método consiste em produzir um campo escalar, no qual a superfície desejada é o conjunto-zero, enquanto que pontos interiores e exteriores à superfície são mapeados em valores positivos e negativos, respectivamente. Infelizmente, a natureza global desta representação limita seu uso na modelagem de superfícies descritas por um grande número de pontos.

Morse *et al.* [77] usaram funções base de suporte compacto, introduzidas por Wendland [108] para confrontar este problema. Kojekine *et al.* [62] melhoraram 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 [77].

Carr *et al* [19] usaram RBFs para reconstruir a superfície de objetos para os quais dados provenientes de escaneamento 3D (*range data*) estavam disponíveis. Para poder lidar com grandes quantidades de dados, eles beneficiaram-se das otimizações reportadas por Beatson [8]. A figura 3.2 mostra alguns dos resultados obtidos. Mais tarde, Laga *et al.* [64] introduziram as *Parametric Radial Basis Functions*, similar ao trabalho de Carr, mas adaptado para suportar objetos suaves assim como também objetos com arestas “afiadas” (*sharp boundaries*).

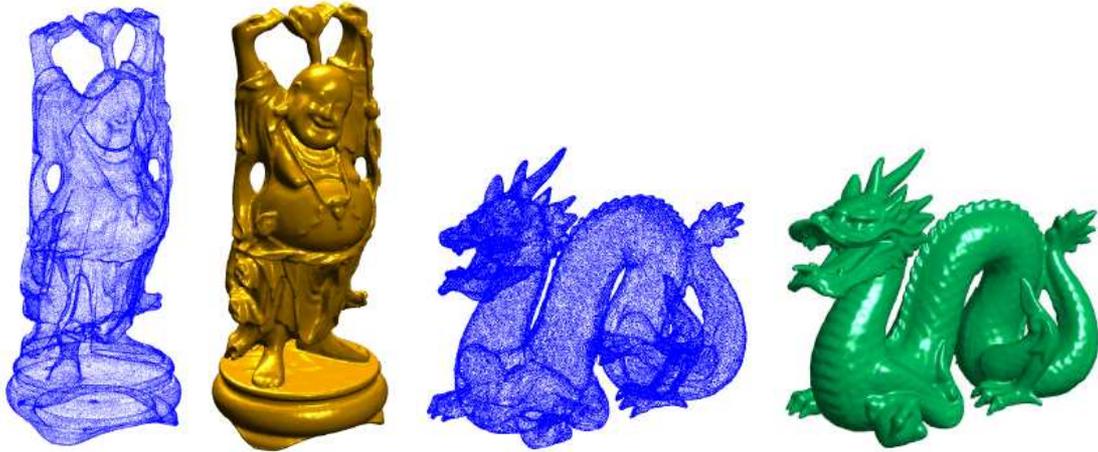


Figura 3.2: Reconstrução de modelos obtidos usando o método *FastRBF*. A nuvem de pontos para o Buda e o dragão é constituída de 543.652 e 437.645 pontos respectivamente. [19]

Recentemente, Ohtake *et al.* [80] introduziram a *Multi-level Partition of Unity Implicit*, 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 locais respeitando um peso derivado do método de Partição da Unidade. O método de Partição da Unidade é visto com maiores detalhes na Seção 3.9.

3.6 Modelagem Implícita usando RBFs

A modelagem implícita de superfícies usando RBFs é uma técnica relativamente recente em computação gráfica. Oferece a habilidade de interpolação/reparação através de falhas grandes e irregulares em superfícies incompletas sem restringir a topologia do objeto e sem precisar de algum conhecimento prévio sobre a forma do objeto. A figura 3.3 do trabalho de Carr *et al.* [19] ilustra isto.

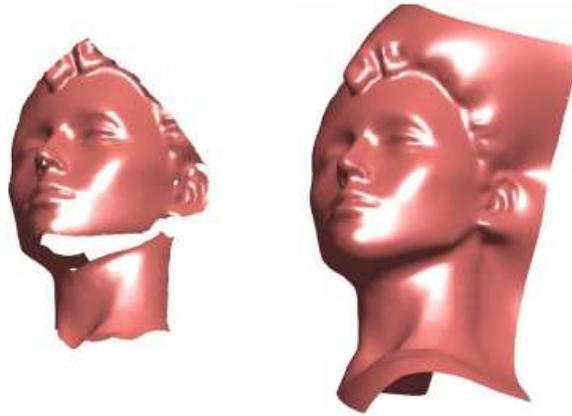


Figura 3.3: Reparação automática dos buracos do modelo. [19]

Além disso, nos oferece as seguintes vantagens:

- Representação compacta numa só 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 os dados.
- Podem ser avaliadas em qualquer lugar do espaço e assim gerar malhas de uma resolução desejada.
- Gradientes e derivadas superiores podem ser analiticamente calculados.
- Normais à superfície são facilmente computadas.

3.7 Funções de base radiais com suporte compacto

Para resolver o sistema de equações, métodos diretos ou iterativos simples têm complexidade $O(n^2)$ para o armazenamento e $O(n^3)$ para as operações. Para dados com mais de dez mil pontos, que geralmente aparecem em problemas de visualização científica, a utilização de métodos simples fica impraticável.

Além disso, para as RBFs da tabela 3.1, a matriz resultante pode ser afetada por problemas de mau condicionamento. Isto ocorre porque as entradas na matriz vão crescendo à medida que se afastam da diagonal.

Estes problemas de instabilidade e o alto custo computacional associado à construção da função interpoladora têm incentivado pesquisas no intuito de obter funções radiais que decresçam com a distância. Um trabalho bastante interessante é devido a Wendland [108], em parte iniciado por Wu [110, 111] que foi um dos pioneiros na pesquisa de matrizes simétricas positiva definidas com suporte radialmente compacto. De fato, Wendland demonstrou que certas funções polinomiais por partes satisfazem à todas essas propriedades.

Geralmente, as funções de base radiais com suporte compacto (CSRBFs para abreviar) têm a seguinte forma [108]:

$$\phi(r) = \begin{cases} p(r) & \text{se } 0 \leq r \leq 1, \\ 0 & \text{se } r > 1, \end{cases} \quad (3.6)$$

onde p é um polinômio.

Para vários graus de continuidade da superfície interpolada, Wendland [108] derivou as CSRBFs mostradas na Tabela 3.2.

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

Tabela 3.2: Funções radiais com suporte compacto com a respectiva ordem de suavidade do interpolante.

Usando CSRBFs, o sistema de equações lineares tende a ser um sistema linear esparsos. As entradas na matriz serão zero para pontos com distâncias maiores que o raio do suporte. Os sistemas lineares esparsos podem ser eficientemente resolvidos por métodos diretos ou iterativamente.

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

3.8 Métodos iterativos

Na prática, métodos de interpolação como funções de base radiais são usados frequentemente para aproximações com grande números de pontos, e, neste caso, a solução numérica resultante dos sistemas lineares tornam-se não-triviais em face de alguns problemas como erros de arredondamento. Além disso, o armazenamento também pode se tornar um problema significativo se a quantidade de centros é muito grande, até mesmo com as mais modernas estações de trabalho as quais frequentemente têm grande quantidade de memória principal.

Vários pesquisadores relatam métodos que conseguem soluções de alta qualidade para o problema de interpolação de dados espalhados. A adoção dos métodos em aplicações maiores, por exemplo em reconstrução de superfícies, onde o número de pontos é grande, tornam-se inviáveis pelo alto custo computacional. Entretanto, isto está associado à solução numérica das equações de interpolação e à avaliação da aproximação resultante.

Para a maioria das funções de base radiais que encontramos até agora, resolver as equações de interpolação (3.3) com métodos diretos, tal como eliminação Gaussiana, requer $O(n^3)$ operações, sendo computacionalmente caro e conseqüentemente proibitivo para o tamanho de n como mencionado anteriormente. Isto ocorre porque a matriz de interpolação não é esparsa a menos que as funções de base radiais sejam de suporte compacto como visto anteriormente. Isso ajuda a aliviar o trabalho numérico, mas na maioria

dos casos não é suficiente. Se a matriz de interpolação é positiva definida, o método de Cholesky [16] pode ser aplicado, visto que requer baixo esforço computacional, mas até mesmo este trabalho reduzido também será grande se houver 5000 pontos, por exemplo, ou mais.

Em resumo, uma aproximação útil consiste em aplicar métodos iterativos para resolver grandes sistemas lineares em vez de algoritmos diretos. Isto é especialmente desejável quando os métodos são usados dentro de outros esquemas iterativos, por exemplo, na solução numérica de equações diferenciais parciais. Assim, soluções aproximadas de problemas de interpolação são aceitáveis, se sua precisão estiver dentro de uma margem de erro especificada, que estão de acordo com a precisão dos métodos de aproximações que usam funções de base radiais.

Destacamos três tipos de aproximações para resolver os sistemas lineares por iteração. As aproximações são responsáveis tanto pela solução rápida dos sistemas de equações lineares acima mencionados quanto pela avaliação rápida das aproximações, uma vez que os dois assuntos estão intimamente relacionados.

A primeira aproximação usa o fato que as funções interpolantes, por exemplo, *thin-plate splines* (ou também multi-quádricas), apesar de seu suporte global, na realidade atuam localmente. Isto ocorre especialmente no caso em que os dados são armazenados em uma grade quadrada de inteiros e no qual ocorre um decaimento rápido das funções de Lagrange no espaço das funções de base radial de interpolação. Assim, de um certo modo, espera-se poder localizar bem a computação dos coeficientes de interpolação. Isto motiva a idéia básica da primeira aproximação que nós chamaremos *método BFGP* devido seus inventores (Beatson-Faul-Goodsell-Powell) o qual tem uma implementação alternativa na forma de um subespaço de Krylov [16].

Este método assume que os coeficientes da função interpolante dependem quase exclusivamente da vizinhança dos pontos espalhados, então decompondo e resolvendo o

problema adequadamente e derivando um método iterativo que reduz rapidamente os resíduos em cada passo. O tipo de aproximação empregada neste método é a de decomposição do domínio. De fato, esta aproximação subdivide o domínio espacial X do problema que contém todos os pontos em pequenos subdomínios com poucos centros (pequenos subconjuntos $X_i \subset X$), onde o problema pode ser resolvido mais facilmente, visto que é muito menor. A idéia difere dos algoritmos de decomposição de domínio habituais, já que os subdomínios onde estão os centros não são disjuntos, isto é, podem estar sobrepostos.

O segundo método denominado multipolos (*multipole*), é uma aproximação criada para funções de base radiais devida principalmente a Powell, Beatson, Luz e Newsam. Originalmente, porém, a idéia básica é devida a Greengard e Rokhlin (1987) que usaram o método de multipolos para resolver equações integrais numericamente usando núcleos com suporte global para matrizes de discretizações grandes e densas. Assim eles queriam reduzir a complexidade computacional de simulações de partícula. O método de multipolos faz uma decomposição dos dados de um modo semelhante ao algoritmo de BFGP, mas não usa a localidade das funções aproximantes ou qualquer função como as de Lagrange. Além disso, os conjuntos X_i nos quais os centros são divididos, são geralmente disjuntos como uma estrutura de grade.

Em vez da localidade da função de Lagrange, os métodos rápidos de multipolos (*fast multipole*) usam o fato de que as funções de base radiais são como *thin-plate splines*, exceto na origem, expansível em séries de Laurent infinitas. Para implementação numérica, estas séries são truncadas depois de alguns termos, e o número de termos é determinado pela precisão que desejamos alcançar. Então a idéia é agrupar os pontos dentro de certos domínios locais e aproximar todos os termos da *thin-plate splines* relacionados a esses centros como um todo, por uma única e curta expansão assintótica. Isto reduz substancialmente o custo computacional, porque os termos da *thin-plate splines* relacionados a cada centro não precisam ser computados um a um. Isto é particularmente pertinente no

caso de *thin-plate splines*, por exemplo, por causa do termo logarítmico que necessita ser avaliado repetidamente, o que é proibitivamente caro. O mesmo é verdade no cálculo de raízes quadradas quando multi-quádricas são usadas.

O terceiro método de aproximação faz o pré-condicionamento direto da matriz de interpolação fazendo um pré-condicionamento da matriz P (conforme equação 3.3) que é a idéia padrão. Métodos iterativos como Gauss-Seidel [16] ou gradiente conjugado podem ser aplicados.

3.9 Método da Partição da Unidade

A idéia básica do Método da Partição da unidade (*Partition of Unity Method*) é dividir o domínio dos dados em várias partes, aproximando os dados em cada subdomínio separadamente, e então “colar” as soluções locais de forma suave, usando funções de peso cujo somatório é 1 em todo o domínio. De modo mais preciso, consideremos um domínio com bordo Ω em \mathbb{R}^3 e um conjunto de funções não-negativas de suporte compacto $\{w_i\}$ tais que

$$\sum_i w_i \equiv 1 \text{ em } \Omega.$$

Denotando o suporte de w_i por $\text{supp}(w_i)$, associamos um conjunto de aproximações locais de funções V_i à cada subdomínio $\text{supp}(w_i)$. Assim, uma aproximação de uma função $f(x)$ definida em Ω é dada por

$$f(x) \approx \sum_i w_i(x) s_i(x), \quad (3.7)$$

onde $s_i \in V_i$.

Dado um conjunto de funções não-negativas com suporte compacto $\{W_i(x)\}$ tal que

$$\Omega \subset \bigcup_i \text{supp}(W_i),$$

as funções $\{w_i\}$ da partição da unidade podem ser geradas por

$$w_i(x) = \frac{W_i(x)}{\sum_{j=1}^n W_j(x)}. \quad (3.8)$$

Uma aproximação feita por meio das equações (3.7) e (3.8) satisfaz a principal característica dos métodos de elementos finitos da partição da unidade feitos por Babuska e Osborn [6], que lembra o método de Shepard modificado feito por Franke e Nielson [40]

3.9.1 Funções de peso

As funções de peso w_i estão relacionadas com a continuidade entre as funções locais f_i e a continuidade global da função de reconstrução f . As funções de peso descritas abaixo são as usadas inicialmente por Franke e Nielson [40], e posteriormente usada por Renka [87] onde denotaremos por c_i o centro do cubo que envolve o conjunto de pontos Ω_i e R_i o raio suporte de f_i .

$$w_i(x) = \left[\frac{(R_i - |x - c_i|)_+}{R_i |x - c_i|} \right]^2 \quad \text{onde} \quad (a)_+ = \begin{cases} a & \text{se } a > 0, \\ 0 & \text{caso contrário} \end{cases} \quad (3.9)$$

A Solução do sistema de equações (3.3) de N entradas requer $O(N^3)$ iterações e $O(N^2)$ posições de memória. Portanto fica claro que métodos diretos não podem ser aplicados para um número grande de pontos.

Com o método da partição da unidade, o domínio Ω é dividido em K subdomínios. Se supomos uma boa distribuição dos pontos, todos os w_i contêm N/K pontos, a nova solução terá $O(K(N/K)^3)$ iterações e $O((N/K)^2)$ posições de memória. Como N/K pode ser considerado como constante, a complexidade de reconstrução é $O(K)$, ou seja, linear em relação ao número de pontos. Outra vantagem deste método é a avaliação da função interpolante, pois os métodos que utilizam RBFs globais requerem $O(N)$ iterações para cada avaliação, enquanto que com funções de suporte local, é necessário apenas dois passos: primeiro encontramos todas as regiões que estão próximas ao ponto a ser

avaliado, e depois, avaliar a função nesta pequena região. Portanto o número de iterações será $O(K + N/K)$.

Capítulo 4

Deformações de Superfícies

Neste capítulo apresentamos um sumário dos recentes avanços em técnicas variacionais lineares de deformação de malhas e técnicas de deformação do espaço do objeto. Tais métodos foram desenvolvidos para a edição detalhada de malhas de alta resolução, geralmente obtidas a partir de objetos do mundo real por meio de um scanner tridimensional. Essas técnicas têm se tornado populares nos últimos anos devido a sua robustez, velocidade de processamento, resultados intuitivos e facilidade de controle, sendo útil em aplicações com interação com o usuário. Uma ferramenta de deformação intuitiva deve fornecer resultados fisicamente plausíveis e esteticamente aceitáveis das deformações da superfície e, adicionalmente, deve conservar seus detalhes geométricos. Os métodos de deformação de superfície recentemente pesquisados geralmente são formulados como um problema de otimização variacional global que representa as propriedades diferenciais da superfície editada. A robustez e eficiência são obtidas pela linearização de um funcional, de modo que o resultado da otimização global é obtido por meio da solução de um sistema de equações lineares esparsos.

Inicialmente discutiremos as técnicas baseadas na superfície do modelo destacando as representações baseadas no gradiente e representações baseadas no laplaciano. Depois fazemos um resumo das técnicas de deformação linear e não linear do espaço.

4.1 Representação Baseada no Gradiente

O trabalho pioneiro de Yu et al [114] usa a equação de Poisson como fundamentação teórica, no qual as modificações obtidas na malha são realizadas através da manipulação do campo gradiente. Eles consideram o gradiente das funções coordenadas da superfície x, y, z definidas no domínio Ω (que geralmente são os dados de entrada da superfície S). No caso contínuo, a superfície deformada é definida através das funções coordenadas x', y', z' que minimizam

$$\int_{\Omega} \|\nabla x' - g_x\|^2 dudv$$

(e o mesmo ocorrendo para y e z) sob alguma restrição de modelagem, onde $g_x = \nabla x$ são os gradientes das funções coordenadas originais da superfície. A equação de Euler-Lagrange desta minimização é a equação de Poisson

$$\nabla x' = \text{div } g_x. \quad (4.1)$$

Como a malha que representa a superfície de um modelo é um conjunto linear por partes e os gradientes das funções coordenadas são constantes em cada face, fica simples definir o gradiente das funções coordenadas neste conjunto discreto. Quando o domínio é a própria malha, então em cada triângulo o gradiente da função x é a projeção do eixo unitário $(1, 0, 0)^T$ sobre os planos dos triângulos, e similarmente para as outras duas coordenadas. De modo mais formal, consideremos uma função f linear por partes sobre o domínio da malha S definida pela interpolação baricêntrica dos vértices $f_i = f(v_i)$, tal que

$$f(u, v) = \sum_{i=1}^n f_i \Phi_i(u, v),$$

onde (u, v) são parâmetros sobre o domínio da malha e $\Phi_i(\cdot)$ são as funções de base lineares por partes associadas ao domínio de vértices da malha, ou seja, $\Phi_i(v_k) = \delta_{ik}$. O Gradiente de f é portanto

$$\nabla f(u, v) = \sum_{i=1}^n f_i \nabla \Phi_i(u, v). \quad (4.2)$$

Os gradientes $\nabla \Phi_i(u, v)$ são constantes em cada face do domínio da malha. Se (p_i, p_j, p_k) são vértices de um triângulo da malha, então os gradientes das funções correspondentes Φ_i, Φ_j, Φ_k são

$$(\nabla \Phi_i, \nabla \Phi_j, \nabla \Phi_k) = \begin{pmatrix} (p_i - p_k)^T \\ (p_j - p_k)^T \\ n^T \end{pmatrix}^{-1} \begin{pmatrix} 1 & 0 & -1 \\ 0 & 1 & -1 \\ 0 & 0 & 0 \end{pmatrix},$$

onde n é a normal unitária da face. Com esta formulação, fica assegurado que o gradiente está no plano da face triangular [15]. A equação (4.2) pode ser formulada usando um operador global G , escrito como uma matriz $3m \times n$ que multiplica o vetor f de n coordenadas dos valores discretos f_i afim de obter um vetor de m coordenadas gradientes, onde cada gradiente tem três coordenadas espaciais, e m é o número de triângulos. Assim, obtém-se a seguinte fórmula para a malha de entrada:

$$Gx' = g_x,$$

e o mesmo ocorrendo para as outras duas funções coordenadas.

No entanto, o resultado desta técnica de edição não é satisfatório, porque tenta conservar os gradientes originais da malha, com sua orientação no sistema de coordenadas global, ignorando o fato que na superfície deformada os gradientes devem ser rotacionados, visto que eles sempre estão nos planos dos triângulos, os quais se alteram com a deformação da superfície.

O *problema de transformação local* é crucial nas técnicas de edição diferenciais; isto ocorre porque a representação depende de uma posição particular da superfície no espaço; ou seja, não é rígida-invariante, e portanto quando a superfície deforma, a representação precisa ser atualizada. Infelizmente isso se torna problemático, pois a superfície deformada não é conhecida a priori.

4.2 Representação baseada no laplaciano

As técnicas baseadas na representação do laplaciano também são conhecidas como coordenadas diferenciais ou coordenadas Laplacianas [2, 97]. Tais coordenadas são obtidas aplicando o operador laplaciano aos vértices p_i da malha, conforme a Equação (B.2), obtendo o vetor curvatura normal (maiores detalhes no Apêndice B):

$$\delta_i = \nabla(p_i) = -2H_i n_i \quad (4.3)$$

onde H_i é o vetor curvatura média $H = \frac{k_1+k_2}{2}$ em v_i , como visto no Apêndice A.5.

As técnicas de deformação baseadas no operador laplaciano foram desenvolvidas paralelamente às técnicas baseadas em gradiente, e de forma semelhante, a obter a deformação minimizando diretamente a diferença entre as coordenadas de entrada da superfície δ_i . No caso contínuo a minimização da energia é formulada como

$$\min_{p'} \int_{\Omega} \|\Delta p' - \delta\|^2 dudv. \quad (4.4)$$

A equação de Euler-Lagrange obtida desta minimização é

$$\Delta^2 p' = \Delta \delta.$$

Quando esta equação toma valores no domínio dos parâmetros de entrada da superfície, o operador de Laplace coincide com o operador Laplace-Beltrami Δ_S obtendo a equação discretizada

$$L^2 p' = L \delta, \quad (4.5)$$

que pode ser separada em três componentes coordenadas. Uma formulação análoga foi usada para definir a técnica *Least-Squares Meshes* [96]. Apesar do sistema de equações ser bastante simples, podendo ser resolvido eficientemente por métodos diretos, os resultados visuais são satisfatórios apenas quando a superfície inicial é uma thin-plate ou membrana. Nos outros casos os detalhes são distorcidos pelos mesmos motivos da técnica

baseada no gradiente: o sistema tenta preservar a orientação dos vetores laplacianos com respeito ao sistema de coordenadas locais, quando na verdade deveriam ser rotacionados com a superfície deformada.

4.3 Deformação do espaço

Nas seções anteriores, todos os métodos baseadas em superfície computam um campo de deformação suave na superfície S . Nos métodos lineares, isto se resume a resolver um sistema laplaciano equivalente à equação diferencial parcial de Euler-Lagrange, de alguma energia quadrática, enquanto métodos não lineares minimizam energias de ordem superior usando técnicas como a de Newton ou Gauss–Newton. Uma desvantagem destes métodos é que sua eficiência e robustez dependem da complexidade e qualidade da malha da superfície. Na ocorrência de triângulos degenerados o operador laplaciano discretizado não está bem definido e conseqüentemente o sistema linear que o modela se torna singular.

Estes problemas são evitados com técnicas de deformação do espaço, que deformam o ambiente do espaço tridimensional e com isso deforma os objetos nele imersos (ver Figura 4.1). Diferentemente dos métodos baseados na superfície, as técnicas de deformação do espaço utilizam uma função de três variáveis $d : \mathbb{R}^3 \rightarrow \mathbb{R}^3$ para transformar todos os pontos da superfície original S na superfície modificada $S' = \{p + d(p) | p \in S\}$. Como a função de deformação d da representação da superfície não depende de uma representação particular da superfície, ela pode ser usada para deformar todo tipo de superfície representada explicitamente.

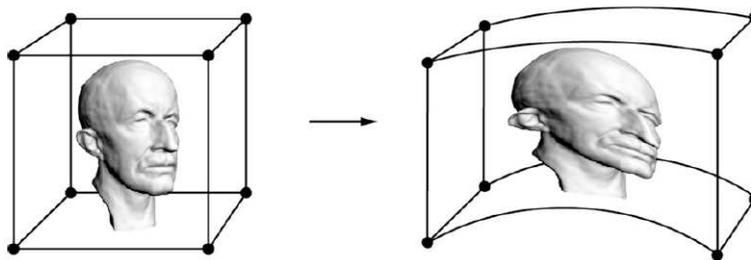


Figura 4.1: Deformação do espaço do objeto

A seguir, daremos uma visão geral de técnicas lineares de deformação do espaço.

4.3.1 Deformação de forma livre

Os métodos clássicos de deformações de forma livre (freeform deformation–FFD) [91] representam a deformação do espaço por um produto tensorial de funções Bézier ou spline

$$d(x, y, z) = \sum_i \sum_j \sum_k \delta c_{ijk} N_i^l(x) N_j^n(y) N_k^m(z).$$

Assim como nas splines de superfície, estas técnicas exigem interações complexas com o usuário e podem ocasionar problemas de *aliasing*. Para satisfazer as restrições de deslocamento, o método FFD inverso [48] resolve um sistema linear para os pontos de controle c_{ijk} para os movimentos requeridos, que também não obtém necessariamente uma boa deformação para baixas energias de curvatura.

4.3.2 Funções de base radiais

No caso das deformações baseadas na superfície, a boa qualidade dos resultados são obtidos pela interpolação de condições de contorno fornecidas pelo usuário através de uma função de distância $d : S \rightarrow \mathbb{R}^3$ que adicionalmente minimiza um funcional de energia. Isto deu motivação para se procurar interpolações suaves tridimensionais de funções de *deformação do espaço* $d : \mathbb{R}^3 \rightarrow \mathbb{R}^3$ que minimizam funcionais de energia similares.

Como visto no Capítulo 3 as funções de base radiais são bastante eficientes em problemas de interpolação de dados espalhados. Uma deformação RBF de três variáveis é definida em função dos centros $c_j \in \mathbb{R}^3$ e pesos $\lambda_j \in \mathbb{R}^3$ por

$$d(x) = \sum_j \lambda_j \phi(\|c_j - x\|) + p(x), \quad (4.6)$$

onde $\phi(\|c_j - \cdot\|)$ é a função base correspondente ao j -ésimo centro c_j e $p(x)$ é um polinômio de baixo grau usado para garantir precisão polinomial. Para construir uma interpolação RBF para as restrições $d(p_i) = d_i$, os centros são tipicamente tomados como

restrições ($c_i = p_i$) e um sistema linear é resolvido (como o da Equação 3.3) para encontrar os pesos λ_i e os coeficientes do polinômio $p(x)$.

4.4 Trabalhos relacionados

Nesta seção, nós listamos e discutimos vários trabalhos que empregam representação diferencial da superfície, e técnicas de deformação do espaço, incluindo as técnicas mais recentes de deformação não linear do espaço, destacando as principais diferenças entre elas.

O primeiro trabalho a usar coordenadas diferenciais para a edição de malhas foi desenvolvido por Alexa [1] que sugere o uso dos laplacianos originais da superfície com constantes de suavização. Como as transformações locais não são computadas de forma apropriada, esta técnica obtém bons resultados na edição de superfícies suaves sem detalhes, ou na realização de deformações que não envolvem rotações.

Em 2004, Lipman et al. [65] adicionaram uma rotação local estimada a fim de simplificar o paradigma de edição por laplaciano. Esta é computada a partir da solução de Alexa [1]. Eles mostram que suavizando a transformação estimada por uma vizinhança maior e utilizando uma média de pesos, os resultados da edição podem melhorar significativamente, embora haja um custo computacional maior para isto. Tal técnica é bem empregada em superfícies relativamente suaves sem protuberâncias acentuadas. Por outro lado, a suposição inicial de que a edição por laplaciano inicial se mantém não se sustenta, e a rotação estimada falha. Em particular, a técnica apresenta dificuldades com características da malha que não podem ser descritas como um campo de altura sobre a superfície suave.

Botsch et al. [15] propuseram uma técnica conceitualmente similar, que faz uma estimativa das rotações locais a partir da superfície base \mathcal{B} e sua versão deformada \mathcal{B}' , e então aplicam esta rotação para o gradiente da malha de entrada para a construção do

resultado final usando a técnica de Poisson.

Sorkine et al. [98] propõem utilizar a representação por laplaciano juntamente com a otimização de transformações implícitas, obtidas a partir do 1-anel da região de deformação. Esta técnica pode manipular superfícies mais complexas, com uma maior variedade de características; entretanto esta técnica é limitada ao intervalo de rotação permitido, visto que a aproximação linearizada da rotação local é válida apenas para ângulos pequenos. Na prática, rotações acima de $\frac{\pi}{2}$ podem ser realizadas [95]. Para grandes rotações, basta “quebrar” a técnica em vários passos, dividindo as rotações grandes em pequenas rotações. O método de Sorkine et al. [98] juntamente com otimização de transformações implícitas foram empregados por Igarashi et al. [56] para a manipulação de malhas triangulares 2D. Eles apresentam um sistema no qual o usuário pode deformar e movimentar a malha sem a necessidade de estabelecer previamente um esqueleto do objeto ou deformação de forma livre (freeform deformation – FFD). O usuário move vários vértices da malha como *handle* e o sistema calcula as posições dos vértices livres restantes para reduzir a deformidade de cada triângulo. Para alcançar iterações em tempo real, o algoritmo é dividido em dois passos. O primeiro passo acha uma rotação apropriada para cada triângulo e o segundo passo ajusta sua escala. A idéia principal é usar métricas de erro quadráticos de modo que cada problema de minimização seja um sistema de equações lineares. Depois de resolver as equações simultâneas no início da interação, as posições de vértices livres são encontradas durante a manipulação interativa. Esta técnica permite rotações maiores que as de Sorkine et al.[98], mas necessita de um cálculo mais refinado da média relativa aos termos da transformação; entretanto, a formulação da transformação implícita pode ser definida pelo 1-anel plano, no qual a perturbação é requerida.

A edição baseada no laplaciano foi posteriormente usada para sistemas de edição baseadas em traços [79] e deformações volumétricas [116]. Nealen et al. [79] empregam a propagação de transformações implícitas e propõem o uso de traços sobre a superfície

como *handles* e para definir as restrições de deformação. Nos editores clássicos que utilizam deformação por *handle*, a posição do *handle* é manipulada diretamente pelo usuário, e portanto, fornece restrições rígidas de posicionamento; no caso de sistemas baseados em traços, restrições flexíveis são geralmente vantajosas, visto que o usuário pode introduzir traços imprecisos que apenas sugerem a forma desejada, mas não o especificam de forma exata. Zhou et al. [116] usam uma variante da propagação da transformação geodésica para obter deformações baseadas em traços, combinando com o laplaciano a edição de um gráfico volumétrico. A utilização do laplaciano no gráfico volumétrico, cria conexões entre os pontos distantes na superfície e assim tende a melhorar o volume do modelo.

Yu et al. [114] propõem a representação baseada no gradiente para a edição de malhas, combinada com a propagação geodésica da transformação local. Zayer et al. [115] trocam a propagação geodésica pela interpolação harmônica e mostram que esta mudança fornece transformações locais suavemente distribuídas, e conseqüentemente melhores resultados. Popa et al. [85] generalizam a propagação harmônica para uma transformação que depende do material. Diferentemente das técnicas citadas anteriormente, estes métodos só trabalham quando um *handle* de transformação é especificado adicionalmente para a translação (não sensíveis a translação). A representação baseada no gradiente foi recentemente relatada por Sumner et al. [99] para a transferência de uma seqüência de deformações de uma malha em uma outra.

Lipman et al. [66] desenvolveram a representação baseada em quadros e usa esta técnica para a edição e interpolação de superfícies. Esta técnica emprega uma representação rígida-invariante, onde as restrições de transformações locais são obtidas explicitamente por meio de otimização. As restrições de deformação podem realizar rotações acima de π ainda preservando os detalhes, tendo também a limitação de não ser sensível a translações, já que a solução para os quadros é decomposta com relação as restrições de posição. Deste modo podem ser especificadas explicitamente restrições rotacionais para os quadros

de *handle*.

Fu et al. [41] apresentam um sistema baseado em um laplaciano implícito linear, que captura efetivamente as informações da rotação local durante a edição. Primeiramente é computada uma transformação afim definida implicitamente por todas as coordenadas Laplacianas por intermédio da solução de vários sistemas lineares esparsos, e em seguida são extraídas as informações de rotação e escala de cada transformação afim resolvida. O sistema produz deformações visualmente coerentes para rotações com ângulos grandes.

Nealen et al. [78] apresentam o sistema *FiberMesh* para a modelagem de superfícies à mão livre a partir de uma coleção de traços desenhados pelo usuário, de modo que as curvas que determinam a superfície podem ser editadas, e desta forma modifica a superfície do modelo. O algoritmo de deformação das curvas usa uma variante do método de deformação com preservação de detalhes pelo uso de coordenadas diferenciais [95] combinado com o método co-rotacional [35]. O modelo é representado usando coordenadas diferenciais, e o resultado final é obtido por meio de uma sequência de soluções de problemas lineares de mínimos quadrados. Para computar de forma apropriada as rotações para as coordenadas diferenciais, é usada uma técnica similar a usada por Sorkine et al. [98] e Fu et al. [41], sendo que as matrizes de rotação são representadas explicitamente por variáveis livres separáveis.

PriMo [13] é uma versão não linear de deformação por minimização variacional de energia. A superfície é modelada como se fosse uma fina camada formada por prismas triangulares, os quais são unidos por uma energia elástica não linear. Durante a deformação os prismas continuam rígidos, o que possibilita uma otimização geométrica extremamente robusta.

Sheffer e Kraevoy [93, 63] consideram as coordenadas piramidais como uma versão não linear das coordenadas Laplacianas, tornando as coordenadas diferenciais invariantes por movimentos rígidos, sendo usado para deformações e *morphing*.

Huang et al. [53] aplicam uma versão não linear do gráfico volumétrico laplaciano, que também preserva características não lineares das restrições de preservação do volume. Para melhorar a performance e eficiência eles usam uma técnica de subespaço: a malha original é imersa em uma malha de controle, e a otimização é realizada nesta malha enquanto as restrições da malha original são consideradas um problema de mínimos quadrados.

Botsch e Kobbelt [12] propõem um método de mínimos quadrados incremental que essencialmente resolve de modo eficiente um sistema linear até um limite de erro especificado. A utilização deste método para pré-computar as funções base de deformação permite a deformação interativa de modelos complexos. Além disso, as avaliações destas funções base são feitas através de placas gráficas acelerando o processo e obtendo deformação do espaço em tempo real. Entretanto, para as deformações do espaço aqui discutidas, a superfície deformada S' depende linearmente das restrições de deslocamento d_i , e conseqüentemente, efeitos não lineares como rotação local de detalhes não podem ser realizadas, similarmente aos métodos lineares baseados em superfície. Apesar de as técnicas de deformação do espaço poderem ser melhoradas por técnicas de multi-resolução [69], elas ainda sofrem as limitações das técnicas lineares, as quais levaram ao desenvolvimento de técnicas não lineares de deformação de espaço.

Sumner et al. [100] realizam a deformação do espaço com preservação de detalhes através de uma coleção de deformações afins organizadas em uma estrutura de grafo. A cada nó do grafo é associada uma transformação que deforma uma região do espaço. As restrições de posicionamento são especificadas nos pontos do objeto imerso. Como o usuário manipula as restrições, um problema de minimização não linear é resolvido para encontrar valores ótimos para as transformações afins. A preservação de características é codificada diretamente na função objetiva pela avaliação da variação de cada transformação a partir de uma rotação real. Além de geometria estática, este método pode ser

aplicado em animações de malhas e sistemas de partículas animados.

Botsch et al. [14] fazem uma extensão do sistema *PriMo* [13] para a deformação de objetos sólidos. O modelo de entrada é representado de forma adaptável por *voxels* e as células hexaedrais resultantes se mantêm rígidas sob deformações para garantir robustez numérica. A deformação é governada por uma energia elástica não linear unindo células rígidas vizinhas.

Uma outra classe de técnicas usa campos de vetores de divergência livre para deformar objetos [4, 106]. A vantagem destas técnicas é que por construção elas produzem deformações livres de intersecções, além de preservarem o volume. Uma desvantagem é a dificuldade em construir campos de vetores que satisfaçam exatamente as restrições de deformação definidas pelo usuário.

Capítulo 5

Sistema para Modelagem e Deformação Baseado em Traços

Neste capítulo, apresentamos o protótipo desenvolvido para a criação e deformação de objetos tridimensionais a partir de traços bidimensionais desenhados pelo usuário.

5.1 Criação

Existem vários paradigmas para a criação de objetos tridimensionais. No contexto da modelagem de objetos implícitos, podemos separá-los em dois grupos: construtivos e à “mão-livre”.

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.

Uma maneira de se obter um objeto 3D à mão-livre é desenhar a silhueta do objeto através de um traço bidimensional, e em seguida, inflá-lo, obtendo um objeto 3D conforme mostra a Figura 5.1.

Dado um traço fechado 2D que representa a silhueta de um objeto 3D, existem várias técnicas para inflá-lo. Um modo de se fazer isto é obter a partir do traço 2D duas regiões planas R e $R' \in \mathbb{R}^2$ com bordo delimitada pelo traço, de modo que seus bordos coincidam. Como R e R' estão no plano da tela (plano xy), para gerar uma altura, alguns pontos do interior de cada região sofrem um acréscimo d na coordenada z na direção

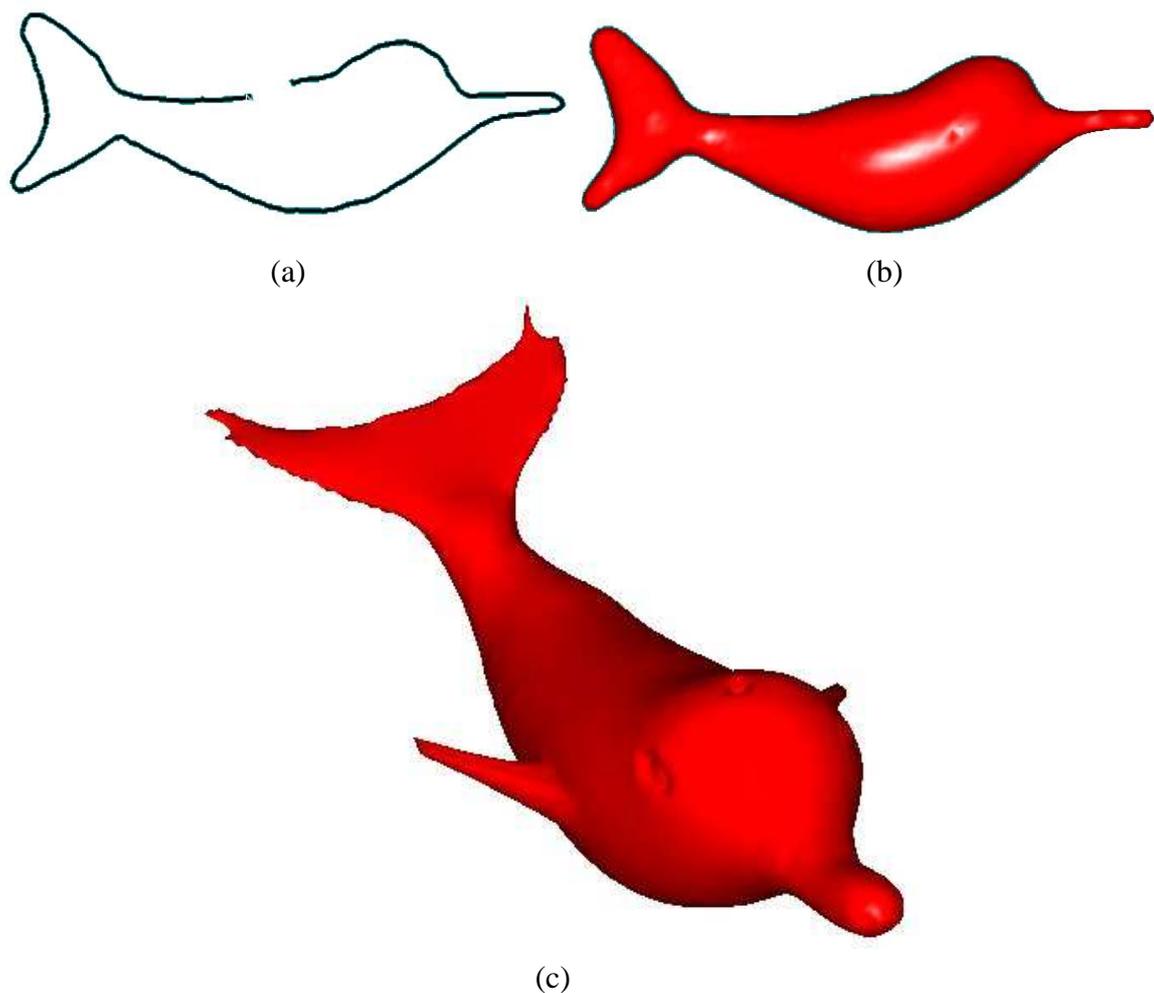


Figura 5.1: Criação de um modelo. (a) O usuário desenha a silhueta do objeto através de um traço. (b) O objeto é inflado. (c) O objeto visto de outro ângulo no qual foram acrescentadas características localizadas dando ao modelo um perfil mais expressivo.

normal à região, obtendo-se os pontos da superfície 3D que serão interpolados por uma função f que representa implicitamente esta superfície. Em outras palavras, a geometria para a superfície abstrata pode ser realizada fazendo corresponder ao gráfico das funções $f, -g : S \rightarrow \mathbb{R}$ dado por $f(x) = g(x) = d(x, \partial S)$, a distância de x ao bordo de R . Alternativamente, f e g podem ser funções contínuas quaisquer tais que se anulem exatamente no bordo de R .

Uma realização geométrica deste tipo produz apenas superfícies C^0 . Para geometrias suaves, nós podemos realizar um procedimento em duas etapas: Uma superfície linear por partes é gerada e então os dados obtidos são interpolados por uma função implícita.

Para o primeiro passo é preciso fazer uma triangulação de R com vértices internos (e arestas). Valores não nulos são atribuídos para os vértices internos que podem ser obtidos automaticamente. Para evitar objetos com artefatos indesejáveis ou com auto-interseções, é necessário que a curva de entrada apenas represente a geometria de R .

O critério de escolha dos pontos interiores às regiões depende do tipo de superfície que se pretende obter após a inflação. Resultados bastante intuitivos e próximos de objetos do mundo real são obtidos escolhendo pontos pertencentes ao Eixo Medial ou Eixo Cordal [86] de cada região, e tomar o acréscimo d como sendo a distância do ponto ao bordo de R . Este é o método usado por Igarashi et al. [55] e já adotado por Parari e Esperança [81].

5.1.1 Geração da superfície linear por partes usando o eixo cordal

Nós obtemos um primeiro esboço da geometria da superfície linear por partes através do uso do eixo cordal da região R .

O Eixo Medial (MA) é o conjunto de centros das circunferências maximais em R , onde uma circunferência é maximal em R se ela não está completamente contida em uma outra circunferência no interior de R . Seja S uma circunferência máxima de R , uma corda C de R é dita uma *corda de tangência máxima* (MCT) se ao menos um dos arcos subentendidos pela corda é livre de pontos de tangência com o bordo de R . A Figura 5.2 mostra alguns exemplos de MCTs.

O Eixo Cordal (CA) de S é o conjunto de todos os pares (ρ, δ) , onde ρ e δ são respectivamente o ponto médio e a metade de uma MCT, ou são o centro e o raio de um círculo máximo que tem pelo menos três MCTs (ver Figura 5.2).

Portanto, de posse da região R determinada pelo traço que representa a silhueta do modelo 3D, e do eixo cordal (ou medial), pode-se obter uma superfície do modelo.

Note que o eixo cordal pode ser determinado de forma interativa com o usuário ou computado automaticamente. Nós geramos automaticamente o eixo cordal utilizando uma triangulação de Delaunay \mathcal{T} , restrita à curva de entrada na qual as faces de \mathcal{T} são

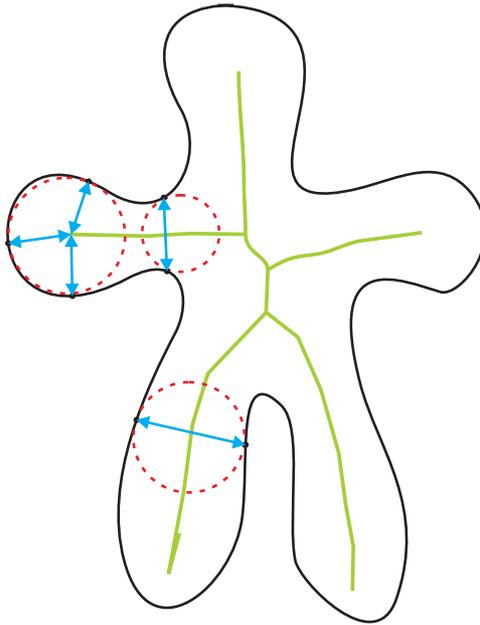


Figura 5.2: Três círculos máximos (vermelho) na região e seus respectivos MCTs (azul), e o eixo cordal (verde).

classificadas em faces terminais (triângulos com duas arestas adjacentes a R), faces de borda (triângulos com uma aresta adjacente a R), faces de junção (triângulos sem arestas adjacentes a R) e faces externas (triângulos que se encontram fora de R). Veja a Figura 5.3(b).

O processo de obtenção do eixo cordal 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 5.3(c)). Para cada ramo de CA , e começando numa face terminal, faz-se a identificação de triângulos irrelevantes, isto é, aqueles que estão contidos no círculo cujo diâmetro é igual ao comprimento da última aresta, não restrita, visitada neste processo. A espinha de S é constituída por todos os vértices de CA , exceto aqueles cujas arestas pertencem somente a triângulos irrelevantes (Figura 5.3(d)).

As arestas da espinha são passadas por um processo de suavização e todos os vértices da espinha são inseridos na triangulação; em seguida, são elevados a uma altura proporci-

onal à distância média entre ele e os seus vértices externos diretamente conectados sendo a triangulação resultante simplificada com base num processo de remoção de arestas internas de comprimento menor que t_1 (t_1 é um valor heurístico que pode ser tomado como um percentual da média dos comprimentos das arestas dos triângulos). Veja Figura 5.3(e).

5.1.2 Obtenção da superfície suave

Para obter a superfície suave a partir de uma superfície linear por partes inicial, faz-se necessário o uso de algum tipo de interpolação de dados. Aqui isso é obtido pela implicitização da superfície por meio de funções de bases radiais (RBFs). Dados um conjunto $X = \{x_1, \dots, x_N\}$, uma seqüência de N valores s_i , um polinômio $p(x)$ e uma função $\phi : \mathbb{R} \rightarrow \mathbb{R}$, uma interpolação dos dados em X pode ser obtida como a solução, para os λ_i , do sistema linear de equações:

$$\begin{aligned} s(x) &= p(x) + \sum_{i=1}^n \lambda_i \phi(\|x - x_i\|) \\ s(x_i) &= s_i, \quad i = 1, 2, \dots, N \end{aligned} \tag{5.1}$$

Denotando por p_i o vértice da superfície de aproximação obtida na seção anterior, a cada vértice p_i , estimamos um vetor unitário normal N_i como a média das normais das faces adjacentes à p_i e construímos uma nova superfície fazendo $q_i = p_i + tN_i$ ($t = 1$ em nossa implementação). Esta nova superfície nos permite discriminar o interior da superfície, onde fazemos $s(p_i) = 0$ e $s(q_i) = 1$. O Sistema 5.1 é então resolvido utilizando o pacote LAPACK [51] e a superfície-zero é poligonizada através do algoritmo de Bloomenthal [10] estendido pelo método [81].

5.2 Transformações geométricas

Uma transformação geométrica muda a forma do objeto sem afetar sua topologia. Uma ferramenta de modelagem deve permitir a aplicação de transformações sobre o objeto de forma rápida e intuitiva. Além disso, são de particular interesse as transformações de

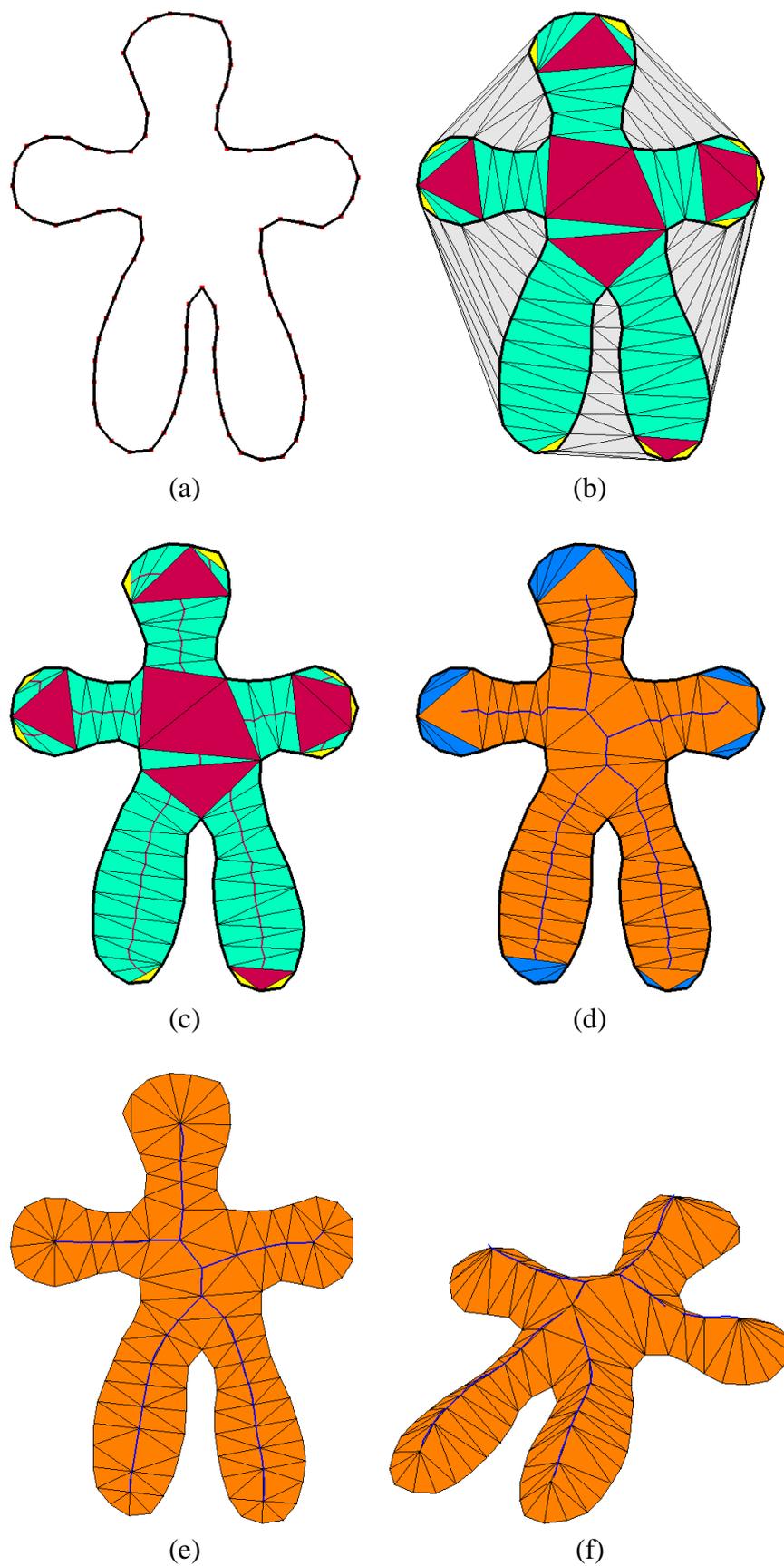


Figura 5.3: (a) Polígono de entrada. (b) Triangulação de Delaunay restrita e classificação dos triângulos. (c) Cálculo do eixo cordal. (d) Determinação da espinha. (e) Remoção de arestas pequenas. (f) Triangulação com elevação da espinha.

deformação que se caracterizam por uma certa coerência física além de permitir grande variabilidade de formas.

Alcançar todas estas propriedades simultaneamente se torna uma tarefa difícil, pois às vezes tais propriedades se tornam incompatíveis. Para satisfazer algumas destas propriedades na deformação nós utilizamos técnicas que realizam a minimização de um funcional de energia [11, 12].

Por motivos computacionais, utilizamos apenas funcionais de energia quadráticos, visto que as equações de Euler-Lagrange são lineares, mas de um modo geral, outros funcionais de energia podem ser utilizados.

Denotemos por S uma superfície compacta. Uma transformação geométrica (ou deformação) é uma função $p \mapsto p + d(p)$, onde $d(p)$ é um vetor de deslocamento. Tomando por exemplo o comprimento e o suporte de d pequenos, pode-se obter transformações bem próximas da superfície original. Usualmente, as transformações de superfícies são obtidas dividindo-se a superfície em três regiões: componente estática, componente livre e componente rígida [11] (veja a Figura 5.4).

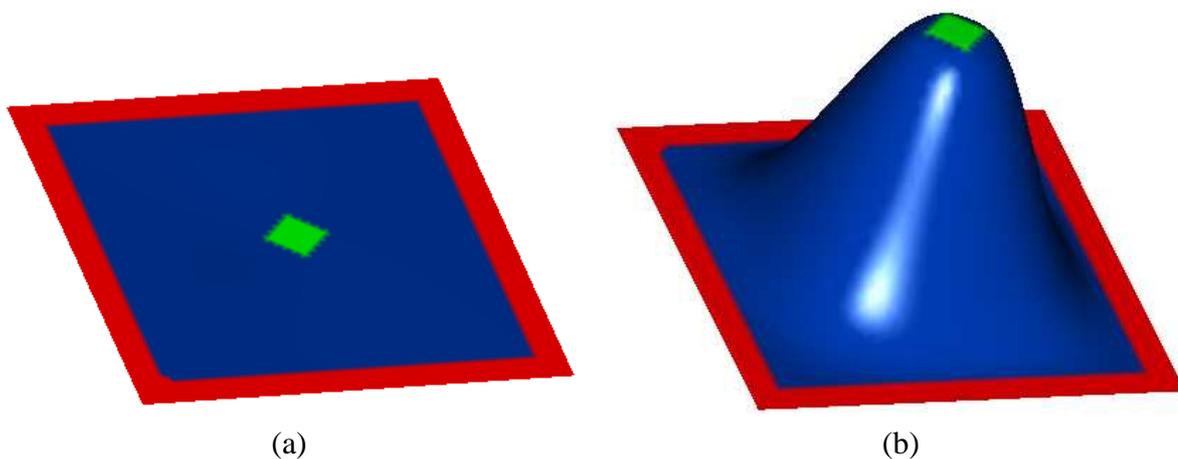


Figura 5.4: (a) Seleção das regiões: pontos fixos (vermelho), pontos de controle (verde) e pontos livres (azul). (b) a região de controle foi transladada para cima, e os pontos livres foram realocados segundo o funcional de energia (5.3) com $k = 2$.

Denotando por F , H e P as regiões estática, rígida e livre respectivamente, tem-se que a região F não é modificada pela deformação. A região H será transformada rigidamente

(composição de rotações e/ou translações) mantendo sua conectividade com a região P . A deformação dos pontos P dependem da transformação sofrida pela região H e devem minimizar a um funcional de energia.

Denotemos por $S \subset \mathbb{R}^3$ uma superfície, cuja parametrização é dada por $p : \Omega \subset \mathbb{R}^2 \rightarrow S \subset \mathbb{R}^3$. Essa superfície será deformada em S' adicionando a cada ponto $p(u, v)$ um vetor deslocamento $d(u, v)$, tal que $S' = p'(\Omega)$ com $p' = p + d$.

Sabe-se da geometria diferencial [27] que a primeira e segunda forma fundamental, $I(u, v), II(u, v) \in \mathbb{R}^{2 \times 2}$, pode ser usada para medir propriedades geométricas intrínsecas de S , tais como comprimentos, áreas e curvaturas. Alterações nas formas fundamentais fornecem uma medida de deformação [101]:

$$E(S') = \int_{\Omega} k_e \|I' - I\|_F^2 + k_d \|II' - II\| dudv, \quad (5.2)$$

onde I' e II' são as formas fundamentais de S' , $\|\cdot\|$ denota a norma de Frobenius, e os parâmetros de rigidez k_e e k_d são usados para controlar a resistência de elasticidade e dobra da deformação.

Quando realizamos uma deformação interativa, S' tem que ser recomputada por E toda vez que o usuário movimenta a região rígida H . Portanto, esta minimização não linear é muito cara para realizar deformações interativas. Por isso, ela é simplificada substituindo as primeiras e segundas formas fundamentais pelas primeiras e segundas derivadas parciais da função de distância d [107]:

$$\tilde{E} = \int_{\Omega} k_e (\|d_u\|^2 + \|d_v\|^2) + k_d (\|d_{uu}\|^2 + 2\|d_{uv}\|^2 + 2\|d_{vv}\|^2) \quad (5.3)$$

onde $d_x = \frac{\partial}{\partial x}d$ e $d_{xy} = \frac{\partial^2}{\partial x \partial y}d$.

A minimização da Equação 5.3 pode ser realizada de modo eficiente por meio do cálculo diferencial, o qual nos fornece a equação diferencial parcial também conhecida

como equação de Euler-Lagrange:

$$-k_e \Delta d + k_d \Delta^2 d = 0, \quad (5.4)$$

que caracteriza a minimização de (5.3). Δ e Δ^2 representam os operadores laplaciano e bi-laplaciano, respectivamente:

$$\begin{aligned} \Delta d &= \operatorname{div} \nabla d = d_{uu} + d_{vv}, \\ \Delta^2 d &= \Delta(\Delta d) = d_{uuuu} + 2d_{uuvv} + d_{vvvv}. \end{aligned} \quad (5.5)$$

Portanto, S' pode ser encontrada diretamente através da solução de (5.4), sujeita a restrições de contorno.

Para que mudanças nas derivadas segundas em (5.3) sejam correspondentes a mudanças na curvatura da superfície, a parametrização p pode ser escolhida o mais isométrica possível. Como consequência, tipicamente Ω é escolhido como sendo a própria superfície inicial S . Isso é conceitualmente similar aos funcionais de Greiner et al. [44].

Como consequência, o operador laplaciano Δ com respeito à parametrização p se torna o operador Laplace-Beltrami $\Delta_S = \operatorname{div}_S \nabla_S$ com respeito à superfície S [27]:

$$-k_e \Delta_S d + k_d \Delta_S^2 d = 0. \quad (5.6)$$

Esta minimização variacional é recentemente relatada em técnicas de modelagem variacionais [107], [76], onde a área e a curvatura da superfície são minimizadas em função destas mudanças. A energia linearizada tipo *membrana* e *thin-plate* correspondente à Equação 5.3 são definidas como

$$\begin{aligned} \tilde{E}_{memb} &= \int_{\Omega} \|p_u\|^2 + \|p_v\|^2 dudv, \\ \tilde{E}_{plate} &= \int_{\Omega} \|p_{uu}\|^2 + 2\|p_{uv}\|^2 + 2\|p_{vv}\|^2 dudv. \end{aligned} \quad (5.7)$$

Analogamente à Equação 5.6, as equações de Euler-lagrange correspondentes são $-\Delta_S p = 0$ e $\Delta_S^2 p = 0$, respectivamente.

Nós implementamos soluções baseadas no laplaciano discreto (geometria diferencial discreta) e interpolação de soluções por funções de bases radiais (métodos de elementos finitos), ver [12].

No contexto da deformação pela técnica de discretização do operador laplaciano, para computar a solução do problema de otimização, as equações de Euler-Lagrange podem ser escritas na forma matricial determinando o sistema linear esparso:

$$\begin{pmatrix} \Delta^k & & \\ 0 & I_F & 0 \\ 0 & 0 & I_H \end{pmatrix} \begin{pmatrix} P \\ F \\ H \end{pmatrix} = \begin{pmatrix} 0 \\ F \\ H \end{pmatrix} \quad (5.8)$$

onde Δ é uma discretização do operador laplaciano de ordem k (detalhado no Apêndice B), $P = (p_1, \dots, p_n)^T \in \mathbb{R}^{n \times 3}$ é o vetor de pontos livres na região suporte, $F = (f_1, \dots, f_l)^T \in \mathbb{R}^{l \times 3}$ são os pontos fixos fora da região suporte, e $H = (h_1, \dots, h_m)^T \in \mathbb{R}^{m \times 3}$ são os pontos da região de controle (handle). F e H determinam as condições de contorno do sistema e apenas $k + 1$ anéis dos pontos fixos são necessários para determinar as C^{k-1} condições de contorno. O laplaciano de ordem k é definido recursivamente por

$$\begin{aligned} \Delta^k S(x) &= \Delta(\Delta^{k-1} S(x)) \\ \Delta^0 S(x) &= S(x) \end{aligned} \quad (5.9)$$

Denotando por L^1 a primeira matriz do lado esquerdo da Equação 5.8 para $k = 1$, observamos que L^1 é uma matriz simétrica dada por

$$L_{ij}^1 = \begin{cases} 0, & i \neq j, x_j \notin N_1(x_i) \\ \cotg \alpha_{ij} + \cotg \beta_{ij}, & i \neq j, x_j \in N_1(x_i) \\ -\frac{2}{A_v(x_i)} \sum_{x_j \in N_1(x_i)} (\cotg \alpha_{ij} + \cotg \beta_{ij}) & i = j \end{cases}$$

onde $N_1(x_i)$ representa a 1-vizinhança de x_i , isto é, os vértices que possuem aresta comum a x_i .

Com respeito ao método dos elementos finitos, depois de determinadas as regiões fixa, livre e de manipulação, uma função de deformação RBF $d : \mathbb{R}^3 \rightarrow \mathbb{R}^3$ que associa a cada ponto $p \in \mathbb{R}^3$ na malha um vetor deslocamento $d(p)$ tal que o ponto resultante após a modificação é dado por $p' = p + d(p)$

Este deslocamento é suavemente interpolado por uma RBF da forma

$$d(x) = \sum_{j=1}^m \lambda_j \phi_j(x) + p(x), \quad (5.10)$$

onde $\phi_j(\cdot) = \phi(\|c_j - \cdot\|)$ é a função base correspondente ao j -ésimo centro c_j e $p(x)$ é um polinômio de grau baixo para garantir precisão polinomial, de modo que $\forall f \in F, h \in H$

se tenha $d(f) = f, d(h) = h'$. Combinando estas restrições em um conjunto $d(x_i) = b_i$, para $1 \leq i \leq m$, e selecionar os centros da RBF como $c_i = x_i$ leva a um sistema linear simétrico

$$\begin{pmatrix} \Phi & P \\ P^T & 0 \end{pmatrix} \begin{pmatrix} \lambda_j \\ q_j \end{pmatrix} = \begin{pmatrix} b_i \\ 0 \end{pmatrix} \quad (5.11)$$

onde $\Phi \in \mathbb{R}^{m \times m}$ é definida por $\Phi_{ij} = \phi_j(c_i)$ e $P \in \mathbb{R}^{m \times q}$ é definida por $P_{ij} = p_j(c_i)$, q_j são os coeficientes do polinômio p e q é a quantidade de elementos da base do espaço dos polinômios $P_n^3 \ni p$, onde P_n^3 denota o espaço dos polinômios em três variáveis de ordem n , isto é, de grau menor ou igual a $n - 1$.

Resolvendo o sistema (5.11) em função das novas posições dos pontos $h' \in H'$ e dos pontos $f \in F$ obtém-se λ_j e q_j , e conseqüentemente os novos pontos $p' \in P'$ respeitando a um funcional de energia do tipo (5.3).

A Matriz P e portanto o sistema completo é singular se todas as restrições c_i estão sobre uma quádrlica [74]. Neste caso omite-se o polinômio, o qual deixa de ter grande influência e as deformações continuam a ser de boa qualidade. Por simplicidade de notação será omitida a parte polinomial de (5.11) em todas as discussões seguintes e focaremos apenas o bloco superior esquerdo $m \times m$, o qual denotamos por

$$\Phi.W = \begin{pmatrix} F \\ H' \end{pmatrix} \quad (5.12)$$

com pesos $W = (\lambda_1, \dots, \lambda_m)^T$, vértices fixos $F = (f_1, \dots, f_f)^T$ e vértices de manipulação $H' = (h'_1, \dots, h'_h)^T$. A generalização para o sistema completo (5.11) é direta.

Afim de evitar que os sistemas (5.8) e (5.11) sejam inteiramente computados toda vez que os pontos de H são modificados, o que acarretaria em um acréscimo no custo computacional, as funções de base são pré-computadas, as quais correspondem ao grau de liberdade da região de manipulação. As soluções de (5.8) e (5.11) podem ser explicitamente expressas por suas inversas e as transformações podem ser expressas por

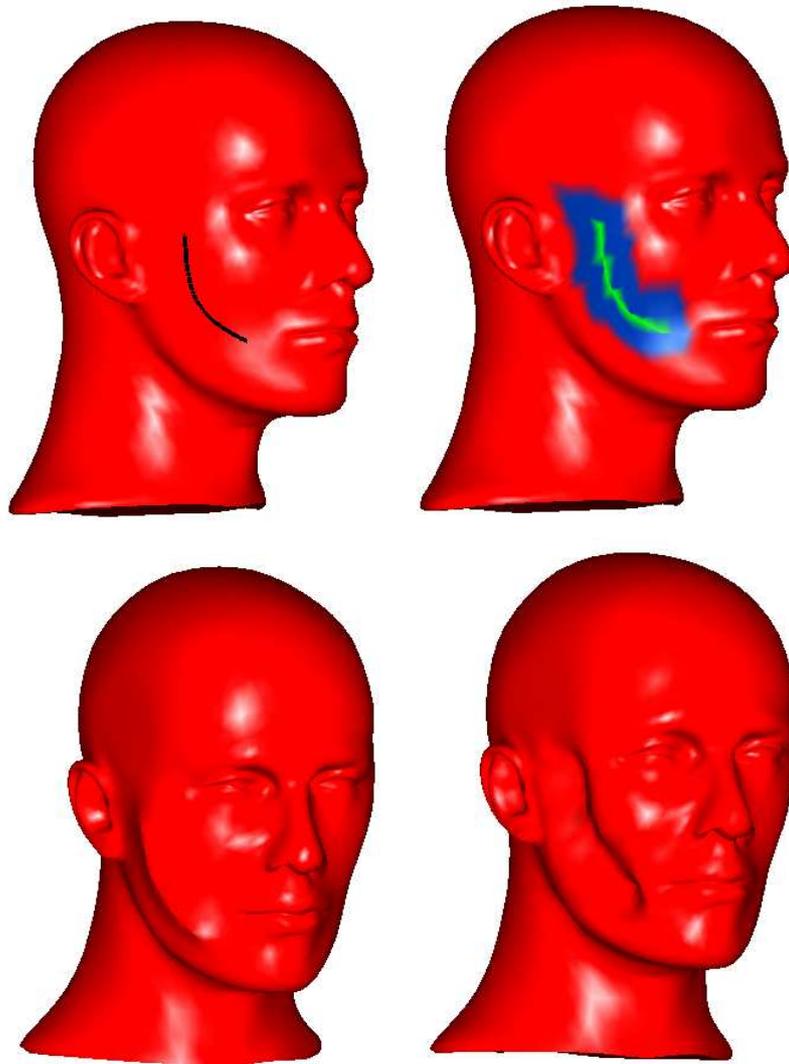


Figura 5.5: Inclusão de características afiadas e suaves: (acima-esquerda) o traço inicial; (acima-direita) as regiões fixa, livre e rígidas são automaticamente determinadas pelo sistema; (abaixo-esquerda) característica afiada adicionada ao modelo; (abaixo-direita) característica suave adicionada ao modelo.

$$P' = P + B \underbrace{(\delta h_1, \delta h_2, \delta h_3, \delta h_4)^T}_{=: \delta C}, \text{ onde } h_1, h_2, h_3 \text{ e } h_4 \text{ são quatro pontos}$$

da região H . A pré-computação das funções de base é tratada com mais detalhes na Seção 6.2.4.

Capítulo 6

Detalhes de Implementação

6.1 Determinação das regiões de deformação

A determinação das regiões P , F e H da superfície pode ser feita de duas formas. Na primeira, o usuário pinta diretamente os vértices da superfície, de acordo com a região de interesse escolhida, através da utilização de um algoritmo de pintura. Para isto, o usuário pode escolher entre dois instrumentos de pintura: a caneta e o quadrilátero. O primeiro tem um funcionamento similar a uma caneta, visto que tem um pequeno alcance pintando apenas os vértices por onde o ponteiro do mouse passa. Já o instrumento de pintura quadrilátero pinta toda a região compreendida em um quadrilátero que é definido também arrastando o mouse.

O usuário pode também determinar as regiões fazendo uso de traços. Depois de posicionar adequadamente o modelo, este desenha um traço sobre a sua superfície para definir automaticamente a região do modelo que será deformada.

Finalmente, após a determinação de regiões, o usuário pode explorar e interagir com a *deformação* do modelo. Uma mudança na posição dos pontos H fará com que a região conformada pelos vértices P sejam reposicionados seguindo um perfil “suave”. Para isto, é criado um triedro (três vetores ortonormais de mesma origem) posicionado no centro de massa da região H definido por quatro pontos. Um desses é a origem comum dos vetores e os outros três pontos são as extremidades de cada vetor do referido triedro. A

translação e/ou rotação do triedro, definem a translação e/ou rotação da região H , visto que os quatro pontos do triedro original e os quatro pontos do novo triedro determinam uma transformação rígida, de modo que esse triedro funciona como um handle de manipulação. O sistema permite mudar a posição de H através da manipulação do handle ou pela descrição dada por um traço.

O usuário pode eleger o método no qual a deformação será baseada (baseada no laplaciano ou em RBF) e modificar a suavidade da deformação em ambos os métodos. A recíproca também é permitida.

Além das deformações o sistema também permite inserir características localizadas nos modelos, ou seja, características afiadas e suaves. Por exemplo, para inserir uma característica afiada numa superfície, o usuário simplesmente faz um traço indicando a região da superfície que ficará afiada. A suavidade desejada e a magnitude do vinco poderão ser controladas pelo usuário. Na verdade, a determinação das regiões e a deformação da região P após uma mudança na posição de H estão baseadas no mesmo paradigma de deformação.

A Figura 6.1 ilustra uma sessão simples de modelagem feita com a ferramenta apresentada.

6.2 Algoritmos

Nesta Seção apresentamos as idéias principais usadas na implementação de nosso protótipo pra modelagem e deformação de superfícies.

6.2.1 Determinação das regiões através de traços

Uma vez que o usuário desenha um traço no plano da tela, o sistema projeta o traço na superfície do modelo. A partir deste traço determinamos dois vetores no plano da tela gerados pelo primeiro e pelo último segmento do conjunto de pontos que representa o traço inicial. Tais vetores determinam dois planos perpendiculares ao plano da tela. Cada plano

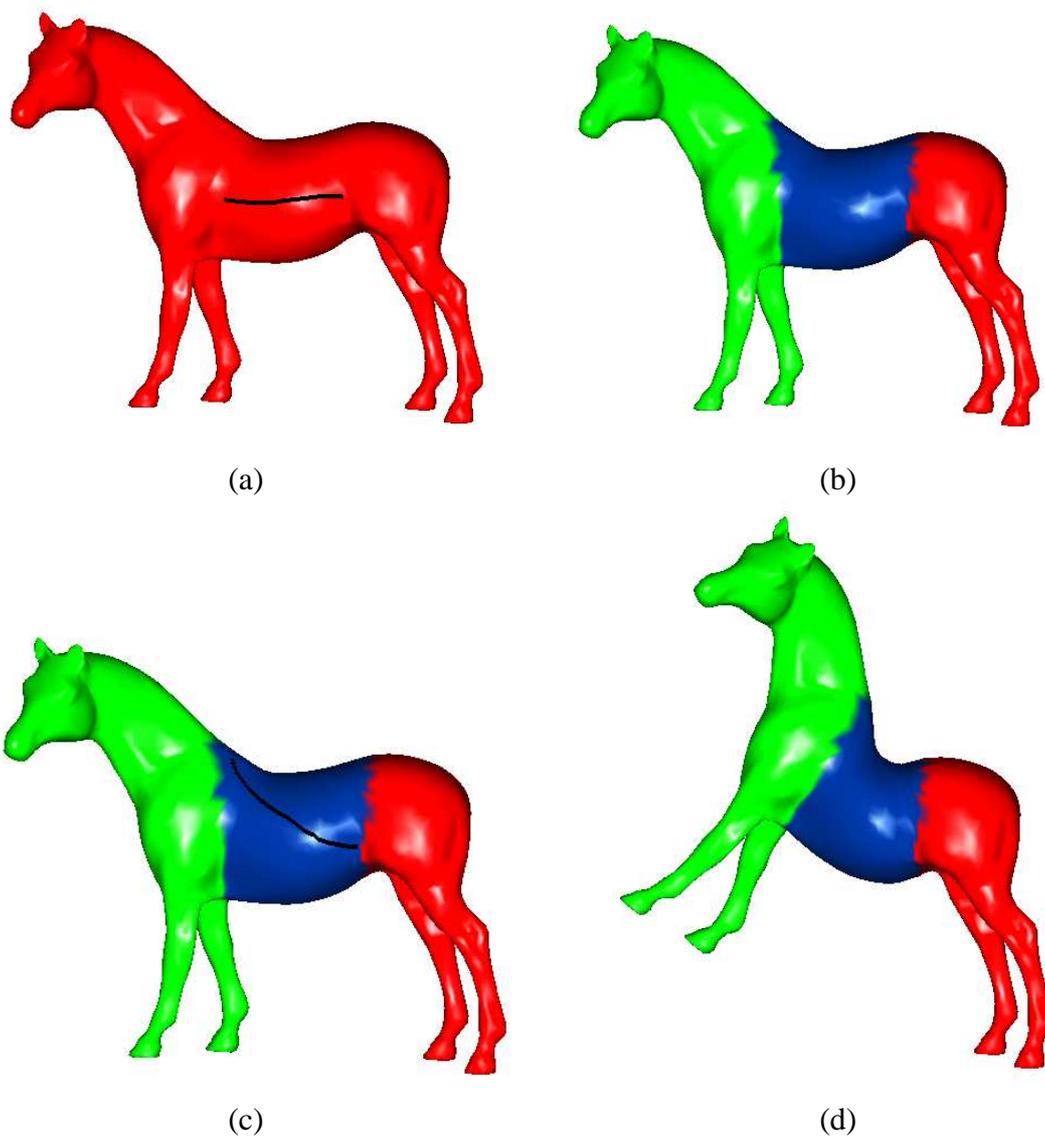


Figura 6.1: Deformação. (a) O usuário seleciona uma região de interesse através de um traço. (b) As três regiões: Fixa (Vermelho), Suporte (Azul) e Handle (Verde) (c) Uma segunda curva induz o perfil da deformação. (d) O modelo deformado.

deve intersectar a superfície do modelo gerando vários anéis de triângulos (Figura 6.6(b)), dos quais escolhemos aquele mais próximo do vetor que gerou o plano de interseção. O mesmo procedimento é realizado para o outro plano, de modo que obtemos dois anéis de triângulos na superfície do modelo (Figura 6.6(c)). Os vértices que pertencem à região do modelo limitada por estes dois anéis de triângulos constituirão o conjunto de pontos livres P . Os vértices do modelo que estão no lado contrário de P , com relação ao segundo loop, constituirão H . Os vértices remanescentes formarão o conjunto F (conforme Figura 6.1 (a) e (b)).

Os vértices em cada uma das três regiões são determinados utilizando um algoritmo de “flood-fill”.

6.2.2 Algoritmo de projeção de curvas

A idéia principal deste algoritmo é projetar de forma coerente uma curva plana na superfície do modelo. Para tanto, determina-se a projeção de um ponto de uma parte da curva, e então computamos os pontos restantes atravessando a superfície. Este procedimento é adequado visto que parte da curva tende a começar ou terminar em uma linha de silhueta. A Figura 6.2 abaixo ilustra este conceito através de um exemplo bidimensional.

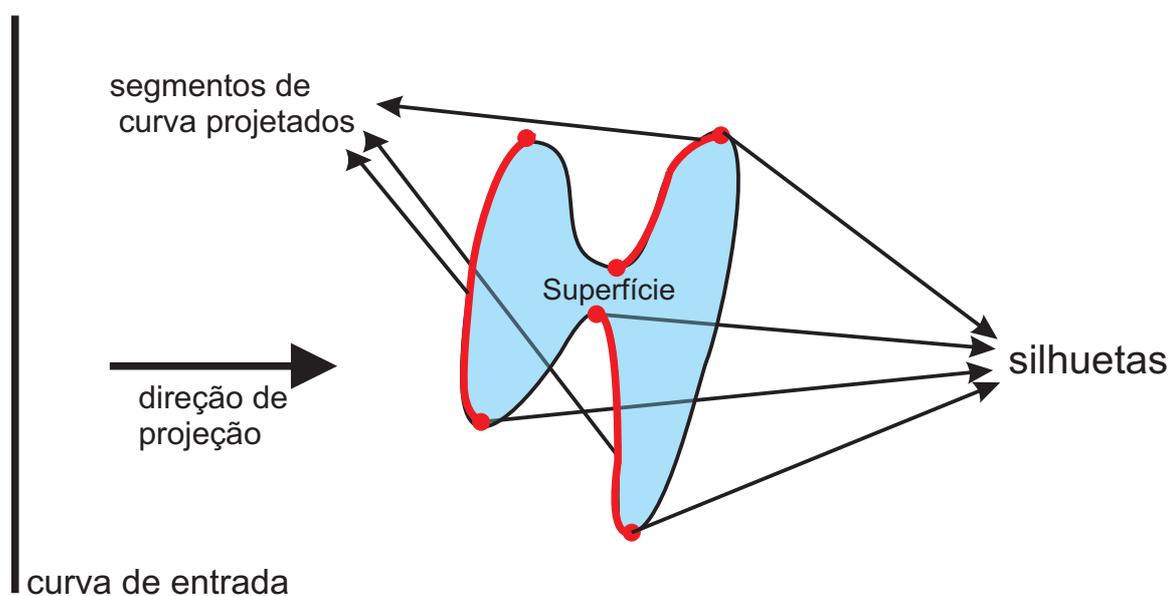


Figura 6.2: Esquema de projeção do traço na superfície

Assumindo que a curva de entrada é representada por uma linha poligonal, a curva projetada será composta por uma ou mais linhas poligonais, uma para cada componente conexa (três segmentos de curva na Figura 6.2, por exemplo).

Um segmento de curva possui dois tipos de vértices. O primeiro tipo consiste de pontos produzidos pela projeção da curva de entrada em vértices na superfície, os quais são obtidos pela interseção entre um raio de projeção e uma face triangular da superfície. O segundo tipo corresponde à interseção entre um *swath* gerado por dois raios paralelos e arestas da superfície (ver Figura 6.3).

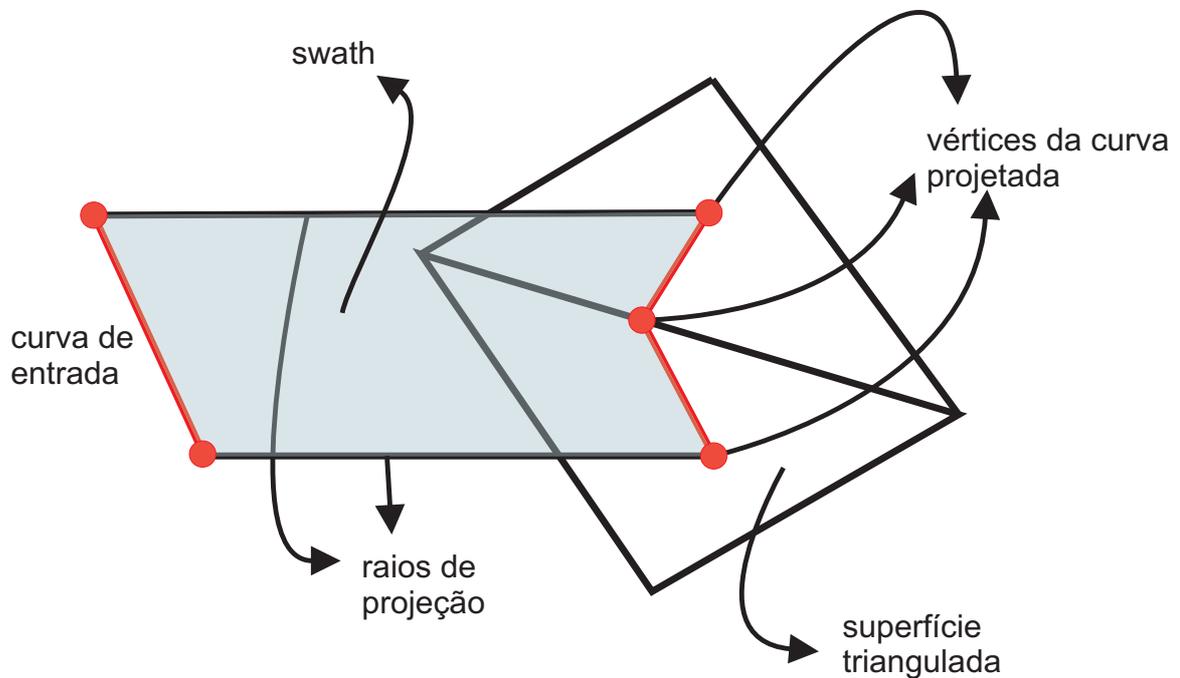


Figura 6.3: interseção entre um *swath* e uma aresta da superfície

O Algoritmo 6.1 abaixo resume os passos para computar os segmentos da curva projetada a partir de uma curva de entrada.

Algoritmo 6.1

1. Compute todas as arestas de silhueta.
 2. Seja r o raio que começa no primeiro vértice da curva de entrada e orientado segundo a direção da projeção.
 3. Para cada ponto de interseção p_i entre r e uma face triangular frontal t_i faça:
 - (a) Crie um segmento de curva projetada c_i e armazene p_i nele.
 - (b) Siga a projeção pela superfície começando por t_i e armazenando os pontos de interseção em c_i .
 4. Defina um *swath* S_j como uma parte do plano entre dois raios r_j e s_j cuja origens são dois pontos consecutivos da curva de entrada e orientados segundo a direção de projeção (ou seja, os dois raios de projeção mostrados na figura 6.3).
 5. Para cada *swath* S_j faça:
 - (a) Para cada vértice de silhueta faça:
 - i. Se S_j intersecta e_k no ponto p_{jk} então:
 - A. Escolha a face triangular frontal t_{jk} entre os dois triângulos que possuem a aresta e_k em comum.
 - B. Crie um segmento de curva projetado c_{jk} e armazene p_{jk} nele.
 - C. Siga a projeção pela superfícies começando por t_{jk} e armazenando os pontos de interseção em c_{jk}
 6. Retorne todos os segmentos de curva c_i .
-

Observe que este algoritmo retorna todos os segmentos de curvas projetadas nas faces frontais e não nas visíveis. A rotina mais importante do Algoritmo 6.1 corresponde aos passos 3(b) e 5(a).i.C. Estas rotinas consistem em seguir a projeção da curva na superfície por continuidade. Para este propósito é dada uma face inicial t na qual está o primeiro ponto do segmento da curva projetada e uma lista v_i de vértices subsequentes da curva de entrada. A rotina segue-curva é detalhada no Algoritmo 6.2.

Algoritmo 6.2

1. Para cada vértice v_i faça:
 - (a) Seja r o raio que começa em v_i e é orientado segundo a direção de projeção.
 - (b) Se r intersecta t no ponto p_i então:
 - i. Armazene p_i na curva de saída.
 - (c) Senão:
 - i. Seja s um raio começando em $v_i + 1$ e orientado segundo a direção de projeção.
 - ii. Seja S o *swath* entre os raios r e s .
 - iii. Enquanto S intersecta alguma das três arestas e_j de t no ponto p_j faça:
 - A. Armazene p_i na curva de saída.
 - B. Se a aresta e_j é aresta de silhueta retorne.
 - C. Tome t como sendo o triângulo do outro lado da aresta e_j .

6.2.3 Seleção das regiões para inserção de características afiadas e suaves

Para realizar operações de edição localizadas, primeiramente utilizamos o algoritmo de projeção visto na Seção 6.2.2 para projetar um traço de entrada feito no plano da tela na superfície. Em seguida utilizamos um algoritmo de corte de malha similar ao descrito por Mitani [75], que simplifica o traço projetado e retriangula – se for o caso – as faces da superfície intersectadas pelo traço, de modo que cada ponto do traço simplificado seja um vértice da superfície retriangularizada (ver Figura 6.4). O Algoritmo 6.3 a seguir descreve este processo, onde os dados de entrada são a malha triangular do modelo e o traço feito pelo usuário.

Algoritmo 6.3

procedure *regionsForSharpSmooth* (*modelo*, *traço*)

1. Projetar *traço* na parte frontal do *modelo* ;
 2. Simplificar o *traço*;
 3. Realocar os vértices do *traço*;
 4. Simplificar novamente o *traço*;
 5. Split (se necessário) das faces intersectadas pelo *traço*;
 6. Determinar P , F e H ;
-

A sub-rotina **Simplificar** no passo 2 retira todos os pontos do traço projetado exceto aqueles que intersectam as arestas da face, reduzindo os casos de interseção entre os triângulos e traço projetado. O passo 3 faz um ajuste nos vértices da malha que estão muito próximos do traço evitando assim o surgimento de triângulos pequenos ou magros. A Figura 6.5 ilustra os casos de splits que podem ocorrer no passo 5. No passo 6 do algoritmo, o conjunto dos vértices do traço projetado simplificado que agora pertencem a superfície são marcados como vértices de controle determinando assim a região H . Os vértice livres P serão os vértices do k -anel de cada ponto de H que não foram marcados como vértices da região de controle (k é um parâmetro definido pelo usuário). Os pontos fixos F são aqueles que não foram marcados. Assim temos as regiões P , F e H determinadas.

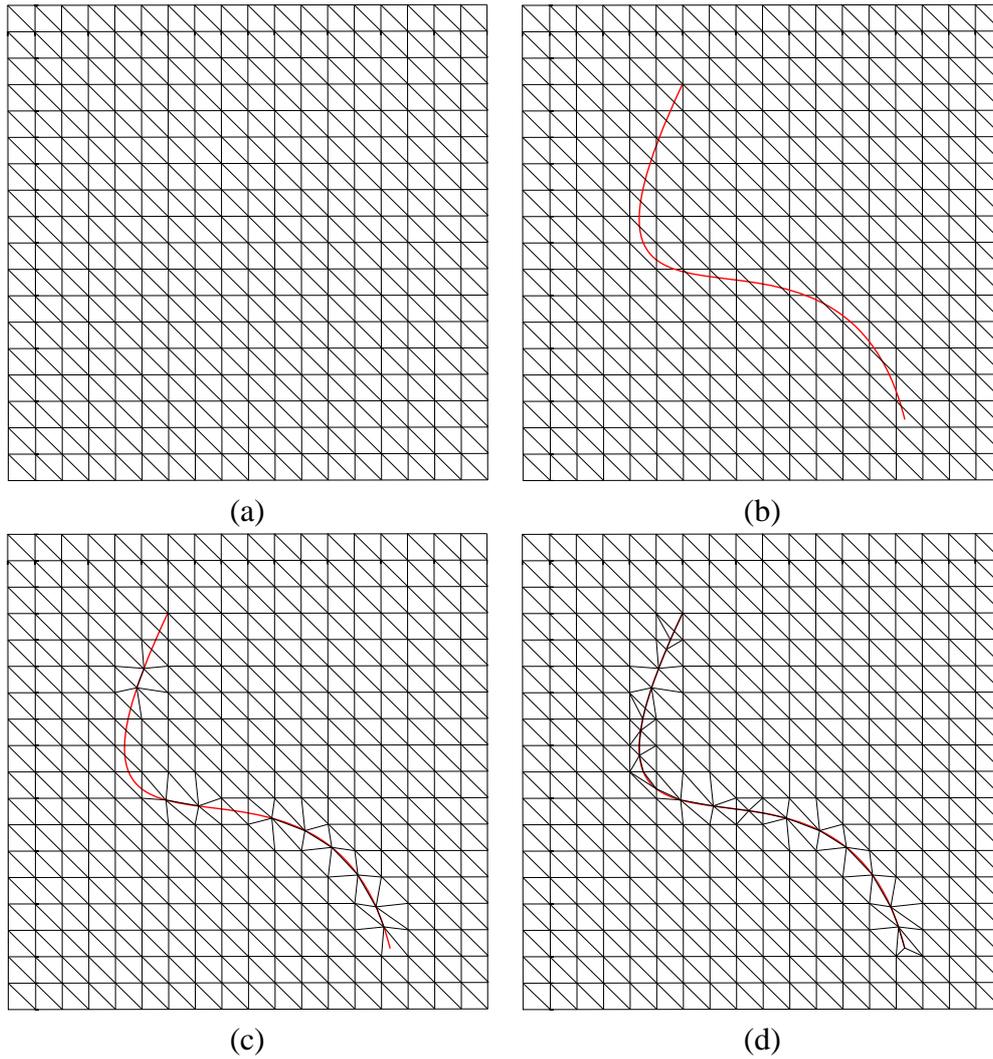


Figura 6.4: Triangulação em função do traço projetado. (a) A malha inicial (b) Um traço é projetado na malha. Uma lista de triângulos intersectados é criada. (c) Os vértices do traço projetado são realocados. (d) A malha é retriangulada.

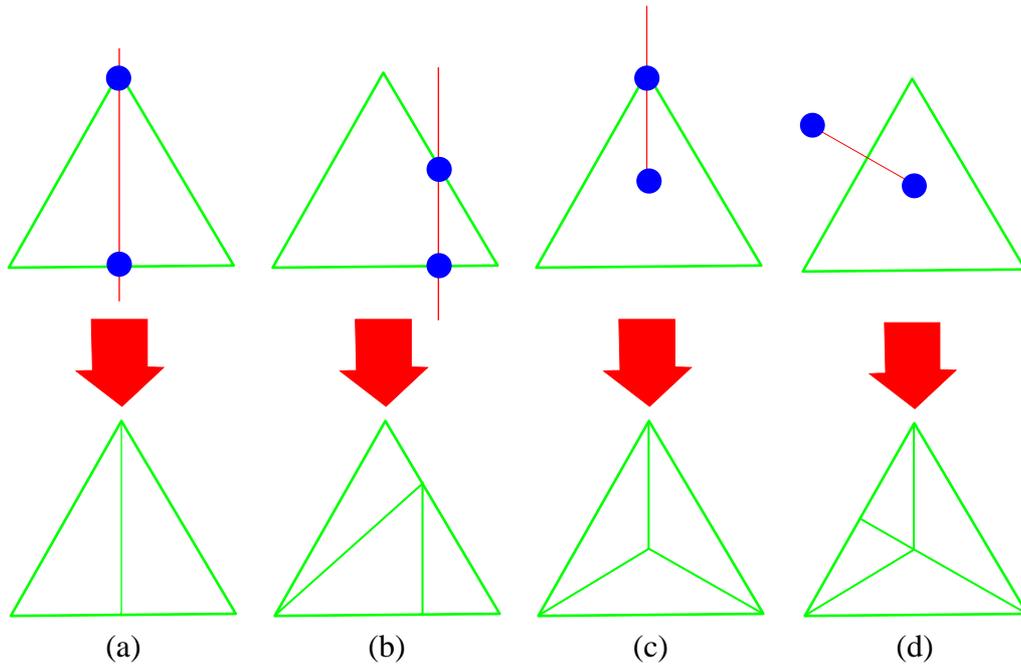


Figura 6.5: Tipos de interseção do traço com os triângulos da superfície e os respectivos splits [75].

6.2.4 Algoritmo de Deformação

A técnica de modelagem à mão-livre baseada no operador laplaciano requer a solução do sistema linear (5.8) para os vértices livres P sempre que o usuário movimenta a região de manipulação H . A taxa de amostragem do objeto pode não ser rápida o suficiente, especialmente quando a suavidade pretendida é elevada e o número de vértices da região de suporte é da ordem de 10^4 ou mais.

Para pré-computar um conjunto especial de funções bases, que correspondem ao grau de liberdade da região de manipulação, o custo computacional pode ser reduzido significativamente a cada amostragem. Se denotarmos por L a matriz da Equação (5.8), a solução para este sistema pode ser expressa explicitamente em termos de sua inversa L^{-1} , de modo que o conjunto de funções bases é representado pelos vetores coluna de L^{-1} . A solução explícita de (5.8) é

$$\begin{pmatrix} P \\ F \\ H \end{pmatrix} =_{L^{-1}} \begin{pmatrix} 0 \\ F \\ H \end{pmatrix} =_{L^{-1}} \begin{pmatrix} 0 \\ F \\ 0 \end{pmatrix} +_{L^{-1}} \begin{pmatrix} 0 \\ 0 \\ H \end{pmatrix}, \quad (6.1)$$

onde o primeiro termo do lado direito da equação permanece constante e o segundo termo depende dos pontos da região de manipulação. James e Pai [57] assim como Desbrun et al. [22] usam funções base pré-computadas utilizando as colunas das matrizes inversas com o objetivo de acelerar a solução de problemas de condições de contorno. Entretanto, para realizar uma deformação complexa com um grande número de vértices de manipulação $m = |H|$, esta pré-computação aumenta, tornando-se equivalente a resolver o problema m vezes, o que é muito caro computacionalmente.

Felizmente, no nosso caso, devido às características de nossas funções de base, podemos restringí-las a interações simples com o usuário, ou seja, transformações de controle simples da região de manipulação. Durante a edição interativa do objeto, usa-se uma interface intuitiva para controlar uma função afim $t : \mathbb{R}^3 \rightarrow \mathbb{R}^3$, que é aplicada aos vértices H da região de manipulação. Pode-se então definir um sistema afim de coordenadas locais gerado por quatro pontos, três a três não-colineares, h_1, h_2, h_3 e h_4 , no interior da região de manipulação. Então existe uma matriz $Q \in \mathbb{R}^{|H| \times 4}$ da combinação afim contendo os pontos $h \in H$ tal que

$$H = Q(h_1, h_2, h_3, h_4)^T.$$

Devido a invariância afim, os pontos transformados $t(H)$ podem ser expressos por meio da função afim

$$\begin{aligned} t(H) &= t(Q(h_1, h_2, h_3, h_4)^T) \\ &= Q(t(h_1), t(h_2), t(h_3), t(h_4))^T. \end{aligned}$$

e conseqüentemente podemos escrever (6.1) como:

$$\begin{pmatrix} P \\ F \\ H \end{pmatrix} =_{L^{-1}} \begin{pmatrix} 0 \\ F \\ H \end{pmatrix} =_{L^{-1}} \begin{pmatrix} 0 \\ F \\ 0 \end{pmatrix} +_{L^{-1}} \begin{pmatrix} 0 \\ 0 \\ Q \end{pmatrix} (h_1, h_2, h_3, h_4)^T$$

Isso faz com o sistema 5.8 seja resolvido em um passo de pré-processamento para sete termos diferentes: as três colunas de $(0, F, 0)^T$ e as quatro colunas de $(0, 0, Q)^T$,

que nos leva a um termo constante pré-computado $C := L^{-1}(0, F, 0)^T$ e à matriz $B := L^{-1}(0, 0, Q)^T$.

Quando o usuário move a região de manipulação, o conjunto $(h_1, h_2, h_3, h_4)^T$ é transformado em $(h'_1, h'_2, h'_3, h'_4)^T$, e a solução do sistema (5.8) pode ser computada por

$$\begin{pmatrix} P' \\ F \\ H \end{pmatrix} = C + B(h'_1, h'_2, h'_3, h'_4)^T \quad (6.2)$$

a qual pode ser realizada em tempo real.

Escrevendo a Equação (6.2) em termos da superfície atualizada e removendo os termos constantes para simplificar a notação temos:

$$P' = P + B \underbrace{(\delta h_1, \delta h_2, \delta h_3, \delta h_4)^T}_{=: \delta C}.$$

Este conceito de matrizes pré-computadas foi proposto por [11, 12]. Em nossa implementação os pontos h_1, \dots, h_4 são tomados de modo a formar um triedro composto de três vetores ortonormais com origem no centro de massa da região de manipulação. Desta forma, considerando a nova posição P' dos vértices livres P como

$$P' = P + B\delta C. \quad (6.3)$$

O primeiro passo é computar a matriz B relacionada com os pontos da região H .

Denotando por T_{init} o triedro da região de manipulação inicial e por T_{end} o triedro da região transformada, o valor de $\delta C = T_{end} - T_{init}$ será a diferença entre a posição atual do triedro e da posição inicial dele. A posição inicial T_{init} é computada pelo sistema no momento de computar a matriz B . A posição T_{end} é dada pelo usuário fazendo um traço ou através da manipulação direta do triedro.

Com esses dados, o sistema computa a nova posição dos vértices P' usando a equação 6.3. A nova posição do conjunto H será dada pela mesma transformação rígida sofrida pelo triedro.

T_{end} e T_{init} são armazenados para possíveis mudanças de suavidade ou de técnicas

de deformação, isto é, após uma transformação por deformação do espaço com uma “suavidade” $k = 2$, por exemplo, o usuário pode executar uma deformação comparativa mudando para deformação por laplaciano, ou se preferir, mudar o valor da “suavidade” k .

Dependendo da escolha do usuário, o sistema usa um dos seguintes algoritmos: *ComputeB_LAPLACIAN* ou *ComputeB_RBF* para computar a matriz B .

Algoritmo 6.4

procedure *ComputeB_LAPLACIAN*(*model*)

1. Posicionar o triedro inicial T_{ini} no centro de massa de H ;
 2. Montar a Matriz L e computar sua inversa L^{-1} ;
 3. Montar a matriz $(00Q)^T$ em função de T_{ini} ;
 4. Montar a matriz $(0F0)^T$;
 5. Computar $B = L^{-1} \times (00Q)^T \times \delta C$;
-

Por outro lado, no caso de deformações baseadas em RBF's, representando-se por Φ^{-1} a inversa da matriz Φ na Equação (5.12), pode-se escrever a solução de (5.11) como

$$W = \Phi^{-1} \begin{pmatrix} F \\ H' \end{pmatrix},$$

Note que analogamente à deformação baseada no operador laplaciano, F permanece constante durante a deformação por RBF's, e que os vértices de manipulação H são somente de forma afim e portanto podem ser representados por uma combinação afim

$$H = M(h_1, h_2, h_3, h_4)^T := MC$$

usando a matriz $M \in \mathbb{R}^{m \times 4}$ das coordenadas afins com respeito às coordenadas locais definidas por quatro pontos de controle $C = (h_1, h_2, h_3, h_4)^T \in \mathbb{R}^{4 \times 3}$. Movendo a região de manipulação de C para uma nova posição $C' = m(C)$, temos que a região é transformada em $H' = MC'$. Explorando este fato, o sistema, acima, para computar os pesos W , simplifica-se em

$$W = \Phi^{-1} \begin{pmatrix} F \\ 0 \end{pmatrix} + \Phi^{-1} \begin{pmatrix} 0 \\ M \end{pmatrix} C'$$

A avaliação da deformação em todos os pontos livres $P = (P_1, \dots, P_N)$ é um operador linear em W e pode ser escrito por $P' = \Phi_p W$ com $(\Phi_p)_{ij} = \phi_j(p_i)$. Portanto, os novos pontos P' podem ser computados por $P' = \underbrace{\Phi_p \Phi^{-1} \begin{pmatrix} F \\ 0 \end{pmatrix}}_{=: B_F} + \underbrace{\Phi_p \Phi^{-1} \begin{pmatrix} 0 \\ M \end{pmatrix}}_{=: B} C'$.

As matrizes $B_F \in \mathbb{R}^{N \times 3}$ e $B \in \mathbb{R}^{N \times 4}$ podem ser pré-computadas e representam as funções base para a deformação. Quando escrita em termos dos vetores deslocamento ($\delta C = C' - C$), esta fórmula reduz-se a

$$P' = P + B\delta C$$

A seguir descrevemos o algoritmo para computar a matriz B para a deformação do espaço.

Algoritmo 6.5

procedure *ComputeB_RBF(model)*

1. Posicionar o triedro inicial no centro de massa de H ;
 2. Montar a matriz Φ usando F e H ;
 3. Computar a inversa de Φ (denotada por Φ^{-1})
 4. Montar a matriz Φ_P
 5. Montar a matriz $(0, M)^T$
 6. Computar $B = \Phi_P \Phi^{-1} (0, M)^T$
-

No caso da deformação baseada em RBF, a suavidade da deformação é determinada pela spline poli-harmônica escolhida. As splines poli-harmônicas em \mathbb{R}^3 são funções da forma

$$f(x) = p_k(x) + \sum_{j=1}^m \lambda_j \|x - x_j\|^{2k-1}, \quad (6.4)$$

onde k é um inteiro positivo e p_k é um polinômio de grau k . Uma justificativa para o nome poli-harmônica spline é que $\|x\|^{2k-1}$ é um múltiplo da solução fundamental Φ da equação de distribuição

$$\Delta^{k+1}\Phi = \delta_0,$$

onde Δ denota o laplaciano e δ_0 é a medida de Dirac na origem. Duchon [31, 32] mostra que as splines poli-harmônicas são da forma

$$\phi_{d,k}(r) = \begin{cases} r^{2k-d}\log(r), & \text{para } d \text{ par} \\ r^{2k-d}, & \text{para } d \text{ ímpar,} \end{cases} \quad (6.5)$$

Como estamos trabalhando em \mathbb{R}^3 temos $d = 3$. Tomando $k = 2$ na Equação 6.5 obtemos a funções base $\phi(r) = r$ dando origem uma spline bi-harmônica e tomando $k = 3$ obtemos uma spline tri-harmônica com as funções base $\phi(r) = r^3$. A suavidade da deformação pode variar continuamente no intervalo $(C^0, C^2]$ através de uma mistura entre as funções bases da RBF de modo a obter uma spline poli-harmônica conveniente.

6.2.4.1 Determinando uma transformação rígida a partir de um traço

Depois que o usuário faz o traço para a seleção das regiões (Figura 6.6(a) a 6.6(d)) o sistema computa o triedro inicial a partir do vetor unitário definido pelo primeiro segmento do traço, juntamente com outros dois vetores, de modo que os três sejam ortonormais, e orientados positivamente. Em seguida o usuário faz um novo traço (Figura 6.6(e)) que induz uma deformação segundo a nova posição da região H definida pelo novo triedro, que é obtido de modo análogo ao triedro inicial, sendo utilizado o segundo traço (Figura 6.6(f)). O usuário pode modificar a posição dos pontos de controle, a suavidade e ainda alternar entre deformações por meio do laplaciano ou deformações por RBF's.

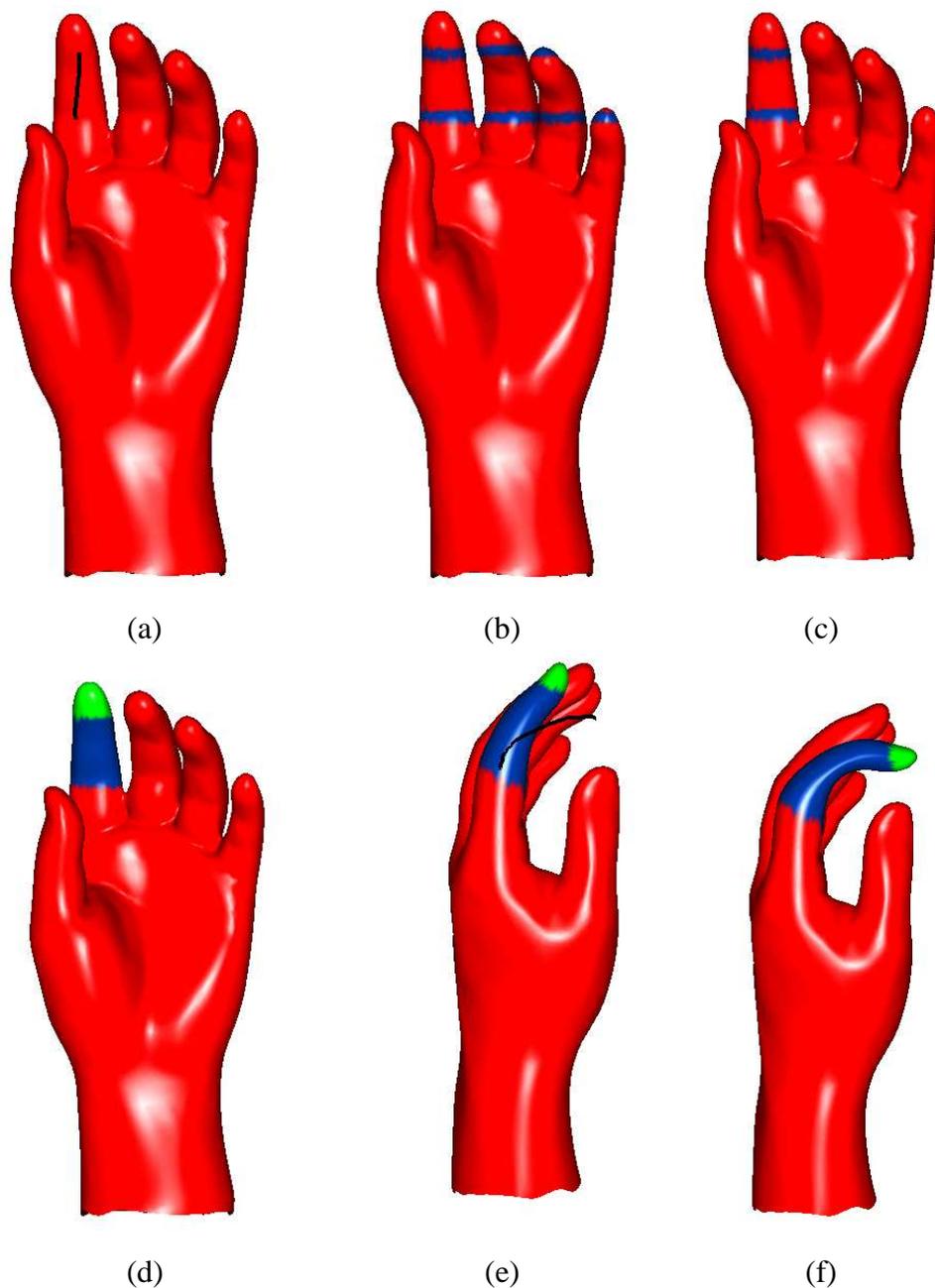


Figura 6.6: Determinação das regiões de deformação por meio de traços. (a) O traço inicial. (b) O sistema detecta vários anéis de triângulos no modelo. (c) Apenas um par de anéis é escolhido. (d) Um algoritmo *flood fill* determina as regiões para a deformação. (e) Um segundo traço define o perfil de deformação. (f) O modelo deformado.

6.2.5 Características suaves e afiadas

Depois de seleccionar as regiões P , F e H conforme a Seção 6.2.3, determinamos uma direção na qual os pontos da região H serão transladados. Esta direção é a média da direção dos vetores normais à superfície em cada ponto da região H . O usuário pode

escolher a deformação a ser executada no modelo, podendo variar linearmente entre RBFs e laplaciano com suavidades que variam entre C^0 e C^2 segundo o método visto em 6.2.4, escolhendo a suavidade que melhor satisfaz as suas necessidades (Ver Figura 5.5 para uma ilustração).

Capítulo 7

Resultados

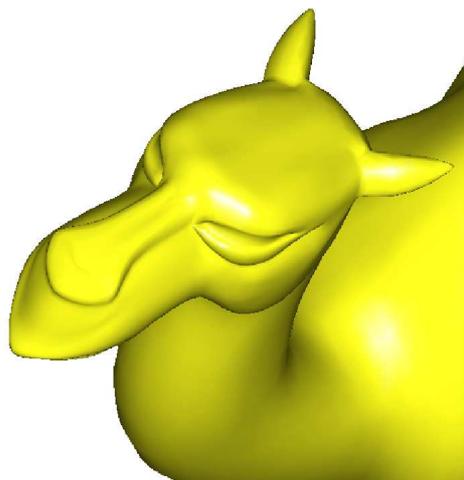
Com o objetivo de avaliar a abordagem proposta e comparar as características diferenciais entre os dois métodos, além de inspeção visual e desempenho, foi construído um protótipo de modelagem e deformação à mão-livre.

Na construção do protótipo utilizamos as bibliotecas CGAL (*Computational Geometry Algorithms Library*) [49] que possui uma vasta coleção de estruturas de dados para tratar geometria, LAPACK (*Linear Algebra Package*) [51] do qual utilizamos as rotinas para resolver os sistemas de equações envolvidos em nossa pesquisa, FLTK (*Fast Light Toolkit*) [50] que foi usado na construção da interface com o usuário e OpenGL (*Open Graphics Library*) [52] para visualização e manipulação dos objetos. Os testes foram realizados em um computador Pentium 4 Core 2 Duo com processador de 2,3 Gz com 2 Gb de memória RAM e sistema operacional Fedora 8.

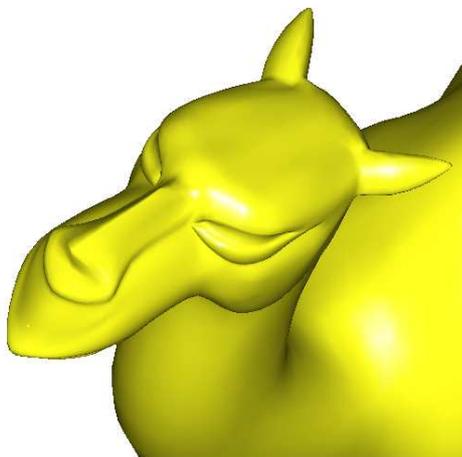
7.1 Poder de expressão

Com o nosso protótipo, é possível realizar deformações bastante expressivas e de grandes proporções, bem como a inserção de características suaves e afiadas na superfície dos modelos.

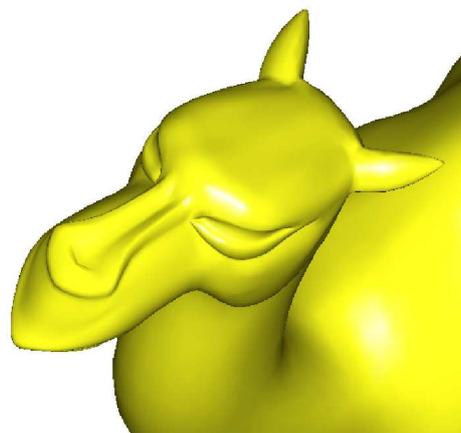
Na Figura 7.1 o camelo sofreu deformação baseada em RBF's para a deformação suave e baseada no laplaciano para característica afiada no nariz.



(a)



(b)



(c)

Figura 7.1: Deformação no nariz do camelo: (a) Modelo original. (b) Características afiadas. (c) Características suaves.

No modelo do manequim a (Figura 7.2) foi modificada a expressão da boca, deformações no nariz e orelha, inserida uma cicatriz na face e uma deformação na testa.

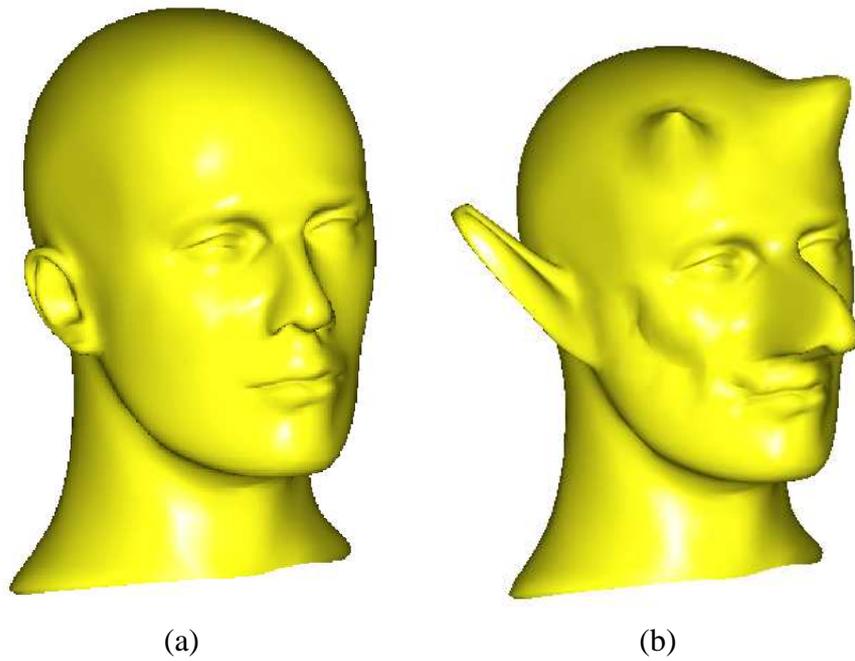


Figura 7.2: (a) Manequim original. (b) Manequim com várias deformações.

O modelo do tigre (Figura 7.3) sofreu várias deformações no posicionamento das pernas, tronco, cabeça, cauda, e foram acrescentadas características afiadas para criar algumas costelas e várias deformações na boca (como mostra a Figura 7.4).

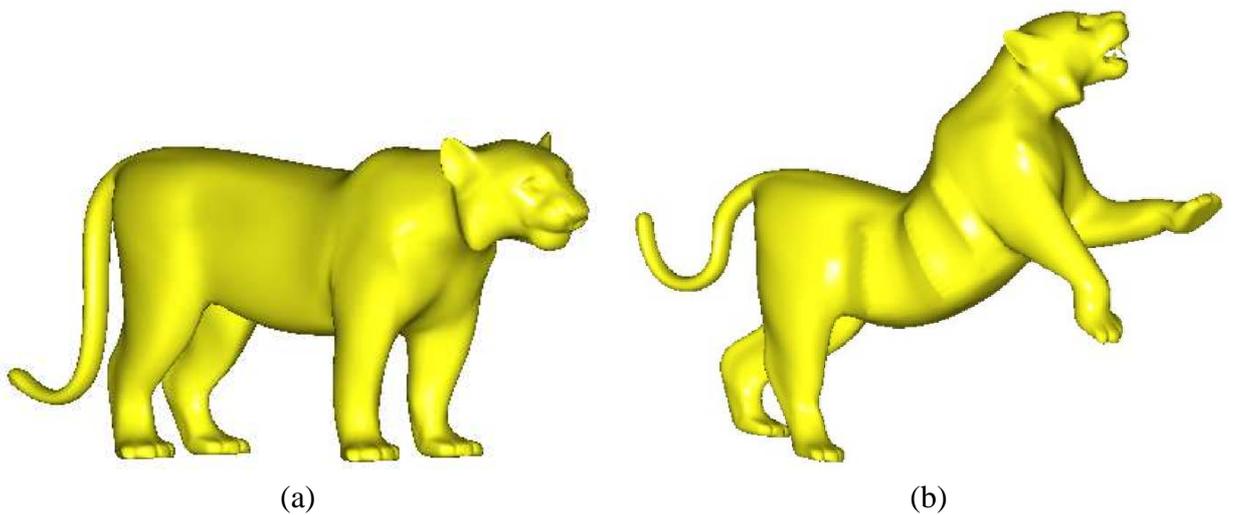


Figura 7.3: Deformação. (a) Modelo do tigre original. (b) Tigre deformado com a ferramenta apresentada.

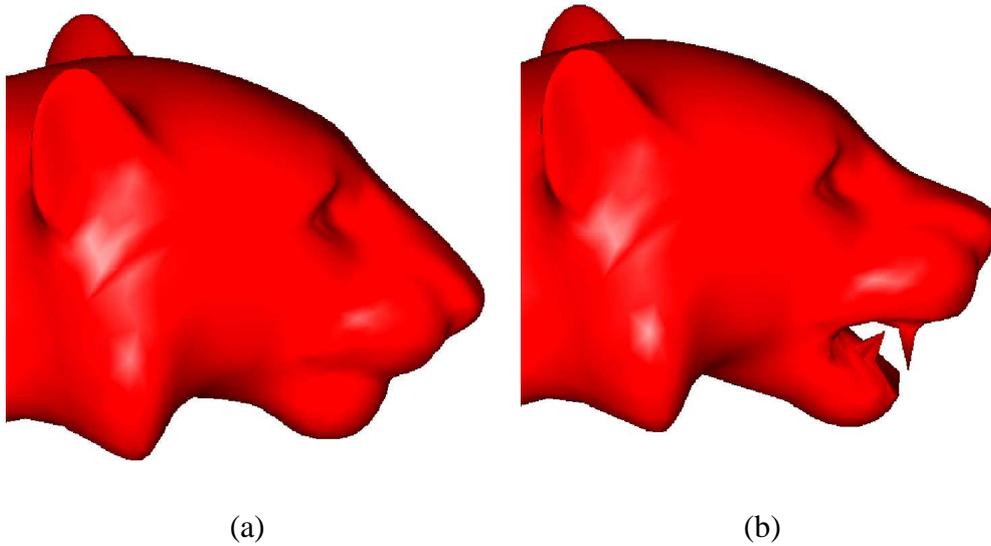


Figura 7.4: (a) Cabeça do tigre original. (b) A boca foi aberta e foram inseridos dentes

7.2 Testes de desempenho

Realizamos vários testes para a comparação entre as técnicas de deformação por laplaciano e por RBFs. Nos testes de desempenho, utilizamos decomposição LU, refinamento iterativo e fatoração de Cholesky para resolver o sistema linear relacionado a cada técnica. As tabelas a seguir relacionam a quantidade de pontos fixos F , livres P e de manipulação H e o tempo de execução das rotinas dos Algoritmos 6.4 e 6.5 para as técnicas de deformação baseadas no laplaciano e em RBFs respectivamente, utilizando decomposição LU.

| S | #P | #F | #H | L | L^{-1} | B | Φ | Φ^{-1} | P_ϕ | B | LAP | RBF |
|---|------|-----|------|------|----------|--------|--------|-------------|----------|--------|--------|--------|
| 0 | 192 | 96 | 96 | 0,04 | 0,04 | 0,08 | 0,01 | 0,05 | 0,01 | 0,11 | 0,08 | 0,11 |
| 0 | 576 | 192 | 384 | 0,12 | 0,65 | 0,78 | 0,12 | 0,54 | 0,1 | 1,21 | 0,8 | 1,22 |
| 0 | 960 | 192 | 768 | 0,2 | 2,67 | 2,87 | 0,32 | 2,55 | 0,26 | 5,08 | 2,9 | 5,09 |
| 0 | 2048 | 384 | 1664 | 0,99 | 30,49 | 31,49 | 1,19 | 23,51 | 1,16 | 44,81 | 31,64 | 44,88 |
| 0 | 3072 | 384 | 2688 | 2,19 | 98,73 | 100,94 | 2,7 | 82,22 | 2,64 | 159,01 | 101,16 | 159,07 |
| 0 | 4096 | 384 | 3200 | 4,37 | 235,28 | 239,76 | 4,76 | 198 | 4,93 | 373,84 | 240,44 | 373,91 |

(a)

| S | #P | #F | #H | L | L^{-1} | B | Φ | Φ^{-1} | P_ϕ | B | LAP | RBF |
|---|------|-----|------|------|----------|--------|--------|-------------|----------|--------|--------|--------|
| 1 | 192 | 96 | 96 | 0,15 | 0,05 | 0,2 | 0,01 | 0,04 | 0,02 | 0,1 | 0,2 | 0,11 |
| 1 | 576 | 192 | 384 | 0,42 | 1,19 | 1,61 | 0,11 | 0,5 | 0,1 | 1,15 | 1,63 | 1,17 |
| 1 | 960 | 192 | 768 | 0,8 | 1,14 | 5,94 | 0,29 | 2,54 | 0,26 | 1,02 | 5,97 | 5,04 |
| 1 | 2048 | 384 | 1664 | 2,27 | 47,91 | 50,19 | 1,18 | 23,58 | 1,17 | 45,1 | 50,33 | 45,17 |
| 1 | 3072 | 384 | 2688 | 6,09 | 164,88 | 170,99 | 2,7 | 82,93 | 2,65 | 160,3 | 171,23 | 160,36 |
| 1 | 4096 | 384 | 3200 | 11,8 | 396,21 | 407,49 | 4,75 | 198,9 | 4,62 | 374,72 | 408,16 | 374,8 |

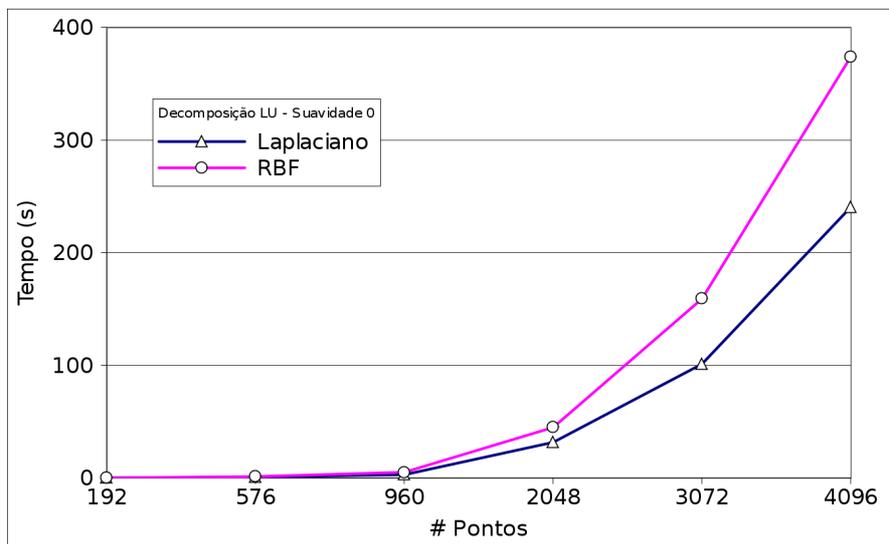
(b)

| S | #P | #F | #H | L | L^{-1} | B | Φ | Φ^{-1} | P_ϕ | B | LAP | RBF |
|---|------|-----|------|-------|----------|--------|--------|-------------|----------|--------|---------|--------|
| 2 | 192 | 96 | 96 | 0,43 | 0,06 | 0,49 | 0,03 | 0,03 | 0,02 | 0,1 | 0,49 | 0,11 |
| 2 | 576 | 192 | 384 | 1,22 | 1,37 | 2,59 | 0,16 | 0,5 | 0,17 | 1,26 | 2,61 | 1,27 |
| 2 | 960 | 192 | 768 | 2,19 | 5,64 | 7,83 | 0,43 | 2,58 | 0,42 | 5,4 | 7,86 | 5,42 |
| 2 | 2048 | 384 | 1664 | 6,63 | 56,26 | 62,8 | 1,9 | 23,75 | 1,91 | 46,76 | 62,94 | 46,82 |
| 2 | 3072 | 384 | 2688 | 6,33 | 163,79 | 170,14 | 4,32 | 83,97 | 4,32 | 164,74 | 170,37 | 164,81 |
| 2 | 4096 | 384 | 3200 | 18,66 | 418,62 | 437,42 | 7,45 | 198,7 | 7,45 | 379,85 | 438,223 | 379,91 |

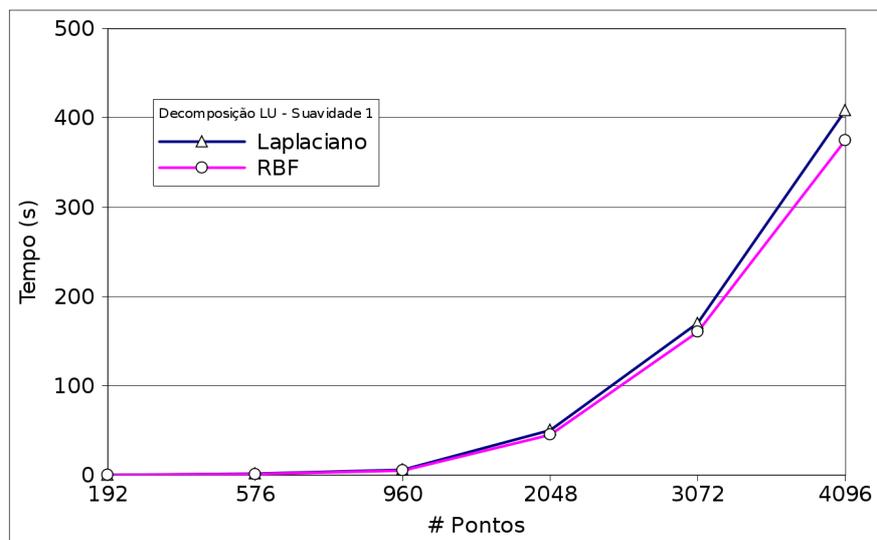
(c)

Tabela 7.1: Tempo de execução em segundos para montar a matriz L , computar sua inversa L^{-1} , computar a matriz B , montar a matriz Φ , computar sua inversa Φ^{-1} , computar a matriz P_ϕ , computar a matriz B , o tempo total da deformação por meio do operador laplaciano e o tempo total da deformação por meio de RBFs de acordo com o número de pontos usando decomposição LU para suavidades C^0 (a) C^1 (b) e C^2 (c).

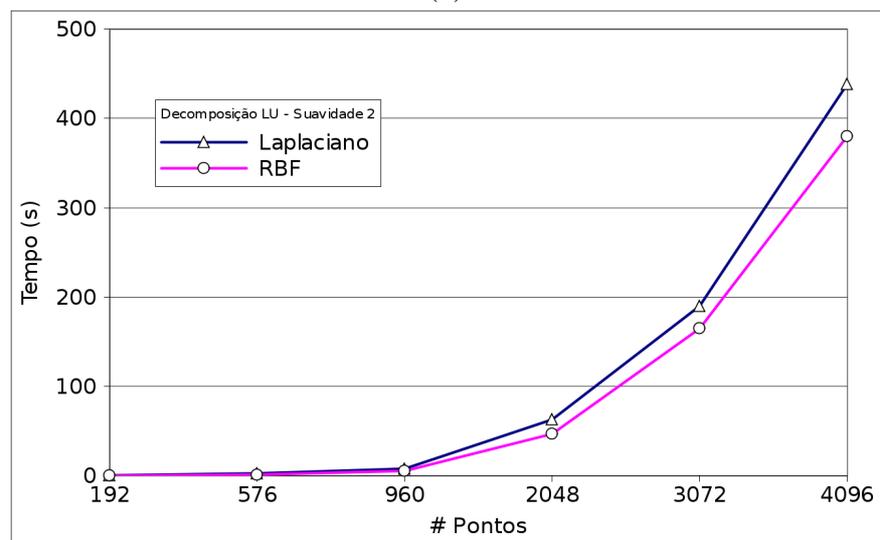
A seguir são mostrados os gráficos de desempenho das deformação por meio do operador laplaciano e por RBFs com base nas informações da Tabela 7.1



(a)



(b)

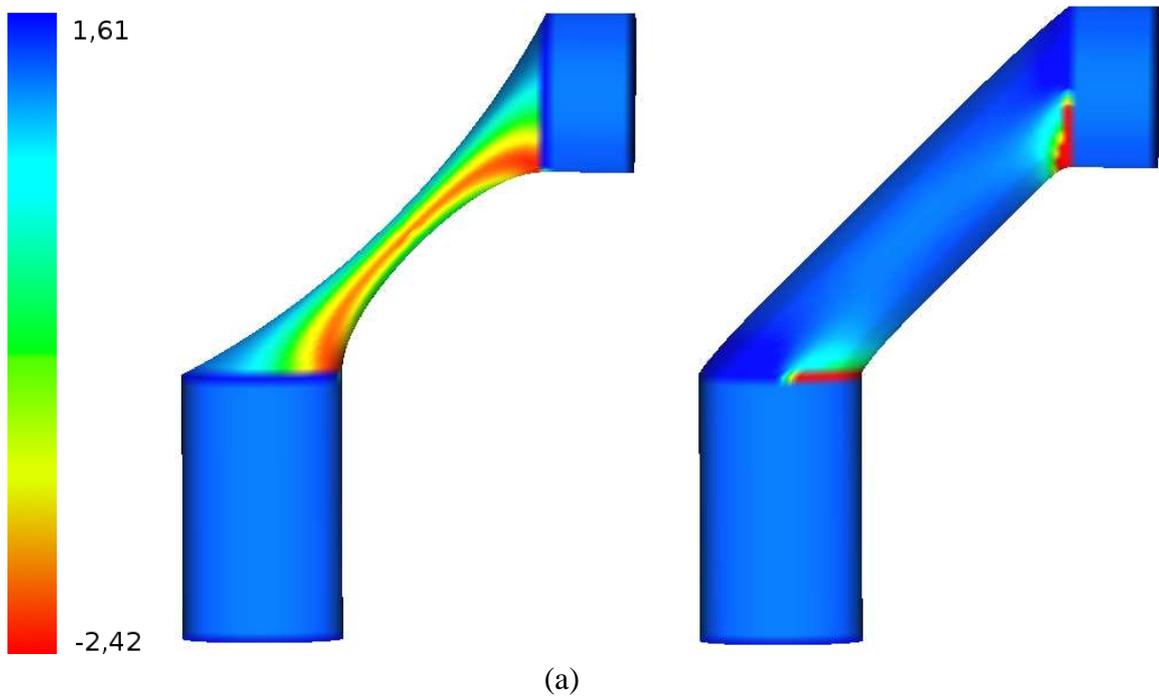


(c)

Figura 7.5: Gráficos baseados nas informações das Tabelas 7.1 (a), (b) e (c) destacando o desempenho das técnicas de deformação por laplaciano e por RBFs usando decomposição LU para as suavidades C^0 (a), C^1 (b) e C^2 (c).

7.3 Propriedades diferenciais

Fizemos uma análise de algumas propriedades diferenciais referentes às duas técnicas de deformação discutidas neste trabalho. Calculamos a curvatura gaussiana e a curvatura média na deformação de um cilindro, considerando as suavidades C^0 , C^1 e C^2 com a finalidade de verificar qual técnica fornece as características diferenciais desejáveis ao usuário. A seguir mostramos o resultado da deformação com suavidades C^0 , C^1 e C^2 destacando a curvatura gaussiana em cada deformação, representando com a cor vermelha os pontos de menor curvatura e de azul os pontos com maior curvatura gaussiana.



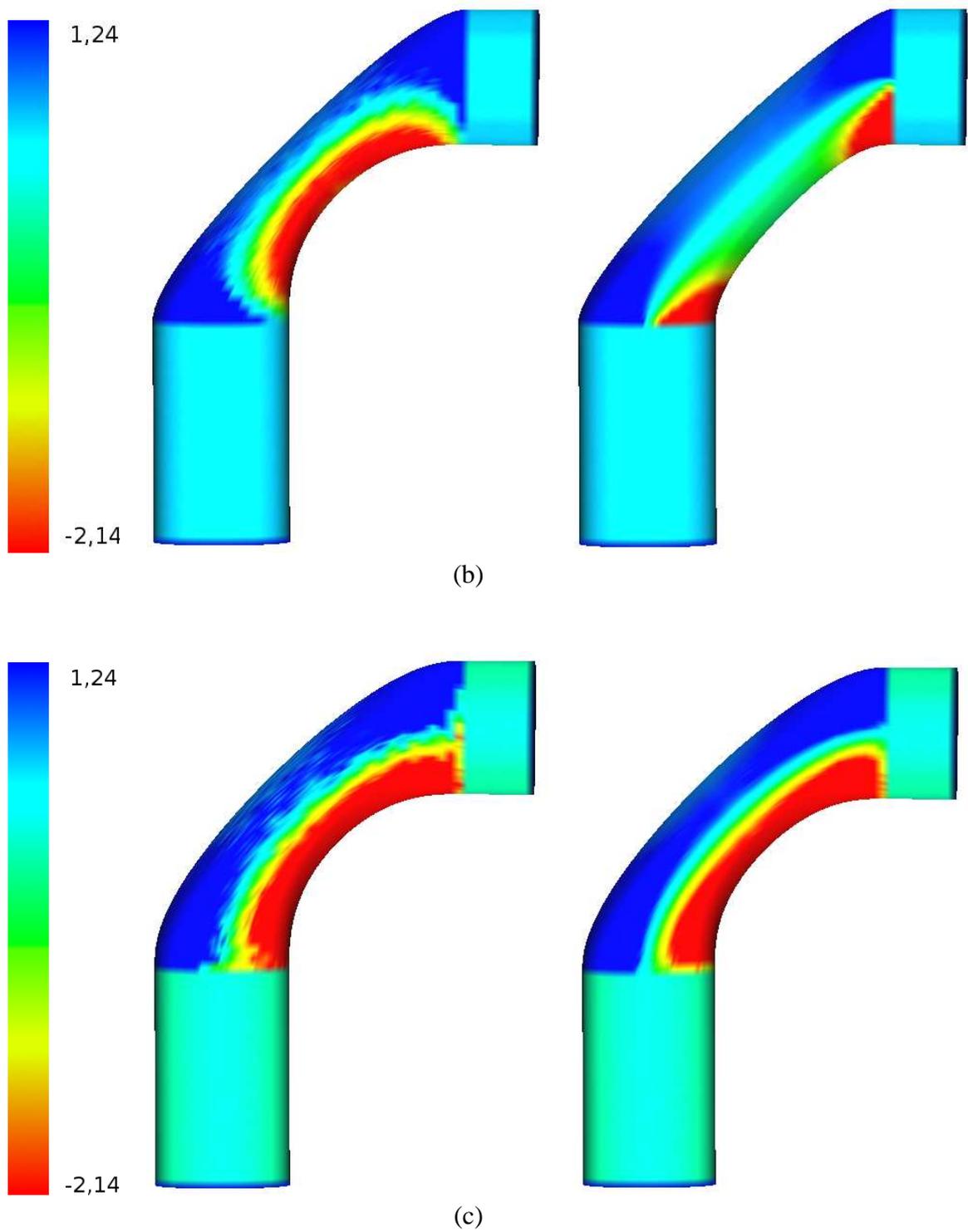
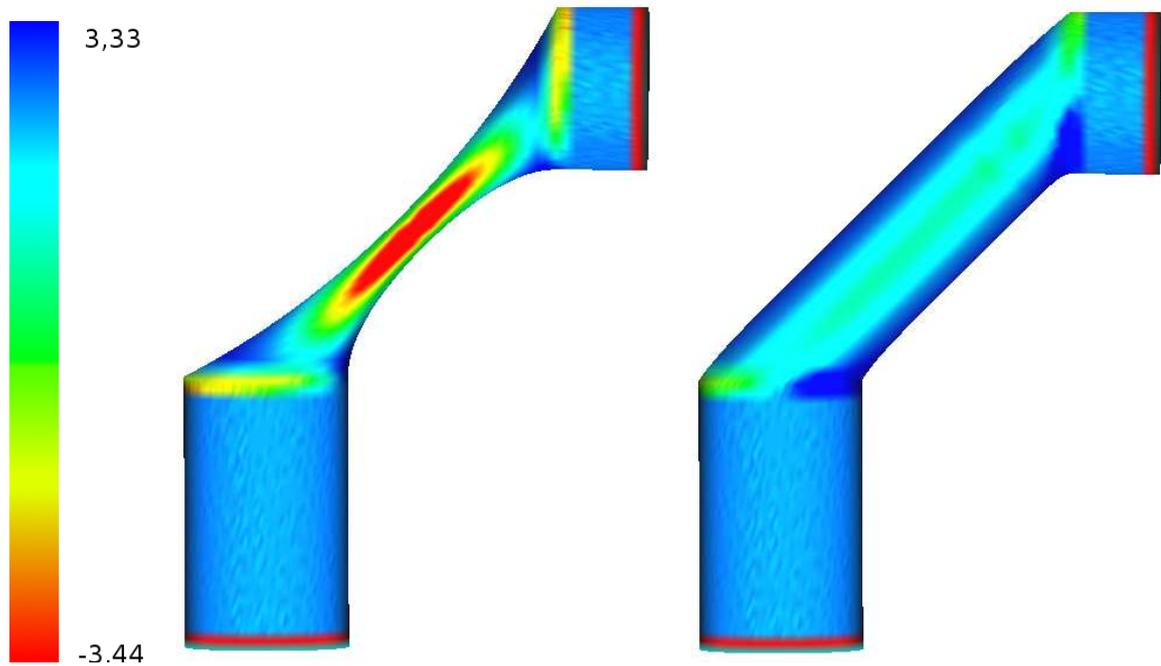


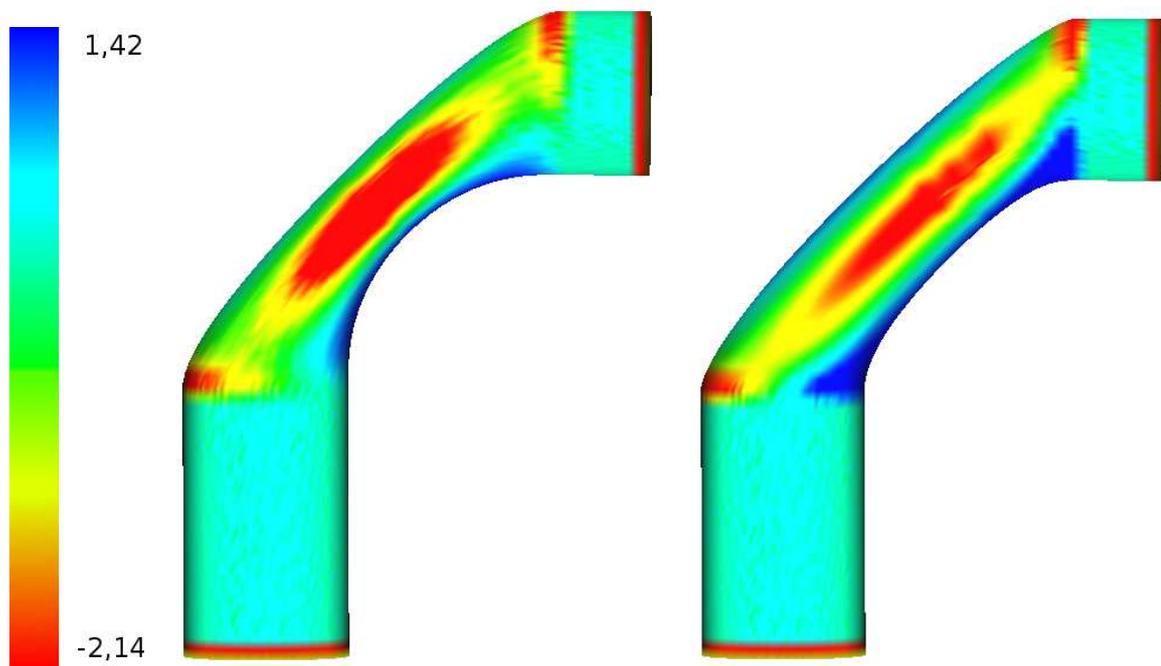
Figura 7.6: Na esquerda deformação por laplaciano e na direita deformação por RBF destacando a curvatura gaussiana com suavidades C^0 (a), C^1 (b) e C^2 (c).

A seguir mostramos o resultado da deformação com suavidades C^0 , C^1 e C^2 destacando a curvatura média em cada deformação, representando com a cor vermelha os

pontos de menor curvatura e de azul os pontos com maior curvatura média.



(a)



(b)

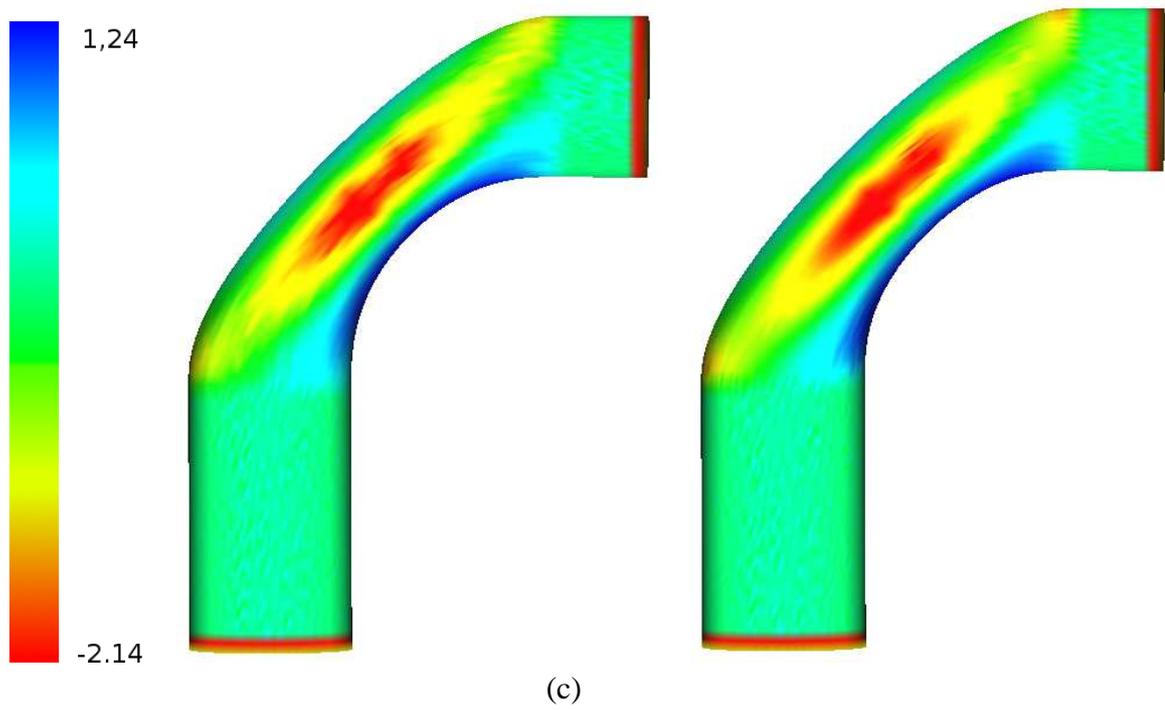


Figura 7.7: Na esquerda deformação por laplaciano e na direita deformação por RBF destacando a curvatura média com suavidades C^0 (a), C^1 (b) e C^2 (c).

Capítulo 8

Conclusão

O trabalho propõe uma ferramenta à mão-livre para a modelagem e deformação de superfícies através de traços. O protótipo permite transformações suaves e afiadas na geometria da superfície do modelo. A edição de objetos é realizada com base no paradigma intuitivo *drag and drop*. A região de manipulação pode ser movida livremente por meio de sucessivos traços ou pela manipulação direta usando o paradigma de *drag and drop*, enquanto a região de suporte se ajusta sem perder muito a sua geometria local. A deformação da região suporte é baseada na minimização de um funcional de energia com apelo físico e é implementado de duas formas diferentes: Geometria Diferencial Discreta (discretização do operador Laplaciano) e Métodos de Elementos Finitos (por Funções de Bases Radiais). Em ambas implementações de deformação, o usuário pode mudar continuamente a suavidade da deformação entre C^0 e C^2 e comparar visualmente os resultados obtidos pelos algoritmos de minimização (Laplaciano e RBF's).

A criação e manipulação de modelos tridimensionais com o nosso sistema é fácil, rápida e intuitiva. Nós seguimos principalmente a técnica baseada em traços para realizar seleções no modelo para a especificação das regiões de manipulação, suporte e fixa. O sistema permite ao usuário criar uma aproximação inicial da geometria de um objeto por meio do desenho no plano da tela uma curva que dá uma idéia da silhueta do objeto, e intuitivamente, realizando transformações geométricas, obter uma geometria final do objeto.

Para inserir características afiadas à superfície de um modelo, a deformação baseada no operador laplaciano produz melhores resultados, visto que a deformação por RBFs tende a gerar deformações suaves devido às características intrínseca deste tipo de deformação.

Para as deformações de continuidade C^0 a deformação dos pontos livres baseada no laplaciano caracteriza-se como uma superfície membrana a qual tenta minimizar a área da superfície (Figura 7.6 (a)). Para continuidade C^2 a superfície deformada por laplaciano dos pontos livres caracteriza-se como uma superfície *thin plate* a qual tenta minimizar o *bending* da superfície (Figura 7.6 (b)) e com continuidade C^3 obtemos uma superfície que tenta minimizar a variação da curvatura mínima linearizada (Figura 7.6 (c)). Continuidades maiores que C^2 não são recomendadas devido a instabilidades numéricas.

As propriedades diferenciais nos resultados obtidos para a suavidade C^3 usando as duas técnicas são muito próximos, sendo que a técnica de deformação por RBFs produz uma malha final com melhor qualidade, e menor tempo de execução para realizar a deformação da região do pontos livres. Este melhor desempenho se deve ao fato de que na deformação por RBFs a matriz para resolver o sistema linear está relacionada apenas com os pontos de manipulação H e os pontos fixos F , sendo tal matriz menor (na grande maioria dos casos) que a usada para deformação por laplaciano que usa pontos de F , H e P .

Na resolução dos sistemas de equações lineares envolvidos nas deformações, testamos o desempenho da decomposição LU, do método iterativo e fatoração de Cholesky, e verificamos que o método que utiliza fatoração LU teve o melhor desempenho. A decomposição LU é uma das técnicas mais usadas para resolver sistemas de equações algébricas, sendo usada para decompor a matriz dos coeficientes A , em duas matrizes L e U , onde U é uma matriz triangular superior (todos os elementos abaixo da diagonal principal são nulos), e L é uma matriz triangular inferior. A decomposição LU tem complexidade as-

sintótica $O(n^3)$, sendo compatível com os resultados obtidos em nossos testes conforme mostra a Tabela 7.1.

As deformações por laplaciano possuem melhor desempenho para suavidades entre C^0 e C^1 enquanto as baseadas em RBFs possuem melhor desempenho entre as suavidades C^1 e C^2 conforme mostram os gráficos (a) e (c) da Figura 7.5.

Como trabalho futuro, pretendemos implementar a avaliação das funções RBFs e a solução dos sistemas de equação por meio de programação em GPU (*Graphics Processing Unit*) com o objetivo de melhorar o desempenho da avaliação das RBFs e da solução dos sistema envolvidos. Também pretendemos estudar outros tipos de RBFs na implementação da deformação baseada em RBFs para obter resultados melhores para as continuidades entre C^0 e C^1 .

Nosso protótipo e alguns modelos podem ser encontrados em www.lcg.ufrj.br/Members/disney/.

Apêndice A

Noções de Geometria Diferencial

Neste capítulo faremos uma breve abordagem de alguns conceitos úteis da Geometria Diferencial aplicados a curvas e superfícies, apresentando as curvaturas média e Gaussiana, mostrando que ambas são generalizações da noção de curvatura para curvas planas. A descrição a seguir, foi baseada em [28] e trata o assunto de modo bastante sucinto. Uma abordagem mais aprofundada com todas as demonstrações pode ser encontrada nos Capítulos I, II e III de [27] e uma excelente exposição intuitiva se encontra no Capítulo IV de [47].

A.1 Curvas

Uma *curva regular* é um conjunto uni-dimensional C que é suave (possui derivadas de todas as ordens) em todos os seus pontos (Figura A.1 (a)). A Suavidade garante que a reta que passa por p_1 e $p_2 \in C$ se aproxime de uma posição limite quando p_1 e p_2 tendem a um ponto $p \in C$. Essa posição limite é chamada de (reta) *tangente* a C no ponto p (Figura A.1 (b)). Portanto uma curva regular C possui uma tangente em cada ponto, e tal tangente varia continuamente com o ponto. Isto evita “bicos” como o da Figura A.1 (c). Por simplicidade, omitiremos frequentemente o adjetivo regular.

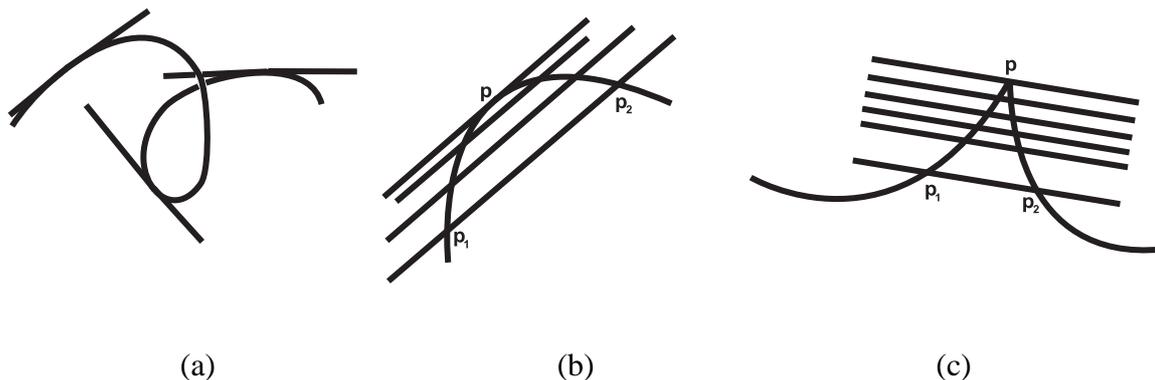


Figura A.1: Em (c), o limite da reta p_1, p_2 depende de como p_1 e p_2 se aproximam de p .

As curvas mais simples são a reta e o círculo. A noção de tangente nasce da comparação de uma curva com uma reta (a tangente a C em p é a reta que melhor aproxima a curva nas proximidades de p). Mostraremos que a comparação de uma curva plana com um círculo dá origem a idéia de curvatura.

A.2 Curvaturas de curvas planas

Seja C uma curva em um plano P . Escolhamos um sentido de percurso para C e um sentido de rotação em P ; diremos então que C e P estão *orientados*. Associemos a cada ponto $p \in C$ um vetor tangente unitário $e_1(p)$ na orientação de C (Figura A.2 (a)).

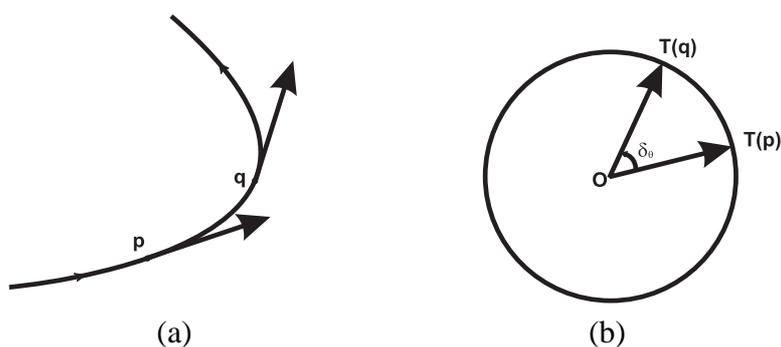


Figura A.2: A aplicação tangente.

Seja S^1 um círculo de raio um e centro O em P . A *aplicação tangente* $T : C \rightarrow S^1$ é definida associando a cada $p \in C$ a extremidade de $e_1(p)$, quando sua origem é transladada para O (Figura A.2 (b)).

Sejam p e q dois pontos próximos de C e seja δ_S o comprimento de arco em C de p a q , onde δ_S é positivo se p se desloca para q na orientação da curva. Indiquemos por δ_θ o ângulo de $T(p)$ a $T(q)$, na orientação do plano P (Figura A.2 (b)). Então como consequência da suavidade de C , existe o limite

$$\lim_{q \rightarrow p} \frac{\delta_\theta}{\delta_S} = n(p)k(p),$$

e o seu valor é chamado a *curvatura* de C em p .

Intuitivamente, a curvatura em p mede a velocidade de variação da reta tangente nas proximidades de p , o que justifica o uso da palavra curvatura (Figura A.3).

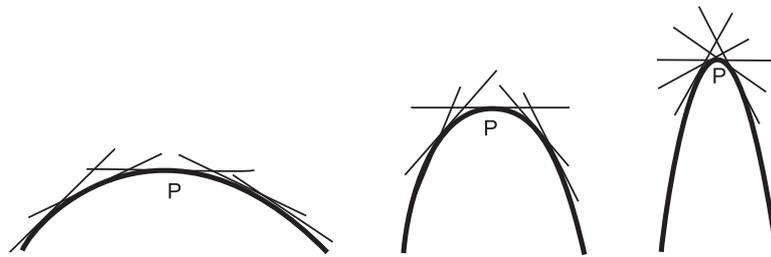


Figura A.3: A curvatura mede a velocidade de variação da tangente

A curvatura de uma reta é nula em todos os pontos (visto que a aplicação tangente é constante) e a curvatura de um círculo de raio R é igual a $1/R$ (pois, neste caso temos $\delta_S = \delta_\theta$).

A curvatura tem um sinal que muda quando um dos dois, a orientação da curva ou do plano muda.

Uma maneira “visual” de determinar o sinal da curvatura é associar a cada ponto $p \in C$, um vetor unitário $e_2(p)$ na direção da normal a C em p (a *normal* é a reta perpendicular à tangente a C em P), e orientado de tal modo que o ângulo de $e_1(p)$ e $e_2(p)$ seja $\frac{\pi}{2}$ na orientação do plano P . Nos pontos onde $k(p) \neq 0$, a curva está, nas proximidades de p , de um mesmo lado da reta tangente a p ; o semi-plano assim determinado é chamado o *lado da concavidade* de C em p , e não depende de qualquer orientação (Figura A.4).

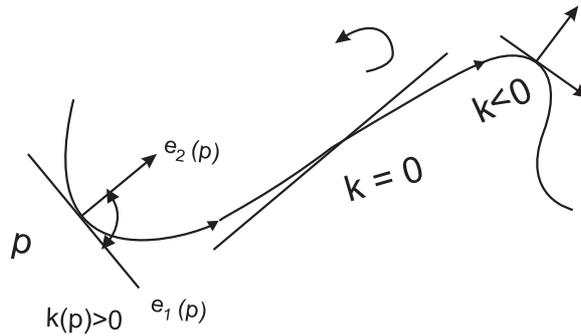


Figura A.4: O sinal da curvatura

A curvatura $k(p)$ é positiva se aponta para o lado da concavidade da curva, e negativa no caso contrário.

A.3 Superfícies

Uma *superfície (regular)* é um conjunto bi-dimensional que é suave em todos os seus pontos. Em particular, dados três pontos p_1, p_2 e $p_3 \in S$, o plano determinado por esses pontos tem uma posição limite quando p_1, p_2 e p_3 tendem para um ponto $p \in S$, de uma maneira qualquer. Este plano é chamado o *plano tangente* $T_p S$ a S em p . Pela própria construção, $T_p S$ contém as tangentes às curvas em S que passam por p . Como no caso de curvas, omitiremos com frequência o adjetivo regular.

Dizemos que o espaço tridimensional E^3 está *orientado* quando escolhermos três vetores unitários e ortogonais e_1, e_2, e_3 , nesta ordem. Qualquer outro terno de vetores unitários e ortogonais f_1, f_2, f_3 está *na orientação* de e_1, e_2, e_3 quando, se deslocarmos (rigidamente) f_1, f_2, f_3 de modo que f_1 coincida com e_1 e f_2 coincida com e_2 , então f_3 coincide com e_3 (Figura A.5).

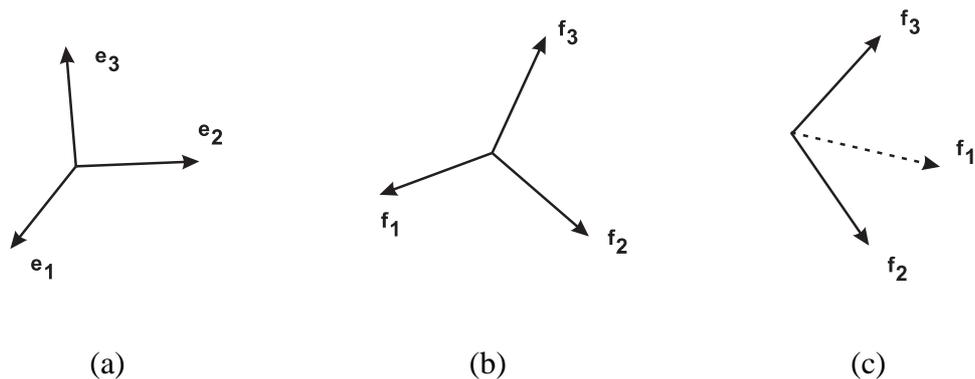


Figura A.5: O terno f_1, f_2, f_3 está na orientação de e_1, e_2, e_3 em (b), e não está na orientação de e_1, e_2, e_3 em (c)

Dada uma superfície S , é possível escolher em cada ponto $p \in S$ dois vetores unitários normais ao plano tangente $T_p S$. Se for possível escolher um destes vetores de maneira contínua em toda a superfície S , diz-se que S é orientável. Um exemplo de superfície não orientável é a faixa de Möbius (Figura A.6).

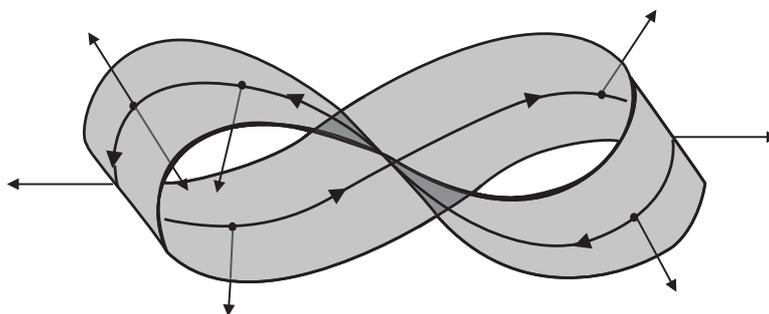


Figura A.6: A faixa de Möbius é não orientável: Percorrendo uma curva fechada na superfície, o vetor unitário normal volta na posição oposta, o que mostra que não é possível defini-lo continuamente

Uma superfície que é limitada (isto é, está contida no interior de uma bola de raio suficientemente grande) e não tem fronteira é chamada uma *superfície fechada*. Esferas e elipsóides são exemplos de superfícies fechadas. Cilindros e parabolóides infinitos não são superfícies fechadas por não serem limitadas. Por outro lado, uma parte do cilindro e uma parte do parabolóide não são fechadas por possuírem pontos de fronteira. (Figura A.7 (e) e (f))

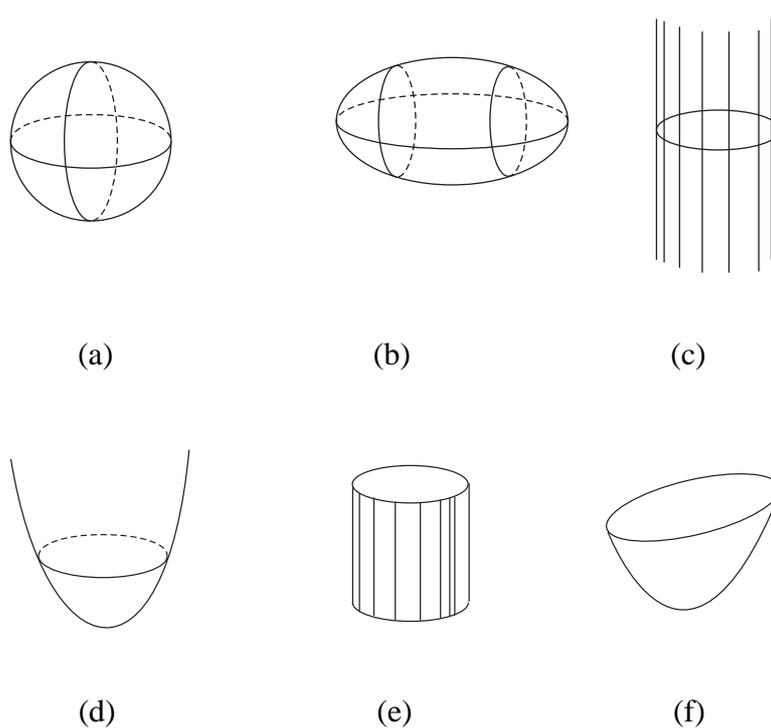


Figura A.7: (a) esfera. (b) elipsóide. (c) cilindro. (d) parabolóide. (e) parte de um cilindro. (f) parte de um parabolóide.

A faixa de Möbius da Figura A.6, que é não orientável, não é fechada. Isto não acontece por acaso, e pode ser provado que toda superfície em \mathbb{R}^3 fechada é orientável.

Uma superfície orientável S está orientada quando escolhemos em cada ponto $p \in S$ um vetor unitário normal $N(p)$ a S que varia continuamente com p . N é chamada um *campo normal unitário*. Intuitivamente, isto corresponde a dar continuamente um sentido de rotação a cada plano tangente $T_p S$ que, junto com N , coincide com uma orientação dada de E^3 (Figura A.8).

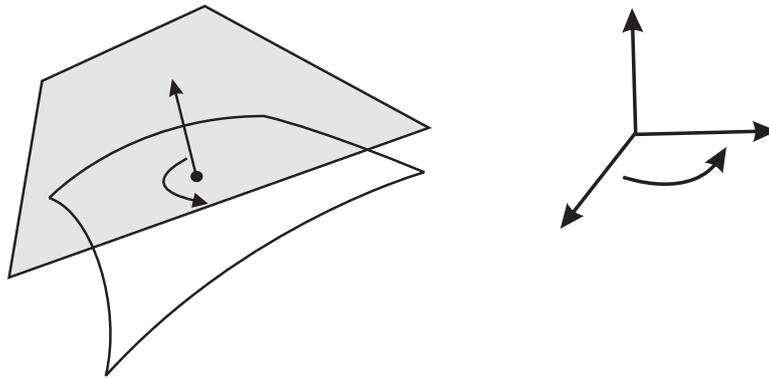


Figura A.8: A orientação do plano tangente, juntamente com N , coincide com a orientação dada de E^3 .

Pretendemos agora estender a noção de curvatura de curvas planas para superfícies. Seja $p \in S$, onde $S \subset \mathbb{R}^3$ é uma superfície orientada de um espaço orientado E^3 . Seja $N(p)$ o vetor unitário normal a S em p , na orientação dada de S , e seja v um vetor unitário do plano tangente $T_p S$. O plano P_v , que contém v e N , corta a superfície S segundo uma curva plana C_v , chamada *secção normal* de S em p segundo v . A *curvatura normal* k_v de S em p segundo v é definida como o valor absoluto da curvatura de C_v com sinal positivo, se N está voltada para a concavidade de C_v , e negativo caso contrário. Note que $k_v = k_{-v}$ (Figura A.9).

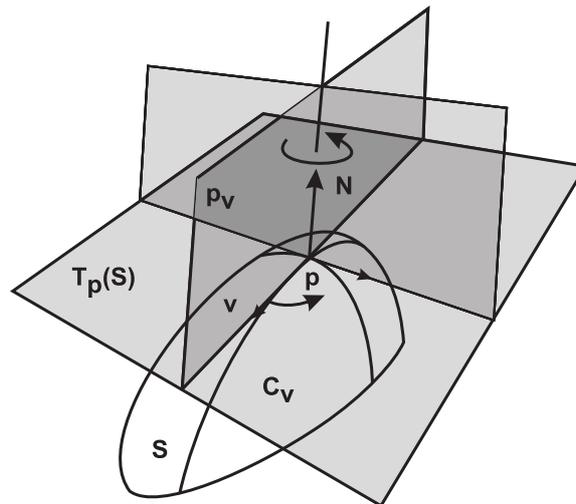


Figura A.9: As curvaturas de Euler

Quando P_v gira em torno de N , v descreve um círculo S^1 de raio um em $T_p S$. É

possível mostrar que k_v depende continuamente de $v \in S$ e, portanto, atinge um máximo k_1 e um mínimo k_2 em S^1 . As curvaturas k_1 e k_2 são chamadas *curvaturas principais* de S em p , e as direções correspondentes (ou seja, os vetores unitários correspondentes) são chamadas *direções principais* em p . Euler provou em [34], que conhecidas as curvaturas principais k_1 e k_2 e o ângulo que faz v com, digamos a direção correspondente a k_1 , k_v é dada por

$$k_v = k_1 \cos^2 \alpha + k_2 \sin^2 \alpha.$$

Portanto, k_1 e k_2 determinam todas as curvaturas normais em p .

Chamamos *curvatura Gaussiana* de S em p o produto $K = k_1 k_2$ das curvaturas principais e *curvatura média* a média aritmética $H = \frac{k_1 + k_2}{2}$ das curvaturas principais. Note que se mudamos a orientação N de S , as curvaturas k_1 e k_2 mudam de sinal; neste caso, a curvatura Gaussiana não se altera mas a curvatura média muda de sinal. Consideremos agora $p \in S$ um ponto de uma superfície S , $H(p)$ a curvatura média em p . Denotaremos por Δ_S o operador que mapeia a cada ponto p de S o operador $\Delta_S(p) = 2H(p)$. Δ_S é conhecido como operador *curvatura média normal*, ou operador de Laplace-Beltrami da superfície S , que vem a ser uma generalização do operador de Laplace.

Na verdade a curvatura Gaussiana foi introduzida por Gauss em 1827 [42], não da maneira acima mas imitando, para superfícies, a definição de curvatura de curvas planas. Gauss provou em [42] que a curvatura por ele definida era igual ao produto das curvaturas principais de Euler.

A.4 Curvatura Gaussiana

Descreveremos agora como Gauss definiu a curvatura de uma superfície.

Seja S uma superfície orientada e $S^2(1)$ uma esfera de raio um e centro O . A *aplicação normal* (ou *aplicação esférica* ou *aplicação de Gauss*) $N : S \rightarrow S^2(1)$ é definida

fazendo corresponder a cada $p \in S$ a extremidade do vetor unitário normal $N(p)$ da orientação de S , quando a origem de $N(p)$ é transladada para o centro O de $S^2(1)$ (Figura A.10).

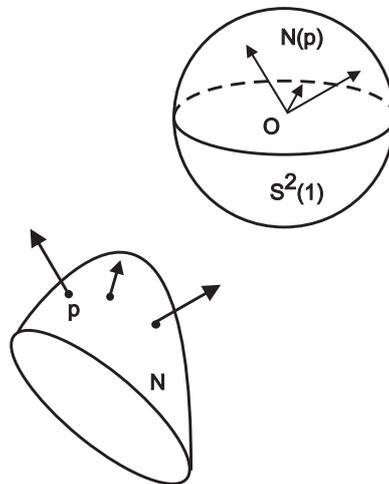


Figura A.10: A aplicação normal de Gauss

Se a aplicação normal de Gauss deixa de ser biunívoca nas proximidades de um ponto $p \in S$, o seu comportamento pode ser bastante complicado: Ela pode transformar um domínio regular em um domínio limitado por uma curva em forma de um 8 (Figura A.11); ela pode recobrir parte já coberta do domínio (Figura A.12); e ela pode transformar a fronteira do domínio em uma curva que dá várias voltas antes de fechar (Figura A.13).



Figura A.11: O domínio regular é transformado em uma curva em forma de 8

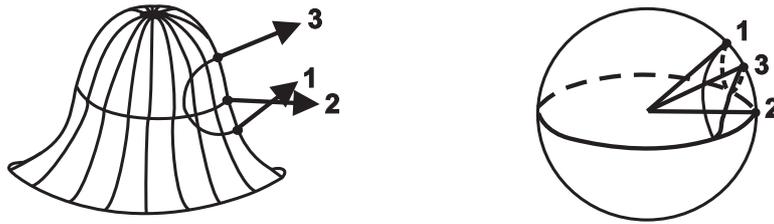


Figura A.12: A Aplicação Normal de Gauss recobre parte já coberta do domínio

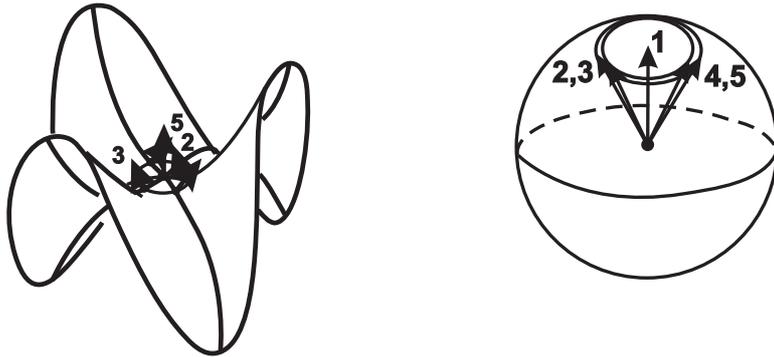


Figura A.13: A Aplicação Normal de Gauss transforma a fronteira em uma curva que dá várias voltas antes de fechar

Entretanto, quando, nas vizinhanças de um ponto $p \in S$, a aplicação normal é biunívoca, podemos tomar um domínio limitado $D \subset S$ em torno de p , e medir a área de sua imagem $N(D)$; dizemos que a área de $N(D)$ é *positiva* se a fronteira de $N(D)$ é percorrida no mesmo sentido que a fronteira de D (Figura A.14 (a)), e *negativa* no caso contrário (Figura A.14 (b)). Indicaremos por A a área de D e por \tilde{A} a área de $N(D)$.

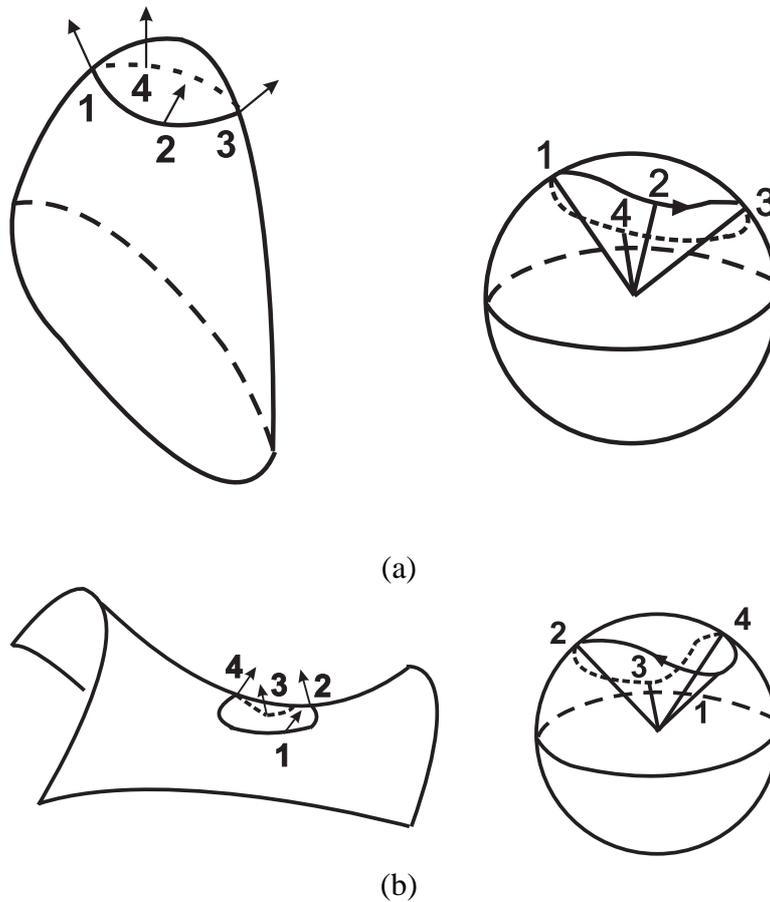


Figura A.14: O sinal da área da imagem esférica quando a aplicação normal é biunívoca

Seja agora $p \in S$. Se não existe uma região D em torno de p na qual N é biunívoca, fazemos $K(p) = 0$. Caso contrário, define-se $K(p)$ como o limite

$$\lim_{D \rightarrow p} \frac{\tilde{A}}{A} = K(p).$$

Observe que esta definição é o análogo para superfícies da definição de curvatura para curvas planas, tomando a aplicação normal no lugar da aplicação tangente (no caso de curvas planas, poderíamos ter tomado a aplicação normal sem ter alterado o resultado) e áreas no lugar de comprimentos.

A.5 Curvatura média

Agora vamos mostrar que a curvatura média H também é uma generalização da curvatura k de curvas planas. Para isto, precisamos olhar k sobre outro ponto de vista.

Seja C uma curva plana com $k(p) \neq 0$ para todo $p \in C$. Na normal a cada ponto $p \in C$ marque, no sentido da concavidade de C , um comprimento constante t . Quando p descreve C , a extremidade do segmento $\bar{p}t$ da normal a p descreve uma curva C^t que será chamada uma *paralela* a C à distância t . Observe agora que (Figura A.15) quanto maior for a curvatura em p , tanto mais rapidamente decresce o comprimento de um pequeno arco em torno de p nas curvas paralelas C^t .

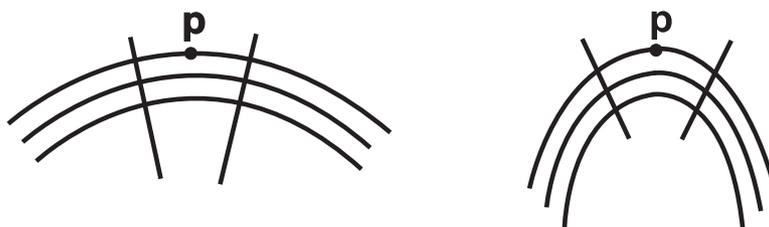


Figura A.15: Curvas paralelas

Para entender melhor esta idéia intuitiva, é conveniente generalizar um pouco a situação acima.

Diremos que uma função definida em uma curva ou superfície é *suave* (*diferenciável*) se, composta com qualquer parametrização da curva ou superfície, ela admite derivadas contínuas de todas as ordens. É possível mostrar que tal escolha não depende da parametrização.

Seja $C \subset E^2$ uma curva orientada em um plano orientado E^2 e seja $f : C \rightarrow \mathbb{R}$ uma função suave em C . Para cada $p \in C$, marque sobre a normal positiva $e_2(p)$ um comprimento $tf(p)$, $t \in (-\epsilon, \epsilon)$. A curva C^t (Figura A.16), cujos pontos são dados por

$$p^t = p + tf(p)e_2(p), \quad t \in (-\epsilon, \epsilon),$$

é ainda uma curva regular se $\epsilon > 0$ é suficientemente pequeno. Daqui em diante, admi-

remos que ϵ satisfaz esta condição.

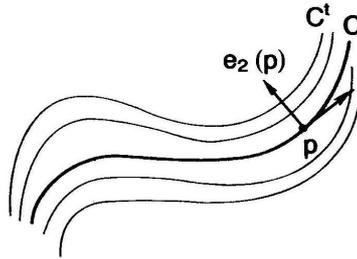


Figura A.16: Uma variação normal

A família de curvas C^t , $t \in (-\epsilon, \epsilon)$, é chamada uma *variação normal* de C dada por f . Para cada t , C^t é chamada *curva da variação*. Observe que $C^0 = C$, isto é, uma variação pode ser pensada como uma perturbação de C , que passa em C para $t = 0$. Observe também que se $f \equiv 1$, as curvas da variação são paralelas a C e recaímos na situação inicial.

Seja $L(t)$ o comprimento de arco de C^t . Pode ser provado que

$$L'(0) = - \int_C k f ds,$$

onde a integral acima tem o sentido usual de somas: Divide-se a curva em n segmentos δ_{S_i} , toma-se o valor da função $k f$ em um ponto p_i deste segmento, e passa-se o limite na soma

$$\sum_{i=1}^n (k f)(p_i) \delta_{S_i}$$

quando $n \rightarrow \infty$ e $\delta_{S_i} \rightarrow 0$.

Podemos agora dar outra interpretação para a curvatura $k(p)$ de C em p . Quando $f \equiv 1$, a expressão acima fornece

$$L'(0) = - \int_C k ds.$$

Tomando um ponto $p \in C$ e um pequeno arco \bar{C} de C de comprimento δ_S em torno de p , o teorema da média do Cálculo Integral, aplicado à expressão acima, mostra que

$$\lim_{\bar{C} \rightarrow p} \frac{L'(0)}{\delta_S} = -k(p),$$

isto é, $-k(p)$ é a velocidade de variação do comprimento de curvas paralelas a C por unidade de comprimento de C .

Esta é a interpretação geométrica de k que estávamos procurando e que foi sugerida pela Figura A.15. Um processo análogo fará aparecer H no lugar de k .

Mais explicitamente, seja S uma superfície orientada e seja $f : S \rightarrow \mathbb{R}$ uma função em S . Uma *variação normal* de S , dada por f , é uma família S^t , $t \in (-\epsilon, \epsilon)$, de superfícies de superfícies dadas por

$$S^t : p^t = p + tf(p)N(p),$$

onde N é o campo normal unitário da orientação de S . Quando $\epsilon > 0$ é suficiente pequeno, cada superfície S^t é uma superfície suave chamada *superfície da variação*. Observe que $S^0 = S$ e que $f \equiv 1$, S^t é uma *superfície paralela* a S a uma distância t .

Seja S^t uma variação normal de S , $t \in (-\epsilon, \epsilon)$, dada por $f : S \rightarrow \mathbb{R}$. Seja $D \subset S$ um domínio limitado de S e seja

$$D^t = \{p^t \in S^t; p \in D\}$$

o domínio correspondente em S^t . Definamos $A(t) = \text{área} D^t$. Pode ser provado que

$$A'(0) = - \int_D H f da.$$

Quando $f \equiv 1$, obtemos de maneira análoga a que fizemos para curvas, que $-H(p)$ é a *velocidade de variação da área de superfícies paralelas a S por unidade de área de S* . Sob este ponto de vista, a curvatura média é uma generalização da curvatura de curvas planas.

A.6 Curvatura média e divergente

A interpretação de curvatura média da Secção A.5, está relacionada com a noção de divergência de campos de vetores em E^3 .

Considere uma superfície orientada e limitada S e a variação normal S^t , $t \in (-\epsilon, \epsilon)$ de S por superfícies paralelas. Como sempre, estamos admitindo $\epsilon > 0$ suficientemente pequeno para que cada S^t seja uma superfície regular. Neste caso, é possível provar que existe um domínio $V \subset E^3$, contendo S , de modo que por cada ponto de V passa uma única superfície da variação. Defina um campo de vetores N em V tomando em cada ponto de V o vetor unitário normal à superfície S^t que passa por este ponto. Afirmamos que, se $p \in S$ então

$$(\operatorname{div} N)(p) = -H(p).$$

A prova é simples. Sejam $D \subset S$ um pequeno domínio que contém p , e D^t o domínio correspondente em S^t . O domínio “cilíndrico” de E^3 que é constituído por D , D^t e pelos segmentos das normais que ligam D a D^t será indicado por W (Figura A.17). Sejam $A(0)$ e $A(t)$ as áreas de D e D^t , respectivamente.

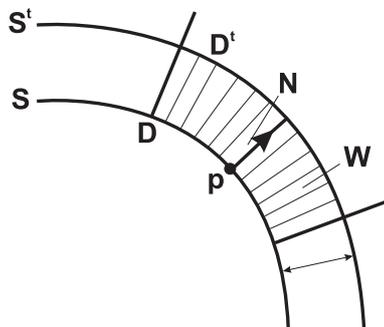


Figura A.17: Representação bidimensional do domínio cilíndrico

Como N é unitário e normal, o fluxo de N através de D é $A(0)$, e o fluxo de N através de D^t é $A(t)$. Como a superfície “lateral” não contribui para o fluxo (pois N é tangente a esta superfície), o fluxo total é $A(t) - A(0)$. O fato que queremos usar sobre a divergência é que

$$\operatorname{div} N(p) = \lim_{W \rightarrow p} \frac{\text{fluxo total}}{\text{volume } W}.$$

como o volume de W é aproximado por $tA(t)$, temos finalmente, usando a interpretação anterior de H , que

$$\operatorname{div} N(p) = \lim_{\substack{D \rightarrow p \\ t \rightarrow 0}} \frac{A(t) - A(0)}{tA(t)} = \lim_{D \rightarrow p} \frac{A'(0)}{A(0)} = -H(p),$$

como havíamos afirmado.

Apêndice B

Geometria diferencial discreta

Neste capítulo, apresentaremos algumas quantidades diferenciais úteis associadas a uma superfície discreta representada por uma malha triangular. A maioria das definições contínuas vistas no Capítulo A precisam ser reformuladas para o caso discreto usando média espacial. Podemos pensar em uma malha como sendo o limite de uma família de superfícies suaves, ou uma aproximação linear de uma superfície arbitrária. Definimos as propriedades (quantidades geométricas) da superfície em cada vértice como uma média espacial em torno deste vértice. Se estas médias são feitas de forma consistente, podemos estender a definição de curvatura ou vetor normal de superfícies para o caso de malhas discretas.

B.1 Curvatura média normal discreta

Denotaremos M por uma malha triangular, x_i um vértice da malha, e_{ij} a aresta (se existir) que conecta os vértices x_i e x_j , $N_1(i)$ os “vizinhos” (ou 1-anel de vizinhos) de x_i , ou seja, todos os vértices x_j tais que existe uma aresta e_{ij} entre x_i e x_j , e α_{ij} e β_{ij} como os ângulos opostos a aresta definida pelos vértices x_i e x_j da vizinhança do 1-anel com relação a x_i (veja a Figura B.1).

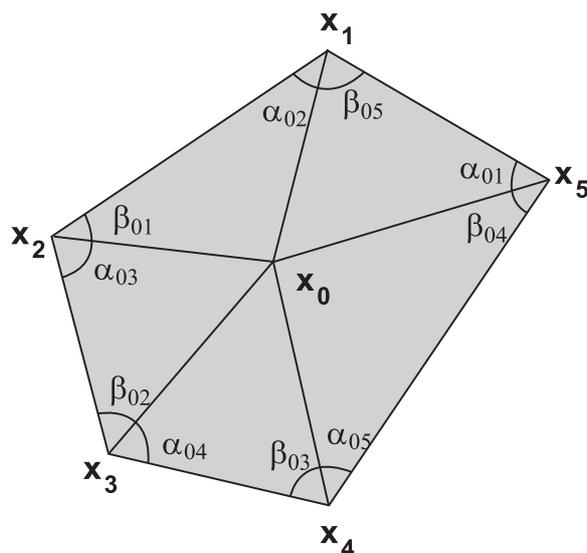


Figura B.1: 1-anel do vértice x_0 .

Como uma malha triangular é considerada uma boa representação visual de uma superfície, escolhemos um elemento finito linear em cada triângulo, isto é, uma interpolação linear entre os três vértices correspondentes de cada triângulo. Então, para cada vértice, é associado um pequeno “pedaço” da superfície (também chamado volume finito), sobre a qual a média será computada.

A diferença entre a posição de um vértice com relação ao centróide de sua vizinhança é conhecida como coordenadas laplacianas [2, 97]. O Laplaciano de uma superfície em um vértice possui componentes normal e tangente. Mesmo quando a superfície é localmente plana, a aproximação do Laplaciano dificilmente se anula [61], e pode introduzir deslocamentos indesejados sobre a superfície, dependendo da parametrização que tomarmos. Para evitar este tipo de problema, usa-se apenas as *propriedades intrínsecas* da superfície. Isto é exatamente o que o fluxo da curvatura nos fornece, ou seja, suaviza a superfície com a movimentação do vetor normal N ao longo da superfície com velocidade igual a curvatura média H .

Na prática, geralmente são utilizados dois tipos de volumes finitos (veja a Figura B.2). Em ambos os casos, a fronteira linear por partes que delimita este volume finito é obtida

pela conexão dos pontos médios das arestas e um ponto interior a cada triângulo adjacente. Este ponto no interior de cada triângulo adjacente pode ser o baricentro ou o circuncentro. A área da superfície formada usando o baricentro será denotada por A_B , enquanto que a área da superfície usando o circuncentro é conhecida como área de Voronoi e será denotada por A_V . No caso geral, quando o ponto interior for qualquer denotaremos a área da superfície por A_M .

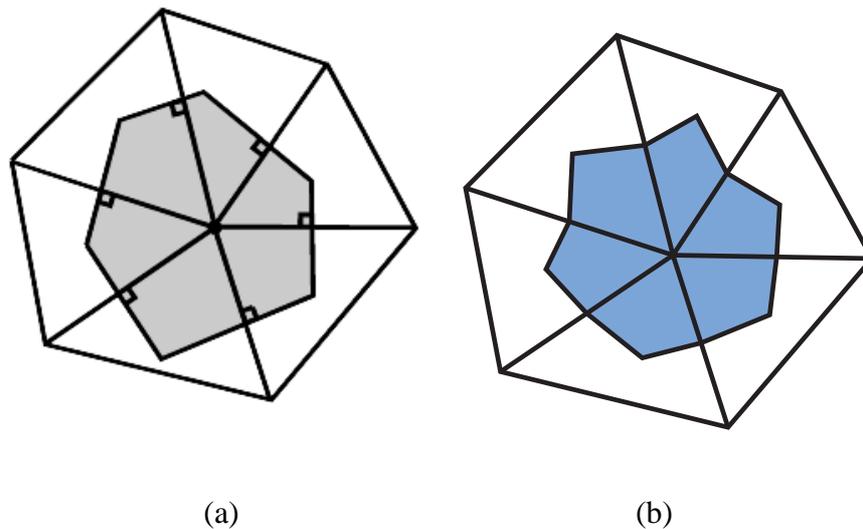


Figura B.2: (a) uma região de volume finito na malha usando *célula de Voronoi*, ou usando (b) *célula Baricêntrica*.

Como visto na Secção A.17, temos que $H = \text{div}N$. Portanto se todas normais das faces adjacentes a um vértice são as mesmas, temos que a curvatura H é nula neste ponto. Como mostra a figura B.3, se movermos o vértice central x_i em uma superfície plana, a área local da superfície não se altera. Por outro lado, se movermos o vértice para cima ou para baixo deste plano, a área local da superfície será incrementada. Portanto temos válida a propriedade de variação de área nula para uma superfície localmente plana, independentemente do aspecto das faces adjacentes ou dos comprimentos das arestas na vizinhança do vértice x_i .

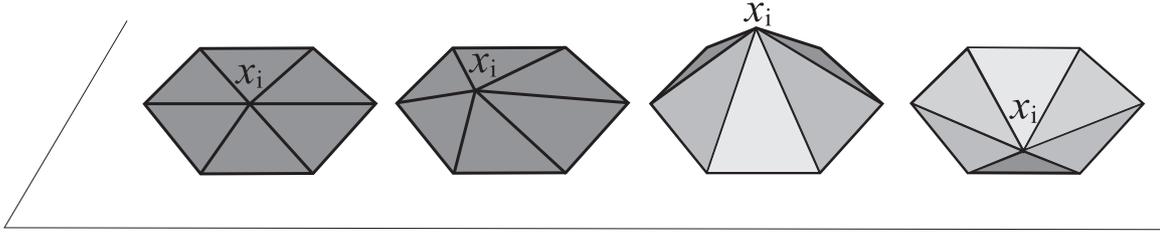


Figura B.3: A área na vizinhança do vértice x_i não muda se o vértice movimentar-se no mesmo plano do 1-anel de x_i , podendo apenas aumentar no caso contrário. Portanto temos um mínimo local, o que mostra que a derivada da área com relação a posição de x_i é zero para regiões planas.

Desejamos obter a integral da curvatura média normal sobre a área A_M . Como o operador curvatura média normal, também conhecido como operador de Laplace-Beltrami, é uma generalização do Laplaciano para variedades [24], primeiramente computamos o Laplaciano da superfície com relação ao *espaço conforme* nos parâmetros u e v . Como em [33] e [84], podemos usar a discretização da superfície como espaço de parâmetro conforme, isto é, para cada triângulo da malha, tal triângulo define a métrica local da superfície. Com essa métrica induzida, o operador de Laplace-Beltrami se simplifica para o Laplaciano $\Delta_{u,v}x = x_{uu} + x_{vv}$ [24]:

$$\int \int_{A_M} \Delta(x) dA = \int \int_{A_M} \Delta_{u,v}x dA. \quad (\text{B.1})$$

Meyer *at al* [72] usaram o teorema de Gauss, e observaram que a integral do Laplaciano sobre o pedaço de superfície que passa pelos pontos médios de cada aresta do 1-anel do domínio da triangulação, pode ser expresso em função dos valores dos nós e dos ângulos da triangulação, obtendo resultado similar a [84, 23] para o operador de Laplace-Beltrami:

$$\Delta(x_i) = \frac{2}{A_v} \sum_{x_j \in N_1(x_i)} (\cot \alpha_{ij} + \cot \beta_{ij})(x_i - x_j), \quad (\text{B.2})$$

onde α_{ij} e β_{ij} são os ângulos opostos à aresta definida pelos vértices x_i e x_j da vizinhança do 1-anel com relação a x_i como mostra a Figura B.1, e A_v é a respectiva área de Voronoi.

B.2 Área de Voronoi

Dado um triângulo não-obtuso A, B, C com circuncentro O como na Figura B.4, desejamos calcular a área de Voronoi com relação ao vértice A . Usando o fato que a soma dos ângulos internos de um triângulo é igual a π , temos que $a + b + c = \frac{\pi}{2}$, e portanto, $a = \frac{\pi}{2} - \angle A$ e $c = \frac{\pi}{2} - \angle C$.

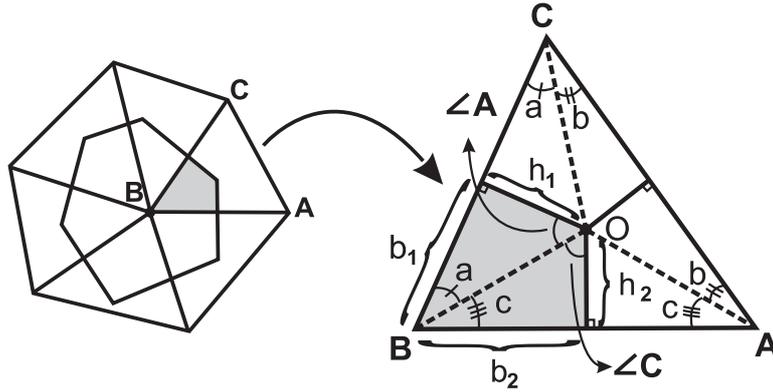


Figura B.4: Região de Voronoi em um triângulo não obtuso.

Observando a Figura B.4 temos que:

$$\begin{aligned} b_1 &= \frac{\|BC\|}{2} \\ h_1 &= \frac{\|BC\|}{2} \cotg \angle A \\ b_2 &= \frac{\|BA\|}{2} \\ h_2 &= \frac{\|BA\|}{2} \cotg \angle A. \end{aligned}$$

Portanto a área dos triângulos da área sombreada (Figura B.4) é respectivamente $\frac{\|BC\|^2}{8} \cotg \angle A$ e $\frac{\|BA\|^2}{8} \cotg \angle A$:

Logo a área sombreada da Figura B.4 é dada por

$$\frac{1}{8} (\|BC\|^2 \cotg \angle A + \|AB\|^2 \cotg \angle C). \quad (\text{B.3})$$

Denotando por α_{ij} e β_{ij} os ângulos opostos a cada aresta e_{ij} (como exemplificado na Figura B.1), e somando estas áreas para toda a vizinhança do 1-anel, podemos escrever

a área de Voronoi para um triângulo não-obtuso para um vértice x_i em função de seus vizinhos x_j como:

$$A_{Voronoi} = \frac{1}{8} \sum_{j \in N_1(i)} (\cotg \alpha_{ij} + \cotg \beta_{ij}) \|x_i - x_j\|^2. \quad (\text{B.4})$$

B.3 Extensão para malhas arbitrárias

A expressão que representa a área de Voronoi da Equação B.3 não está bem definida para triângulos obtusos, visto que neste caso, o circuncentro não se encontra no interior do triângulo. Entretanto a integral de Laplace-Beltrami dada na Equação B.1 está definida para triângulos obtusos (a única hipótese usada é que a região de volume finito passa pelos pontos médios das arestas. Ela ainda é válida mesmo no caso de triângulos obtusos). Portanto podemos simplesmente dividir o valor da integral pela área baricêntrica do volume finito em vez da área de Voronoi para vértices com ângulos obtusos para calcular a média espacial. No entanto, usamos uma área um pouco mais sutil, de modo a garantir uma cobertura perfeita de nossa superfície, e deste modo, otimizando a precisão, onde cada ponto da superfície é contado uma única vez. Definimos uma nova área da superfície para cada vértice x , denotamos por A_{Mix} da seguinte maneira: para cada triângulo não obtuso, usamos o circuncentro, e para triângulos obtusos, usamos o ponto médio da aresta oposta ao ângulo obtuso (conforme mostra a Figura B.5).

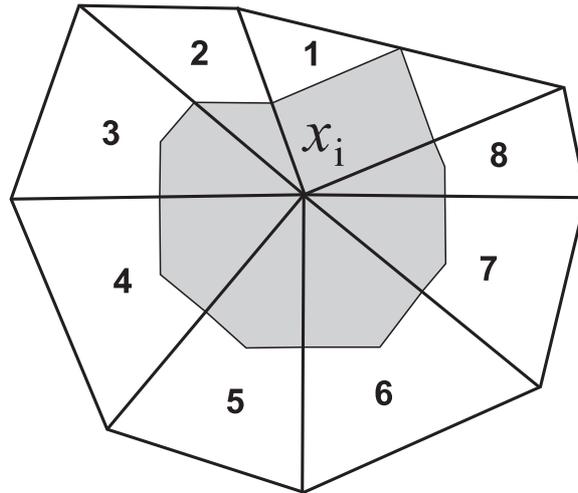


Figura B.5: Área mista entorno do vértice x_i : os triângulos 1 e 2 são obtusos, então o ponto que substitui o circuncentro é o ponto médio da aresta oposta ao ângulo obtuso

Note que a obtenção da integral da curvatura média para o 1-anel continua válida para esta área mista, já que as arestas da área continuam no interior do 1-anel e passam pelos pontos médios de cada aresta. Além disso, esta área mista cobre a superfície sem sobreposição.

Referências Bibliográficas

- [1] Marc Alexa. Local control for mesh morphing. In *Shape Modeling International*, pages 209–215, 2001.
- [2] Marc Alexa. Differential coordinates for local mesh morphing and deformation. *The Visual Computer*, 19(2-3):105–114, 2003.
- [3] Eugene L. Allgower and Stefan Gnutzmann. An algorithm for piecewise linear approximation of implicitly defined two-dimensional surfaces. *SIAM J. Numer. Anal.*, 24(2):452–469, 1987.
- [4] Alexis Angelidis, Marie-Paule Cani, Geoff Wyvill, and Scott King. Swirling-sweepers: constant volume modeling. *Graphical Models (GMOD)*, 68(4), jul 2006. Special issue on PG’04.
- [5] E. Azevedo and A. Conci. *Computação Gráfica*. Editora Elsevier Ltda., Rio de Janeiro, 2003.
- [6] I. Babuska, Carloz G., and J. E. Osborn. Special finite element methods for a class of second order elliptic problems with rough coefficients. *SIAM J. Numeric Analysis*, 31(4):745–981, 1994.
- [7] 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.

- [8] R. Beatson and G. Newsam. Fast evaluation of radial basis functions. *Computational Mathematics and Applications*, 24(12):7–20, 1992.
- [9] G. H. Bendels and R. Klein. Mesh forging: editing of 3d-meshes using implicitly defined occluders. In *Proceedings of the Eurographics/ACM SIGGRAPH symposium on Geometry processing*, pages 207–217. Eurographics Association, 2003.
- [10] J. Bloomenthal. Polygonization of implicit surfaces. *Computer Aided Geometric Design*, pages 341–335, November 1988.
- [11] Mario Botsch and Leif Kobbelt. An intuitive framework for real-time freeform modeling. *ACM Trans. Graph.*, 23(3):630–634, 2004.
- [12] Mario Botsch and Leif Kobbelt. Real-time shape editing using radial basis functions. *Computer Graphics Forum*, 24(3):611–621, 2005.
- [13] Mario Botsch, Mark Pauly, Markus Gross, and Leif Kobbelt. Primo: coupled prisms for intuitive surface modeling. In *SGP '06: Proceedings of the fourth Eurographics symposium on Geometry processing*, pages 11–20, Aire-la-Ville, Switzerland, Switzerland, 2006. Eurographics Association.
- [14] Mario Botsch, Mark Pauly, Martin Wicke, and Markus Gross. Adaptive space deformations based on rigid cells. In *Proceedings of Eurographics '07*, 2007. to appear.
- [15] Mario Botsch, Robert W. Sumner, Mark Pauly, and Markus Gross. Deformation transfer for detail-preserving surface editing. *Vision, Modeling & Visualization*, pages 357–364, 2006.
- [16] M. D. Buhmann. *Radial Basis Functions: Theory and Implementations*. Cambridge University Press, 2003.

- [17] R. E. Carlson and T. A. Foley. The parameter r^2 in multiquadric interpolation. *Computers and Mathematics with Applications*, 21(9):54–69, 1991.
- [18] R. E. Carlson and B. K. Natarajan. Sparse approximate multiquadric interpolation. *Computers and Mathematics with Applications*, 26:99–108, 1994.
- [19] 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 *Computer Graphics Proceedings*, editor, *Proceedings of ACM SIGGRAPH*, Annual Conference Series, pages 67–76, 2001.
- [20] E. E. Catmull and C. H. Clark. Recursively generated b-spline patches on arbitrary topological meshes. *Computer-Aided Design*, 10(1):123–146, 1978.
- [21] G. Chaikin. An algorithm for high speed curve generation. *Computer Graphics and Image Processing*, 3:346–349, 1974.
- [22] Mathieu Desbrun, Mark Meyer, and Pierre Alliez. Intrinsic parameterizations of surface meshes. volume 21, pages 209–218, 2002. Eurographics conference proceedings.
- [23] Mathieu Desbrun, Mark Meyer, Peter Schröder, and Alan H. Barr. Implicit fairing of irregular meshes using diffusion and curvature flow. In *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 317–324, New York, NY, USA, 1999. ACM Press/Addison-Wesley Publishing Co.
- [24] Ulrich Dierkes, Stefan Hildebrandt, Albrecht Kuster, and Ortwin Wohlrab. *Minimal Surfaces (I)*. Springer Verlag, New York, 1992.
- [25] H. Q. Dinh. *Implicit Shapes: Reconstruction and Explicit Transformation*. PhD thesis, College of Computing. Georgia Institute of Technology, August 2002.

- [26] 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.
- [27] M. P. do Carmo. *Differential Geometry of Curves and Surfaces*. Prentice Hall, 1976.
- [28] M. P. do Carmo. *Superfícies Mínimas*. 16° Coloquio Brasileiro de Matematica - IMPA, Rio de Janeiro, 1987.
- [29] D. Doo and M. Sabin. Behavior of recursive division surfaces near extraordinary points. *Computer-Aided Design*, 10(6):365–360, 1978.
- [30] Geoffrey Draper and Parris K. Egbert. A gestural interface to free-form deformation. In *Graphics Interface*, pages 113–120, 2003.
- [31] J. Duchon. Interpolation des fonctions de deux variables suivant le principe de la flexion des plaques minces. *Analyse Numeriques*, 10:5–12, 1976.
- [32] J. Duchon. Spline minimizing rotation-invariant seminorms in sobolev spaces. In *Constructive Theory of Functions of Several Variables*, W. Schempp and K. Zeller, Eds., pages 85–100, 1977.
- [33] G. Dziuk. An algorithm for evolutionary surfaces. *Numer. Math.* 58, 6:603–611, 1991. ISSN 0029-599X.
- [34] L. Euler. Researches sur la courbure des surfaces. *Memoires de l'academie des sciences de Berlin*, 16:119–143, 1760.
- [35] C. Felippa. Nonlinear finite element methods. www.colorado.edu/engineering/cas/courses.d/nfem.d/. acessado em março/2008.

- [36] M. S. Floater and A. Iske. Multistep scattered data interpolation using compactly supported radial basis functions. *Journal of Comp. Appl. Math.*, 73:65–78, 1996.
- [37] T. A. Foley. Near optimal parameter section for multiquadric interpolation. *Journal of Applied Science and Computation*, 1:19–42, 1994.
- [38] R. Franke. Scattered data interpolation: Tests of some methods. *Mathematics of Computation*, 38(157):181–200, 1982.
- [39] R. Franke, H. Hagen, and G. M. Nielson. Least square surface approximation to scattered data using multiquadric functions. *Advances in Computational Mathematics*, 2:81–99, 1994.
- [40] R. Franke and G. Nielson. Smooth interpolation of large sets of scattered data. *International Journal for Numerical Methods in Engineering*, 15(11):1691–1704, 1980.
- [41] Hongbo Fu, Oscar K.-C. Au, and Chiew-Lan Tai. Effective derivation of similarity transformations for implicit Laplacian mesh editing. *Computer Graphics Forum*, 26(1):34–45, 2007.
- [42] K. F. Gauss. General investigations on curved surfaces. (*Tradução inglesa do original em latim*), *Astérisque, Soc. Mat. de France*, 62:3–81, 1979.
- [43] F. Girosi. Some extensions of radial basis functions and their applications in artificial intelligence. *Computers and Mathematics with Applications*, 24(12):61–80, 1992.
- [44] Günther Greiner, Joachim Loos, and Wieger Wesselink. Data dependent thin plate energy and its use in interactive surface modeling. *Comput. Graph. Forum*, 15(3):175–186, 1996.

- [45] R. L. Harder and R. N Desmarais. Interpolation using surface splines. *Journ. Aircraft*, 9:189–191, 1972.
- [46] R. L. Hardy. Theory and applications of multiquadric-biharmonic method. *Computer and Mathematics with Applications*, 19:163–208, 1990.
- [47] D. Hilbert and S. Cohn-Vossen. *Geometry and the Imagination*. Chelsea Publish. Co., NY, USA, 2nd edition, 1990. 357 pages.
- [48] William M. Hsu, John F. Hughes, and Henry Kaufman. Direct manipulation of free-form deformations. *Computer Graphics*, 26(2):177–184, 1992.
- [49] CGAL <http://www.cgal.org/>. Acessado em março/2008.
- [50] FLTK <http://www.fltk.org/>. Acessado em março/2008.
- [51] LAPACK <http://www.netlib.org/lapack/>. Acessado em março/2008.
- [52] OpenGL <http://www.opengl.org/>. Acessado em março/2008.
- [53] Jin Huang, Xiaohan Shi, Xinguo Liu, Kun Zhou, Li-Yi Wei, Shang-Hua Teng, Hujun Bao, Baining Guo, and Heung-Yeung Shum. Subspace gradient domain mesh deformation. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Papers*, pages 1126–1134, New York, NY, USA, 2006. ACM Press.
- [54] T. Igarashi. Freeform user interfaces for graphical computing. In *Proceedings of 3rd International Symposium on Smart Graphics*, pages 39–48. Springer, July 2003.
- [55] 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.

- [56] Takeo Igarashi, Tomer Moscovich, and John F. Hughes. As-rigid-as-possible shape manipulation. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Papers*, pages 1134–1141, New York, NY, USA, 2005. ACM Press.
- [57] Doug James and Dinesh Pai. Artdefo - accurate real time deformable objects. In *In Proceedings of SIGGRAPH 99*, pages 65 – 72, 1999. AVI Video available at: <http://www.cs.cmu.edu/djames/movies/ArtDefo.avi>.
- [58] E. J. Kansa and R. E. Carlson. Improved accuracy of multiquadric interpolation using variable shape parameters. *Computers and Mathematics with Applications*, 21(12):99–120, 1992.
- [59] O. Karpenko, J. F. Hughes, and R. Raskar. Free-form sketching with variational implicit surfaces. *Computer Graphics Forum*, 21(3):585–594, 2002.
- [60] Youngihn Kho and Michael Garland. Sketching mesh deformations. In *SI3D '05: Proceedings of the 2005 symposium on Interactive 3D graphics and games*, pages 147–154, New York, NY, USA, 2005. ACM Press.
- [61] Leif Kobbelt, Swen Campagna, Jens Vorsatz, and Hans-Peter Seidel. Interactive multi-resolution modeling on arbitrary meshes. In *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 105–114, New York, NY, USA, 1998. ACM Press.
- [62] N. Kojekine. *Computer Graphics and Computer Aided Geometric Design by means of Compactly Supported Radial Basis Functions*. PhD thesis, Tokyo Institute of Technology, 2003.
- [63] Vladislav Kraevoy and Alla Sheffer. Mean-value geometry encoding. *International Journal of Shape Modeling*, 12(1):29–46, 2006.

- [64] 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.
- [65] Yaron Lipman, Olga Sorkine, Daniel Cohen-Or, David Levin, Christian Rössl, and Hans-Peter Seidel. Differential coordinates for interactive mesh editing. In *Proceedings of Shape Modeling International*, pages 181–190. IEEE Computer Society Press, 2004.
- [66] Yaron Lipman, Olga Sorkine, David Levin, and Daniel Cohen-Or. Linear rotation-invariant coordinates for meshes. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Papers*, pages 479–487, New York, NY, USA, 2005. ACM Press.
- [67] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. *Computer Graphics (Proceedings of ACM SIGGRAPH 87)*, 21(4):163–169, 1987.
- [68] W. R. Madych and S. A. Nelson. Multivariate inertepolation and conditionally positive definite functions. *Approximation Theory and Applications*, 4:77–89, 1988.
- [69] Martin Marinov, Mario Botsch, and Leif Kobbelt. Gpu-based multiresolution deformation using approximate normal field reconstruction. *journal of graphics tools*, 12(1):27–46, 2007.
- [70] J. R. McMahon and R. Franke. Knot selection for least squares thin plate splines. *SIAM Journal on Scientific and Statistical Computing*, 13(2):484–498, 1992.
- [71] J. Meinguet. Multivariate interpolation at arbitrary points made simple. *Z. Angew. Math. Phys.*, 30:292–304, 1979.
- [72] Mark Meyer, Mathieu Desbrun, Peter Schröder, and Alan H. Barr. Discrete differential-geometry operators for triangulated 2-manifolds. In Hans-Christian

- Hege and Konrad Polthier, editors, *Visualization and Mathematics III*, pages 35–57. Springer-Verlag, Heidelberg, 2003.
- [73] C. A. Micchelli. Interpolation of scattered data: distance matrices and conditionally positive definite functions. *Constructive Approximation*, 2:11–22, 1986.
- [74] C. A. Micchelli. Interpolation of scattered data: Distance matrices and conditionally positive definite functions. *Const. Approx.*, 2:11–22, 1986.
- [75] Jun Mitani. A simple-to-implementation method for cutting a mesh model by a hand-drawn stroke. *Eurographics Workshop on Sketch-Based Interfaces and Modeling*, pages 35–41, 2005.
- [76] Henry P Moreton and Carlo H. Sequin. Functional optimization for fair surface design. Technical report, Berkeley, CA, USA, 1992.
- [77] 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. *Proceedings of Shape Modeling International*, pages 89–98, 2001.
- [78] Andrew Nealen, Takeo Igarashi, Olga Sorkine, and Marc Alexa. FiberMesh: Designing freeform surfaces with 3D curves. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH)*, 26(3), 2007.
- [79] Andrew Nealen, Olga Sorkine, Marc Alexa, and Daniel Cohen-Or. A sketch-based interface for detail-preserving mesh editing. *ACM Trans. Graph.*, 24(3):1142–1147, 2005.
- [80] Y. Ohtake, A. Belyaev, M. Alexa, G. Turk, and H.-P. Seidel. Multi-level partition of unity implicits. *Proceedings of ACM SIGGRAPH 2003*, 2003.

- [81] Alvaro Cuno Parari, Claudio Esperança, Antonio Alberto Fernandes de Oliveira, and Paulo Roma Cavalcanti. Fast polygonization of variational implicit surfaces. In Arnaldo de Albuquerque Araújo, João Luiz Dihl Comba, Isabel Navazo, and Antônio Augusto de Sousa, editors, *Proceedings*. IEEE Computer Society, 17–20 Oct. 2004 2004.
- [82] 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.
- [83] A. Pasko and V. Savchenko. Constructing functionally-defined surfaces. In *Implicit Surfaces'95*, 1995.
- [84] Ulrich Pinkall and Konrad Polthier. Computing discrete minimal surfaces and their conjugates. *Experimental Mathematics*, 2(1):15–36, 1993.
- [85] Tiberiu Popa, Dan Julius, and Alla Sheffer. Material-aware mesh deformations. In *SMI '06: Proceedings of the IEEE International Conference on Shape Modeling and Applications 2006 (SMI'06)*, page 22, Washington, DC, USA, 2006. IEEE Computer Society.
- [86] L. Prasad. Morphological analysis of shapes. In CNLS Newsletter, editor, *Mathematical methods in CAGD III*, volume LALP-97-010-139. Center for Nonlinear Studies, Los Alamos National Laboratory, 1997.
- [87] Robert J. Renka. Multivariate interpolation of large sets of scattered data. *ACM Trans. Math. Softw.*, 14(2):139–148, 1988.
- [88] K. Salkauskas. Moving least squares interpolation with thin-plate splines and radial basis functions. *Computers and Mathematics with Applications*, 24(12):177–185, 1992.

- [89] V. V. Savchenko, A. Pasko, O. G. Okunev, and T. L. Kunii. Function representation of solids reconstructed from scattered surface points and contours. *Computer Graphics Forum*, 14(4):181–188, 1995.
- [90] R. Schaback. Creating surfaces from scattered data using radial basis functions. In M. Daehlen, T. Lyche, and L. L. Schumaker, editors, *Mathematical methods in CAGD III*. Vanderbilt Press, 1995.
- [91] Thomas W. Sederberg and Scott R. Parry. Free-form deformation of solid geometric models. In *SIGGRAPH '86: Proceedings of the 13th annual conference on Computer graphics and interactive techniques*, pages 151–160, New York, NY, USA, 1986. ACM Press.
- [92] Tevfik Metin Sezgin, Thomas Stahovich, and Randall Davis. Sketch based interfaces: early processing for sketch understanding. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Courses*, page 22, New York, NY, USA, 2006. ACM.
- [93] Alla Sheffer and Vladislav Kraevoy. Pyramid coordinates for morphing and deformation. In *3DPVT '04: Proceedings of the 3D Data Processing, Visualization, and Transmission, 2nd International Symposium on (3DPVT'04)*, pages 68–75, Washington, DC, USA, 2004. IEEE Computer Society.
- [94] N. Sivakumar and J. D. Ward. On the least squares fit by radial functions to multi-dimensional scattered data. *Numerische Mathematik*, 65:219–243, 1993.
- [95] Olga Sorkine. *Laplacian Mesh Processing*. PhD thesis, School of Computer Science, Tel Aviv University, 2006.
- [96] Olga Sorkine and Daniel Cohen-Or. Least-squares meshes. In *Proceedings of Shape Modeling International*, pages 191–199. IEEE Computer Society Press, 2004.

- [97] Olga Sorkine, Daniel Cohen-Or, and Sivan Toledo. High-pass quantization for mesh encoding. In *Proceedings of the Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*, pages 42–51. Eurographics Association, 2003.
- [98] Olga Sorkine, Yaron Lipman, Daniel Cohen-Or, Marc Alexa, Christian Rössl, and Hans-Peter Seidel. Laplacian surface editing. In *Proceedings of the Eurographics/ACM SIGGRAPH symposium on Geometry processing*, pages 179–188. Eurographics Association, 2004.
- [99] Robert W. Sumner and Jovan Popović. Deformation transfer for triangle meshes. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Papers*, pages 399–405, New York, NY, USA, 2004. ACM Press.
- [100] Robert W. Sumner, Johannes Schmid, and Mark Pauly. Embedded deformation for shape manipulation. *ACM Trans. Graph.*, 26(3):80, 2007.
- [101] Demetri Terzopoulos, John Platt, Alan Barr, and Kurt Fleischer. Elastically deformable models. In *SIGGRAPH '87: Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, pages 205–214, New York, NY, USA, 1987. ACM.
- [102] G. Turk and J. O'Brien. Variational implicit surfaces. *Technical Report GIT-GVU-99-15, Georgia Institute of Technology*, 1998.
- [103] G. Turk and J. O'Brien. Shape transformation using variational implicit functions. *Proceedings of ACM SIGGRAPH 99*, pages 335–342, 1999.
- [104] G. Turk and J. F. O'Brien. Modelling with implicit surfaces that interpolate. *ACM Transactions on Graphics*, 21(4):855–873, 2002.
- [105] L. Velho, J. Gomes, and L. H. Figueiredo. *Implicit Objects in Computer Graphics*. Springer Verlag, New York, 2002.

- [106] Wolfram von Funck, Holger Theisel, and Hans-Peter Seidel. Vector field based shape deformations. *ACM Trans. Graph.*, 25(3):1118–1125, 2006.
- [107] William Welch and Andrew Witkin. Variational surface modeling. *Computer Graphics*, 26(2):157–166, 1992.
- [108] H. Wendland. Piecewise polynomial, positive definite and compactly supported radial functions of minimal degree. *Advances in Computational Mathematics*, 10(4):389–396, 1995.
- [109] A. P. Witkin and P. S. Heckbert. Using particles to sample and control implicit surfaces. *Proceedings of ACM SIGGRAPH 94*, pages 269–278, 1994.
- [110] Z. Wu. Characterization of positive definite radial functions. In Vanderbilt University Press, editor, *T. Lyche and M. Daehlen and L. L. Schumaker*, pages 573–578, Nashville, Tenn., 1995.
- [111] Z. Wu. Multivariate compactly supported positive definite radial functions. *Adv. Comp. Math.*, 4:283–292, 1995.
- [112] 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.
- [113] G. Yngve and G. Turk. Robust creation of implicit surfaces from polygonal meshes. *IEEE Transactions on Visualization and Computer Graphics*, 8(4):346–359, 2002.
- [114] Yizhou Yu, Kun Zhou, Dong Xu, Xiaohan Shi, Hujun Bao, Baining Guo, and Heung-Yeung Shum. Mesh editing with poisson-based gradient field manipulation. *ACM Trans. Graph.*, 23(3):644–651, 2004.
- [115] Rhaleb Zayer, Christian Rössl, Zachi Karni, and Hans-Peter Seidel. Harmonic guidance for surface deformation. In Marc Alexa and Joe Marks, editors, *The*

European Association for Computer Graphics 26th Annual Conference : EUROGRAPHICS 2005, volume 24 of *Computer Graphics Forum*, pages 601–609, Dublin, Ireland, 2005. Eurographics, Blackwell.

- [116] Kun Zhou, Jin Huang, John Snyder, Xinguo Liu, Hujun Bao, Baining Guo, and Heung-Yeung Shum. Large mesh deformation using the volumetric graph laplacian. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Papers*, pages 496–503, New York, NY, USA, 2005. ACM Press.