UNIVERSIDADE FEDERAL FLUMINENSE

MARCOS DOS SANTOS RAMOS JÚNIOR

UMA FERRAMENTA PARA MODELAGEM DE FORMAS 3D A PARTIR DE DESENHOS À MÃO LIVRE USANDO SUPERFÍCIES DE CONVOLUÇÃO

NITERÓI

2012

UNIVERSIDADE FEDERAL FLUMINENSE

MARCOS DOS SANTOS RAMOS JÚNIOR

UMA FERRAMENTA PARA MODELAGEM DE FORMAS 3D A PARTIR DE DESENHOS À MÃO LIVRE USANDO SUPERFÍCIES DE CONVOLUÇÃO

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense como requisito parcial para a obtenção do Grau de Mestre em Computação. Área de concentração: Computação Visual

Orientador:
Prof. Anselmo Antunes Montenegro

NITERÓI

2012

MARCOS DOS SANTOS RAMOS JÚNIOR

UMA FERRAMENTA PARA MODELAGEM DE FORMAS 3D A PARTIR DE DESENHOS À MÃO LIVRE USANDO SUPERFÍCIES DE CONVOLUÇÃO

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense como requisito parcial para a obtenção do Grau de Mestre em Computação. Área de concentração: Computação Visual

Aprovada em Setembro de 2012.

BANCA EXAMINADORA

Prof. Anselmo Antunes Montenegro - Orientador, UFF

Prof. Marcos de Oliveira Lage Ferreira, UFF

Prof. Ricardo Farias, UFRJ

Niterói

2012

Resumo

Os programas de modelagem 3D convencionais são ferramentas extremamente poderosas e sofisticadas. Entretanto, juntamente com este grau de sofisticação, vem uma interface de usuário complexa, com uma curva íngreme de aprendizagem. Uma proposta para resolver este problema é desenvolver um programa que atue como uma ferramenta complementar e que possua uma interface de usuário mais simples e intuitiva. Uma abordagem utilizada para isso é chamada de *sketch-based modeling*, que consiste em obter malhas 3D a partir de esboços 2D desenhados pelo usuário. Esta abordagem não pretende substituir as ferramentas convencionais, mas oferecer uma interface simples para que o modelador possa prototipar rapidamente seus modelos. Este trabalho apresenta alguns dos principais trabalhos que constituem estado-da-arte na área de *sketch-based modeling* e propõe uma ferramenta que utiliza um conjunto de técnicas combinadas, que compõem um método para extrair um modelo a partir de um esboço 2D desenhado à mão livre pelo usuário e gerar uma malha 3D implicitamente representada por uma superfície de convolução.

Palavras-chave: modelagem baseada em rascunhos; superfícies de convolução; interfaces de modelagem; geometria computacional;

Abstract

Standard 3D modeling software are extremely powerful and sophisticated tools. Nevertheless, together with this degree of sophistication comes a complex user interface with a steep learning curve. A proposal to solve this problem is to develop a software that acts as a complementary tool with a simpler and more intuitive user interface. An approach used to do this is called sketch-based modeling, which consists in obtaining 3D meshes from 2D sketches drawn by the user. This approach does not intend to replace standard tools, but to offer a simple interface for the modeler to quickly prototype his models. This work describes some of the main works of the state-of-the-art in the field of sketch-based modeling and proposes a tool that uses a set of combined techniques that compose a method to extract a model from a 2D sketch hand-drawn by the user and generate a 3D mesh implicitly represented by a convolution surface.

Keywords: sketch-based modeling; convolution surfaces; modeling interfaces; computational geometry;

Sumário

Capítulo 1

Introdução

A maior desvantagem nos sistemas de modelagem convencionais é a complexidade de suas interfaces. A necessidade de usar essas interfaces sofisticadas é um obstáculo entre a ideia do usuário e a representação dessa ideia. Por esse motivo, muitos artistas preferem desenhar suas ideias com lápis e papel, o que é uma das mais simples metáforas para a representação de modelos de objetos. Por outro lado, obviamente, desenhar com lápis e papel tem a desvantagem de ser limitado a apenas um ponto de vista. O usuário não pode rotacionar o objeto e nem combinar diferentes desenhos em diferentes ângulos. O objetivo da *sketch-based modeling* é unir as poderosas capacidades da modelagem 3D à simplicidade do desenho 2D. A técnica de *sketch-based modeling* é útil para aplicações gráficas de entretenimento, como modelagem rápida de protótipos e *story boarding*. A técnica também pode ser útil em aplicações educacionais [1].

Diversas abordagens já foram exploradas nessa área [1, 7, 10, 20, 22, 27], cada uma com diferentes características. Olsen fez um *survey* sobre o estado da arte em *sketch-based modeling* [18], no qual ele compara diferentes técnicas.

Motivados por esses trabalhos anteriores, nós desenvolvemos um novo sistema de modelagem para design de protótipos baseado em geração de superfícies de convolução a partir de silhuetas desenhadas à mão livre. Nosso objetivo é oferecer uma interface simples e flexível, mas diferentemente das interfaces que analisamos em nossa pesquisa, em que o usuário é forçado a gerar o modelo sempre que desenha um traço, no nosso sistema, é permitido ao usuário escolher quando gerar o modelo. Ele pode gerar traço por traço, como em outros sistemas, ou ele pode desenhar vários traços, até mesmo em ângulos diferentes, se desejado, e só então gerar o modelo. O usuário também pode fazer uma combinação de métodos, desenhando alguns traços, gerando o modelo e depois desenhando outros traços, unindo-os ao modelo, repetindo esse processo até o ponto desejado. Desconhecemos a

existência de outro sistema em que o usuário tenha a liberdade de escolher o momento da geração.

Diferentemente do método usado em [10], que manipula diretamente as malhas, nosso sistema usa superfícies de convolução, assim como em [1, 27], que permite ao usuário desenhar formas com alta complexidade topológica, devido à natureza das superfícies implícitas.

Para permitir um controle mais preciso do *blending* sem precisar de um passo de otimização (como o usado em [27]), nós usamos a função de *blending* do o método de composição por CSG (*Constructive Solid Geometry*) descrito em [1]. O *blending* por CSG tem a vantagem de manter a estabilidade do *blending*, mesmo após várias operações de composição. Outra vantagem do *blending*, é que ele pode preservar detalhes pequenos no modelo.

Diferentemente do método descrito em [1], que define manualmente o peso de influência do esqueleto na superfície, nosso método usa uma abordagem mais genérica que calcula automaticamente esses pesos do esqueleto, como descrito em [27].

Neste trabalho, é apresentado o método usado no sistema que nós desenvolvemos. Nosso método usa superfícies de convolução para representar as modelos de objetos geradas a partir de um contorno 2D. Nós também explicamos como funciona a interface desenvolvida, e mostramos alguns resultados obtidos usando nosso sistema.

As contribuições deste trabalho são:

1. Combinar a técnica de inferência automática de pesos do esqueleto, descrita em [27], com *blending* descrito em [1], obtendo as vantagens supracitadas de ambos os métodos sem depender de definição manual de pesos, como em [1], ou de um passo de otimização global como em [27].

2. Oferecer uma interface mais flexível em que o usuário pode desenhar livremente quantos traços quiser em ângulos diferentes, e gerar o modelos apenas quando quiser. A interface proposta provê um modo interativo de fazer isso, sem forçar o usuário a gerar o modelo toda vez que ele desenha um traço.

3. Contribuição secundária: Apresentar um meta-método que funciona como uma generalização dos métodos de *sketch-based modeling*.

No capítulo 2, apresentamos os principais trabalhos da área de *sketch-based modeling*. No capítulo 3, apresentamos os conceitos matemáticos relacionados às superfícies de convolução. No capítulo 4, apresentamos um meta-método que generaliza o funcionamento das ferramentas de *sketch-based modeling* e falamos sobre diferentes técnicas que podem ser utilizadas. No capítulo 5, apresentamos em detalhes o método proposto nesse trabalho e o sistema que foi desenvolvido. No capítulo 6, apresentamos alguns exemplos, experimentos e limitações do sistema. E, por último, no capítulo 7, apresentamos nossas conclusões e idéias para trabalhos futuros.

Capítulo 2

Trabalhos relacionados

O objetivo em comum dos métodos de modelagem *sketch based*, tipicamente, é inferir ou recuperar dados a partir de traços 2D desenhados à mão livre e uma das aplicações mais diretas é a reconstrução de malhas 3D.

Alguns dos primeiros trabalhos da área [9, 10, 23, 32] são baseados em manipulação direta da malha 3D. Teddy [10] é um dos trabalhos mais importantes, pois basicamente definiu o paradigma de interface para muitos sistemas que vieram após ele. O usuário pode desenhar um traço e o sistema gera uma malha poligonal, que pode ser rotacionada com uma interface *trackball*, além de poder ser editada através de operações de extrusão, deformação (*bending*) e corte, todas controladas por traçados. Entretanto, a representação poligonal tem algumas desvantagens típicas: são mais difíceis de manipular e sua superfície tem resolução fixa, isto é, não possui uma representação capaz de se adpatar a múltiplas resoluções de malha.

Outros trabalhos usam representação da malha por superfícies paramétricas [5, 9, 29, 31], que é uma abordagem bem conhecida e simples de integrar em uma aplicação. Entretanto, a topologia de uma supefície paramétrica simples é limitada a formas homeomórficas ao plano, o que torna tediosa a tarefa de construir uma malha com topologia mais complexa [18].

Outra vertente de trabalhos [16, 21, 30] utiliza *fair surfaces* para ajustar uma superfície a um conjunto de pontos de restrição, resolvendo um problema de otimização com restrições, restrições essas, que são definidas com base no desenho do usuário. Essa técnica é bastante flexível, mas possui algumas desvantagens: primeiro, as superfícies ajustadas são normalmente muito suaves, o que limita as possibilidades, e segundo, a superfície resultante pode ser inesperada, devido ao fato de ela ser gerada pela solução de

um problema de otimização com restrições [18].

Por último, uma outra categoria de trabalhos usa superfícies implícitas para representar o modelo gerado [1, 6, 7, 12, 20, 22, 27]. Por exemplo, Parari [20] usa superfícies implícitas baseadas em RBFs para representar o modelo. Esse método produz malhas de melhor qualidade, entretanto, dado a sua propriedade isotrópica, ele não é flexível para criar malhas como extremidades agudas. Outra desvantagem do método de RBF é o alto custo computacional para resolver o sistema linear e avaliar a função RBF no passo de visualização.

Mais recentemente, a técnica que está ganhando mais popularidade dentre as superfícies implícitas, é a superfície de convolução. Alguns trabalhos já foram feitos usando essa técnica, por exemplo Alexe *et al.* [1] e Tai *et al.* [27]. Ambos usam extração de esqueleto a partir de silhuetas e usam convolução como representação do modelo.

Alexe *et al.* combina superfícies de convolução com composição CSG, o que dá uma maior precisão na combinação (*blending*) de primitivas e melhor representação de detalhes pequenos [1]. Entretanto, os pesos esqueletais são definidos manualmente em vários casos.

Tai *et al.* usa um cálculo automático de pesos e usa uma etapa de otimização global em vez da composição CSG, mas essa otimização não é boa para representar pequenos detalhes [1].

Nosso trabalho combina as técnicas de composição de primitivas com CSG [1], cálculo automático de pesos [27] e representação por superfície de convolução [24].

Capítulo 3

Superfícies de convolução

3.1 Introdução

Neste capítulo, apresentamos a definição de superfície de convolução, o conceito de *kernel* e de primitivas geométricas da convolução. O texto deste capítulo foi adaptado com base no trabalho de Sherstyuk [24].

Uma superfície de convolução é um tipo de superfície implícita, portanto, antes de definirmos superfícies de convolução, vamos definir o que é uma superfície implícita.

Uma superfície implícita S é a isosuperfície em um dado campo escalar F definido no espaço 3D como:

$$S = \{\mathbf{p} \in \mathbb{R}^3 | F(\mathbf{p}) - T = 0\}, \tag{3.1}$$

onde o parâmetro T é o valor isopotencial. A função $F(\mathbf{p})$ pode assumir qualquer forma, entretanto, uma das formas mais comuns é a soma dos potenciais de pontos:

$$F(\mathbf{p}) = \sum_{i=1}^{N} f_i(\mathbf{p}), \tag{3.2}$$

onde $f_i(\mathbf{p})$ são normalmente campos gaussianos que diminuem de acordo com a distância de seus centros. Essas formas de superfícies implícitas são conhecidas na computação gráfica como *BLOBs* ou *Metaballs* [24].

As funções f_i que compõem a equação 3.2 são tipicamente definidas como sendo monotonicamente decrescentes e com suporte compacto, isto é, com um campo de contribuição desprezível a partir de uma certa distância do centro (origem). Com uma escolha ade-

quada de funções, as origens são combinadas suavemente resultando em uma isosuperfície visualmente atrativa.

No campo da animação, as superfícies implícitas permitem obter fluidez nos movimentos, pois é possível animar apenas movendo as origens dos pontos.

Além disso, superfícies implícitas também permitem construir formas com topologias complexas, o que é difícil de se obter com outros modelos de representação.

Essas características tornaram as superfícies implícitas [2] uma técnica popular para várias tarefas de modelagem, especialmente onde as formas a serem modeladas são formas naturais, ou são superfícies suaves.

As capacidades da modelagem baseada nas equações 3.1 e 3.2 dependem das escolhas das funções de campo f_i, portanto, essa escolha é a essência do modelo.

Várias abordagens já foram descritas, sendo que as primeiras propostas usavam apenas primitivas simples, como campos esféricos ou elípticos, tanto gaussianos quanto polinomiais. Posteriormente, surgiram primitivas mais complexas como superquádricas, cilindros implícitos generalizados e superfícies de convolução [24].

Superfícies de convolução são baseadas na composição de funções de campo f_i, que são obtidas pela convolução espacial entre primitivas geométricas (como polígonos ou curvas) e uma função potencial. Essas superfícies possuem as propriedades de superposição e suporte compacto, o que as torna adequadas para a modelagem de formas livres.

3.2 Superfícies de convolução

Uma superfície de convolução é a superfície implícita definida por uma função de campo $f(\mathbf{p})$, obtida pela convolução espacial de duas funções $\mathbb{R} \mapsto \mathbb{R}^{\mathbf{3}}$, $g(\mathbf{p})$ e $h(\mathbf{p})$:

$$f(\mathbf{p}) = g(\mathbf{p}) \times h(\mathbf{p}) = \int_{\mathbb{R}^3} g(\mathbf{r})h(\mathbf{p}-\mathbf{r})d\mathbf{r} \tag{3.3}$$

onde $g(\mathbf{p})$ é a função de geometria, que define a distribuição espacial das origens dos campos e $h(\mathbf{p})$ é a função potencial, também conhecida como *kernel* de convolução, que define como o campo gerado pela origem decai com a distância. A convolução das duas funções produz a distribuição do potencial gerado por todos os pontos no espaço 3D.

Diferentemente do que ocorre com a função $h(\mathbf{p})$, que deve satisfazer certas propriedades (ver seção 3.3), a integral da convolução, por definição, não restringe a forma da

função $g(\mathbf{p})$, porém, para nossos propósitos, são consideradas apenas funções contínuas que possam ser associadas com alguma primitiva sólida P :

$$g(\mathbf{p}) = \begin{cases} 1, & \mathbf{p} \in P \\ 0, & qualquer\, outro\, lugar \end{cases} \tag{3.4}$$

Para uma primitiva de ponto, $g(\mathbf{p})$ é uma função delta e a solução da integral de convolução resulta o próprio *kernel* de convolução, isto é, $f(\mathbf{p}) = h(\mathbf{p})$. Para primitivas diferentes de um ponto, a integral de convolução precisa ser resolvida sobre o volume V da primitiva:

$$f(\mathbf{p}) = \int_V h(\mathbf{p} - \mathbf{r}) d\mathbf{r} \tag{3.5}$$

A equação acima descreve um campo genérico que é componente de uma equação de modelagem para uma superfície de convolução. Esse campo é chamado de *primitiva implícita*.

Cada primitiva implícita é construída em termos de uma forma geométrica única (ponto, reta, triângulo etc.) e dessa forma gera sua função de campo única que pode ser usada nas equações de isosuperfície.

3.3 Kernels e primitivas

Uma superfície de convolução é definida pela sua forma geométrica $g(\mathbf{r})$ e seu *kernel* de convolução $h(\mathbf{r})$. A escolha desses dois componentes é essencial e influencia bastante o resultado.

Um *kernel* utilizável deve possuir as seguintes propriedades: ser contínuo, monotônico, diminuir sua influência a um valor desprezível após uma certa distância do centro e possuir gradiente zero ou quase zero a essa distância. Ou seja, o *kernel* deve ser representado por uma função em forma de sino, similar a uma distribuição gaussiana:

$$h(\mathbf{r}) = b\ e^{-a\, r^2} \tag{3.6}$$

onde b e a são parâmetros que controlam a altura e largura da distribuição e r é a distância euclidiana ao ponto de interesse. Existem muitos *kernels* que são similares a essa definição,

porém são poucos os que podem ser convolvidos analiticamente com algo mais complexo do que uma primitiva de ponto.

3.3.1 Kernels

A escolha do *kernel* de convolução é uma tarefa extremamente importante para a definição da forma resultante da superfície [24]. Podemos citar sete funções que satisfazem todos os requisitos para a modelagem implícita.

- Função quasi-Cauchy

$$h(r) = 1/(1 + s^2 r^2)^2, \quad r > 0 \tag{3.7}$$

- Função gaussiana

$$h(\mathbf{r}) = b\ e^{-a\,r^2}, \quad r > 0 \tag{3.8}$$

- Função inversa

$$h(r) = 1/r, \quad r > 0 \tag{3.9}$$

- Função inversa ao quadrado

$$h(r) = 1/r^2, \quad r > 0 \tag{3.10}$$

- *Metaballs*

$$h(r) = \begin{cases} 1 - 3r^2, & 0 \leq r \leq \frac{1}{3}; \\ \frac{3}{2}(1-r)^2, & \frac{1}{3} < r \leq 1; \\ 0, & r > 1; \end{cases} \tag{3.11}$$

- *Soft objects*

$$h(r) = \begin{cases} 1 - (\frac{4}{9})r^6 + (\frac{17}{9})r^4 - (\frac{22}{9})r^2, & r < 1; \\ 0, & r > 1; \end{cases} \tag{3.12}$$

- Polinômio quártico em forma de W

$$h(r) = (1 - r^2)^2, \quad r < 1 \tag{3.13}$$

Dentre os sete *kernels* citados, o considerado mais flexível é o *kernel* quasi-Cauchy, por ser o único a possuir soluções analíticas para todas as primitivas analisadas (ponto, segmento de reta, arco, triângulo e plano). Em [24], todas essas primitivas são implementadas. Por outro lado, os *kernels* polinomiais são os que apresentam melhor desempenho computacional, porém só possuem soluções analíticas para as primitivas ponto e reta. Outro aspecto a ser considerado é a estética do resultado, o que pode ser uma escolha muito pessoal.

Devido ao seu aspecto estético e flexibilidade escolhemos utilizar o *kernel* quasi-Cauchy no nosso trabalho.

Capítulo 4

Processos de extração de um modelo 3D a partir de traços em 2D

4.1 Introdução

O processo de extração de um modelo 3D a partir de traços em 2D pode ser definido como uma sequência de etapas que em conjunto formam um método. Para cada uma dessas etapas existem diferentes técnicas que podem ser utilizadas. Neste capítulo, descreveremos essas etapas, na forma de um meta-método, e exemplificamos algumas possíveis variações de técnicas que podem ser utilizadas.

4.2 Etapas do processo de extração

4.2.1 Visão geral

O processo de extração consiste em três macro-etapas: obtenção dos pontos, obtenção do esqueleto e obtenção da malha. Essas macro-etapas podem ser executadas de diversas maneiras dependendo de que técnicas estão sendo utilizadas pelo método em questão, uma macro-etapa pode inclusive ser vazia, caso um determinado método não a utilize. (I) A primeira macro-etapa, a obtenção dos pontos, contém duas sub-etapas: captura e reamostragem dos pontos. A captura pode ser feita de forma direta, ou utilizando uma matriz de pixels; já a reamostragem pode ser feita usando filtragem direta ou interpolação. (II) A segunda macro-etapa, a obtenção do esqueleto, pode ser feita, por exemplo, usando CDT (*Constrained Delaunay Triangulation*) como em [10], Morfologia Matemática como em [1] ou interpolação de superfície implícita variacional 2D como descrito em [13]. (III) Por fim, a terceira e última macro-etapa, a obtenção da malha, pode ser implementada

de duas formas: manipulação direta da malha, ou por meio de superfície implícita. Neste último caso, o processo consiste em duas etapas: representação da superfície e visualização da malha. A representação pode ser construída por RBF (*Radial Basis Function*) como em [20], *Metaballs* como em [7] ou superfícies de convolução como em [1] e [27], por exemplo. Já a visualização pode ser feita usando *Marching Cubes* [15], *Algoritmo de Bloomenthal* [3] ou *Ray Tracing* [14], por exemplo. A Figura 4.1 ilustra o método geral apresentando o processo e possíveis variações de técnicas em cada uma das etapas do processo.

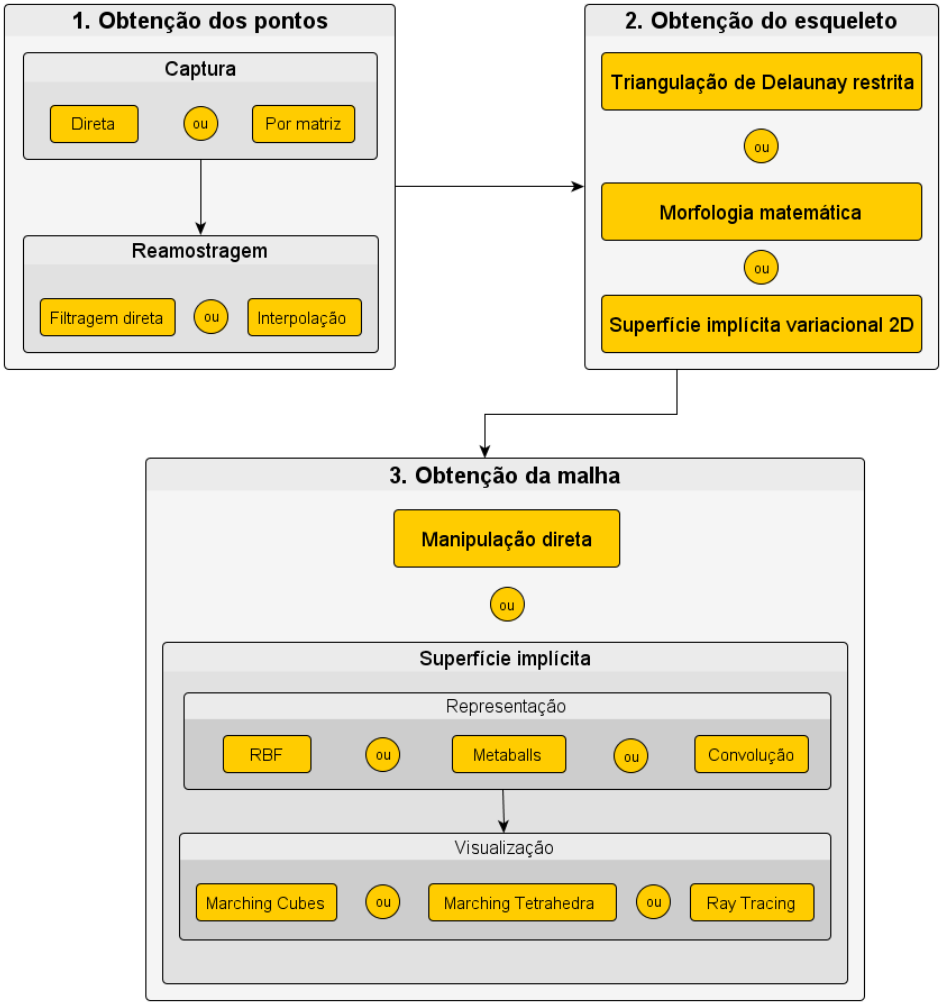

Figura 4.1: Diagrama do método geral.

Nas próximas subseções, detalhamos cada uma das etapas:

4.2.2 Obtenção dos pontos

Esta etapa consiste em obter os pontos capturados do *input* do usuário através da interface. Uma técnica bastante comum consiste em capturar diretamente as posições do

cursor do mouse conforme o usuário desenha. Outra possível técnica, conforme visto em [1], é utilizar uma matriz de pontos que será preenchida pelo desenho do usuário e depois utilizar um algoritmo de varredura para capturar os pontos.

Após a obtenção dos pontos, é necessário utilizar alguma técnica para redistribuí-los de maneira mais uniforme. Para isso existem algumas técnicas, como interpolar uma curva após a captura [10] (ver Figura 4.2), ou filtrar os pontos no momento da captura de forma a controlar a distância entre eles [27].

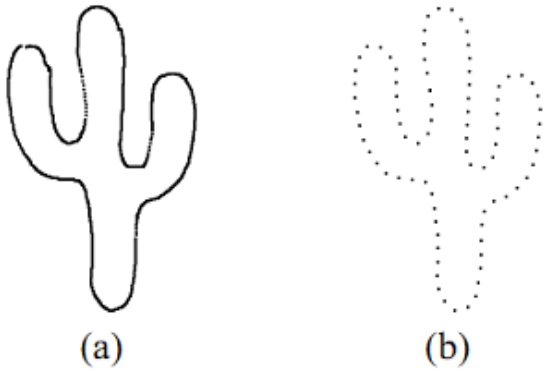

Figura 4.2: Exemplo de reamostragem de pontos. Imagem extraída de [20].

4.2.3 Obtenção do esqueleto

Uma vez que se tem uma lista de pontos já capturados e redistribuídos, é necessário obter o esqueleto. Para o caso em que a entrada do usuário é tratada como sendo o próprio esqueleto, esse processo é bastante direto, bastando considerar os segmentos extraídos do desenho como sendo os componentes do esqueleto. Porém, no caso em que o desenho do usuário é tratado como uma silhueta, o processo se torna mais complexo, necessitando de alguma técnica para extrair o esqueleto a partir dessa silhueta.

Uma classe de métodos bastante popular para isso é a classe dos métodos que utilizam uma triangulação de Delaunay com restrições (CDT) [8, 25], como base para obter os pontos que formarão o esqueleto.

A diferença básica entre esses métodos consiste em quais pontos obtidos a partir dos triângulos serão usados para compor o esqueleto. Por exemplo, em [10], são obtidos os pontos médios das arestas internas dos triângulos, em [27] são obtidos apenas os circuncentros dos triângulos e em [12] é utilizada uma técnica em que são usados circuncentros, incentros e centros de circunferências tangentes à aresta externa.

Outra diferença existente entre essas técnicas é o método de filtragem de ramos irrelevantes e segmentos redundantes. Por exemplo, em [27] são usados alguns critérios que consideram apenas a posição dos circuncentros; em [20] é usada uma métrica que considera a significância morfológica do ramo do esqueleto baseada no tamanho do mesmo em relação a aresta do triângulo onde o ramo se inicia; em [17] é usado um critério de circularidade para evitar ramos insignificantes e em [28] é usada uma técnica de retração dos ramos menos significativos.

Existem outros algoritmos que não se baseiam em uma CDT para obter o esqueleto, como, por exemplo, em [1], em que é usada uma técnica que consiste em um afinamento sucessivo da forma obtida na entrada do usuário até que se tenha um esqueleto formado por segmentos de reta e polígonos. Depois, é feita uma CDT nesses polígonos, mas diferentemente dos outros métodos, que extraem segmentos a partir dos triângulos, neste método, os próprios triângulos são usados, junto aos segmentos de reta, como primitivas geométricas.

Outro método mais recente, proposto em [13], se baseia na utilização de uma superfície implícita variacional 2D como aproximação da curva desenhada pelo usuário.

4.2.4 Obtenção da malha

Após a obtenção do esqueleto, a próxima etapa é a obtenção da malha 3D a partir desse esqueleto. Nesta etapa, existem basicamente duas classes de técnicas: as baseadas em manipulação direta da malha e as baseadas em representação implícita.

Por exemplo, em [10], a malha é manipulada diretamente da seguinte forma: após a obtenção do esqueleto, a triangulação é refeita de forma a incluir o esqueleto como arestas da nova triangulação. Após isso, o esqueleto é inflado e a malha resultante é replicada de forma espelhada, gerando uma malha 3D completamente fechada. Além disso, são feitas suavizações nas bordas usando *triangle fans*, e também é utilizado um algoritmo de refinamento de malha para melhorar a qualidade da malha obtida.

Dentre os algoritmos baseados em representação implícita existem diversas técnicas, entre elas, funções RBF [20], *Metaballs* [7] e superfícies de convolução [1, 27]. Independentemente do tipo de representação implícita escolhida, o processo consiste em definir uma função de avaliação, tendo como base o esqueleto extraído (no caso da RBF é necessário uma etapa de manipulação da malha antes de ser possível criar a função de avaliação) e depois usar um algoritmo de visualização, como o *Marching Cubes* [15], *algoritmo de*

Bloomenthal[3] ou *Ray Tracing* [14].

Capítulo 5

O método proposto

5.1 Introdução

Conforme visto no capítulo anterior, o processo de extração de um modelo 3D a partir de traços em 2D pode ter variações em todas as suas etapas, desde a forma de se obter os pontos, até a técnica utilizada para obtenção da malha. Neste capítulo, descrevemos, o método que implementamos, detalhando cada uma das etapas. O método proposto oferece duas possibilidades de interpretação do traçado feito pelo usuário; o modo silhueta, que interpreta o traçado como uma silhueta e extrai um esqueleto a partir dela, e o modo esqueleto, que interpreta o traçado como sendo o próprio esqueleto. A Figura 5.1 ilustra as etapas do nosso método e os Algoritmos 1 e 2 descrevem o método em mais detalhe.

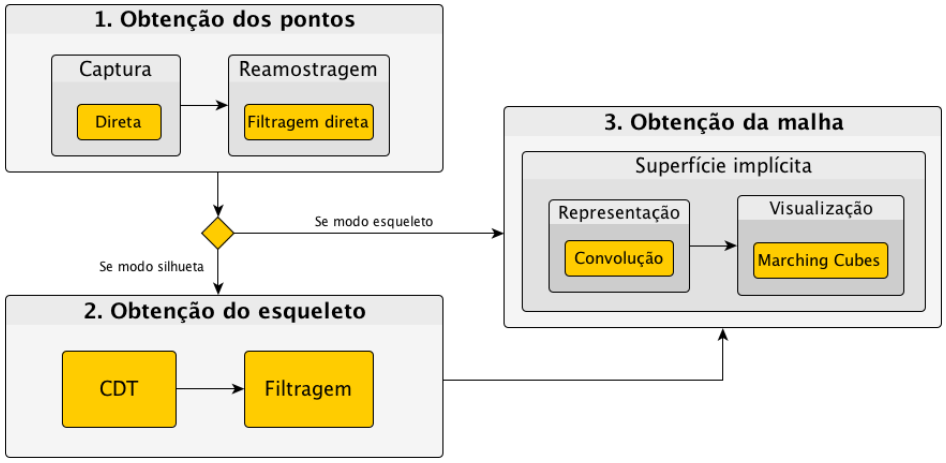

Figura 5.1: Diagrama do método proposto.

Algoritmo 1 Método proposto

```
pontos2D ← capturarEReamostrarPontos()
se modoSilhueta então
  triangulos ← gerarTriangulacaoDelaunay(pontos2D)
  esqueleto ← obterEsqueleto(triangulos)
senão se modoEsqueleto então
  esqueleto ← obterEsqueleto(pontos2D)
fim se
/* funcaoDeAvaliacao é um ponteiro de função /*
funcaoDeAvaliacao ← ponteiro(avaliarPonto) /* Função avaliarPonto definida no
Algoritmo 2 */ /* funcaoDeAvaliacao define a função de avaliação utilizada pelo Mar-
ching Cubes para detectar se há interseção da superfície com o voxel */
marchingCubes(funcaoDeAvaliacao)
```

Algoritmo 2 Função avaliarPonto(ponto, esqueleto)

```
campoTotal ← 0
para cada segmento s no esqueleto faça
  campoDaPrimitiva ← calcularConvolucao(ponto, esqueleto)
  campoTotal ← composicaoBlending(campoTotal, campoDaPrimitiva)
fim para
retorna campoTotal
```

5.2 Obtenção e reamostragem dos pontos

Inicialmente, é necessário capturar o desenho do usuário, e organizar essa informação, para que ela possa servir como entrada para o nosso método. A captura é feita de forma direta: a cada iteração do *loop* principal do sistema, se o botão do *mouse* estiver pressionado, a posição do cursor é capturada e depois passa por um filtro que determinará o que será feito com o segmento definido pelos dois últimos pontos capturados. Esse filtro é importante para que possamos obter uma distribuição de pontos adequada ao método, uma vez que os pontos obtidos através de captura da posição do cursor são bastante irregulares, apresentando muitos pontos muito próximos ou muito distantes uns dos outros, o que prejudicaria o restante do método.

O filtro implementado foi inspirado pelo descrito em [27] e consiste em garantir que todos os segmentos de reta tenham comprimento entre $\mu_0\ W$ e $\mu_1\ W$, onde W é a largura da tela de desenho, e μ_0 e μ_1 são constantes (na nossa implementação definimos $\mu_0 = 0,01$ e $\mu_1 = 0,05$). Se o segmento for maior que $\mu_1\ W$, dividimos esse segmento em n segmentos iguais de tamanho l, de forma que $l \leq \mu_1\ W$ e n seja o menor possível.

Além disso, para manter os detalhes da silhueta, é necessário identificar onde existem mais oscilações na curva, e fazer uma amostragem mais densa nessas regiões. Para isso,

o tamanho do segmento é definido de acordo com o ângulo θ (ver Figura 5.2) formado entre os vetores definidos pelos dois últimos segmentos de reta obtidos durante a captura do desenho do usuário.

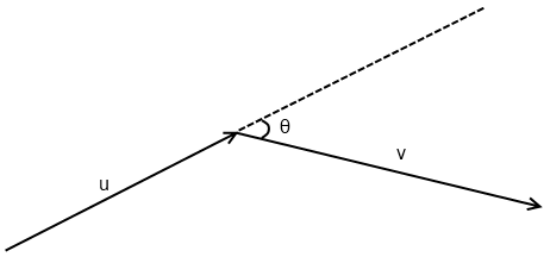

Figura 5.2: Ângulo θ. Os vetores u e v são correspondentes aos 2 últimos segmentos obtidos através da movimentação do cursor.

Para isso, é necessário definir μ de forma que $\mu_0 \leq \mu \leq \mu_1$. Essa definição é feita de seguinte forma:

$$\mu = \begin{cases} \mu_0 & \text{, se } \theta > 20° \\ \mu_1 + \theta/20 * (\mu_0 - \mu_1) & \text{, se } \theta \leq 20° \end{cases} \tag{5.1}$$

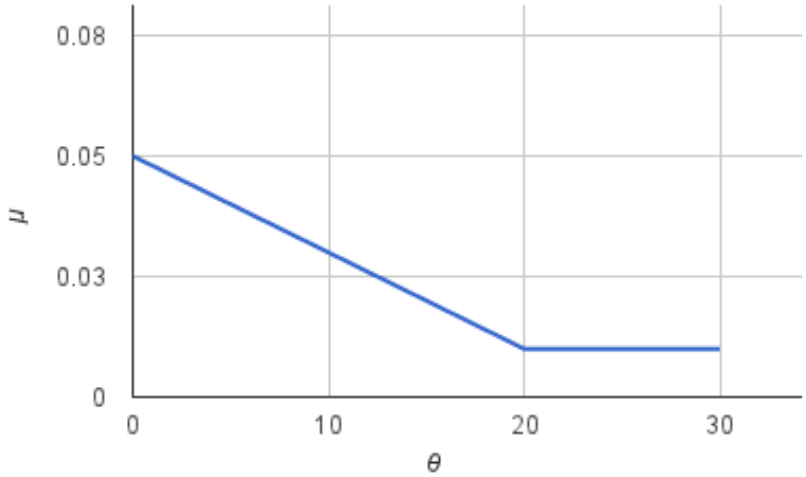

Figura 5.3: Gráfico de μ em função de θ no intervalo definido por $0 <= \theta <= 30$. Valores de $\theta < 0$ não existem na nossa aplicação e valores de μ para $\theta >= 20$ são truncados em $\mu = \mu_0$

.

Com esse critério, temos que, quanto maior θ, menor μ (ver Figura 5.3). Em termos práticos: quanto mais oscilações na curva, menores os tamanhos dos segmentos. Sendo assim, para gerar segmentos de tamanho μ W, verificamos se a soma dos tamanhos dos

dois últimos segmentos é menor que $\mu\, W$, caso seja, os segmentos são combinados em um único segmento. Dessa forma garantimos o tamanho mínimo de segmento. O Algoritmo 3 descreve o método em detalhe.

Algoritmo 3 Reamostragem dos pontos

```
/* Sejam u e v os dois últimos segmentos capturados a partir da movimentação do
cursor */
se ||v|| > μ1 W então
  divideSegmentoEmNSegmentos(v)
senão se ||u|| + ||v|| < μ W então
  uneSegmentos(u, v)
fim se
```

5.3 Geração da triangulação de Delaunay

Uma vez que os pontos da silhueta estão redistribuídos, no caso do modo silhueta, é calculada com base nesses pontos, uma Triangulação de Delaunay com Restrições (CDT - *Constrained Delaunay Triangulation*) [8, 25], cujos limites são definidos pelos segmentos da silhueta desenhada pelo usuário. A triangulação de Delaunay maximiza o menor ângulo de todos os triângulos gerados, característica essa, que se provou, empiricamente, ser suficientemente adequada para o propósito deste trabalho, que é obter o esqueleto a partir da silhueta desenhada. Embora uma CDT não seja uma triangulação de Delaunay perfeita, ela é o mais próximo possível que se pode obter quando precisamos incluir restrições à triangulação, isto é, forçar que determinadas arestas façam parte da triangulação. Mesmo com essa restrição, o resultado da CDT é bom o suficiente para o propósito deste trabalho.

Em uma CDT existem três tipos de triângulos: triângulos T (*terminal triangles*), triângulos S (*sleeve triangles*) e triângulos J (*junction triangles*). Os triângulos T têm duas arestas externas e uma interna, os triângulos S têm uma aresta externa e duas internas, e os triângulos J têm três arestas internas. Logo, os triângulos T estão nas extremidades do esqueleto, os triângulos S são os prolongamentos e os triângulos T são as junções dos prolongamentos.

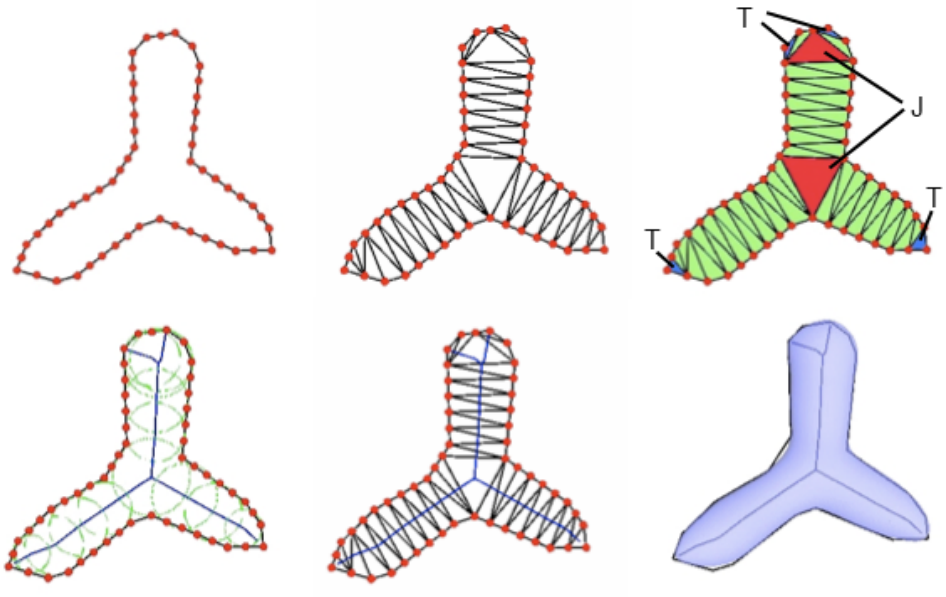

Figura 5.4: Processo de geração da superfície de convolução a partir da silhueta. Da esquerda para a direita: Silhueta com os pontos reamostrados; triangulação de Delaunay com restrições; triângulos gerados: T e J indicados por setas, e S os restantes; círculos de influência; esqueleto filtrado; superfície de convolução gerada. Imagem extraída de [12].

5.4 Obtenção do esqueleto

Depois que a triangulação é calculada, ainda no caso do modo silhueta, é necessário extrair o esqueleto. Para isso, usamos uma variação do método descrito em [12]. O método consiste em percorrer a triangulação a partir dos triângulos J até os triângulos T ou J, obtendo um ponto associado a cada triângulo. Se o triângulo for T, é obtido o incentro do triângulo. Se for S, obtém-se o centro da circunferência que é tangente ao segmento externo do triângulo e que também passa pelo ponto oposto a esse segmento. No caso de triângulo J, obtém-se o centro da circunferência que é tangente ao menor segmento do triângulo e que também passa pelo ponto oposto a esse segmento. O Algoritmo 4 demonstra com mais clareza o método. No caso do modo esqueleto, os pontos obtidos do traçado do usuário são adicionados diretamente ao esqueleto.

Além da obtenção desse primeiro esqueleto simples, é necessária uma filtragem adicional. Para essa filtragem usamos o critério de sobreposição definido em [27] e o método de eliminação de ramos morfologicamente insignificantes utilizado em [10, 20]. A verificação do critério de sobreposição é feita durante a criação do esqueleto. Ao criar um novo segmento no esqueleto, verificam-se os três últimos vértices adicionados. Sejam $\mathbf{p}$, $\mathbf{q}$ e $\mathbf{r}$, os três últimos vértices adicionados ao esqueleto.

Algoritmo 4 Obtenção do esqueleto

se $tipo(triangulo) = T$ **então**
  $ponto \leftarrow incentro(triangulo)$
  $adicionaPontoAoEsqueleto(ponto)$
senão se $tipo(triangulo) = S$ **então**
  $ponto \leftarrow centroDaCircunferenciaTangenteAoSegmentoExterno(triangulo)$
  $adicionaPontoAoEsqueleto(ponto)$
senão se $tipo(triangulo) = J$ **então**
  $ponto \leftarrow centroDaCircunferenciaTangenteAoMenorSegmento(triangulo)$
  $adicionaPontoAoEsqueleto(ponto)$
fim se

Se a circunferência de centro em **r** possui uma interseção significativa com a circunferência de centro **q**, **r** é removido. Para testar esse critério usamos:

$$||\mathbf{q} - \mathbf{r}|| + \tau R_r \leq R_q \tag{5.2}$$

onde R_r e R_q são os raios das circunferências cujos centro são, respectivamente, **r** e **q**. Na nossa implementação fixamos $\tau = 0,2$.

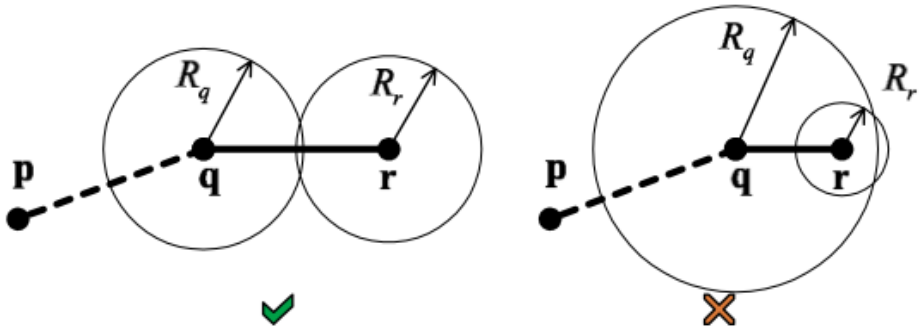

Figura 5.5: Critério de sobreposição. Imagem extraída de [27].

Por último, é feita uma varredura no esqueleto para executar o processo de eliminação de ramos morfologicamente insignificantes. O processo consiste em verificar os comprimentos do ramos que são iniciados por triângulos T e comparar o comprimento L do ramo com o raio R da circunferência associada ao triângulo J onde o ramo termina. Para isso utiliza-se o seguinte teste: $\phi L < R$ (na nossa implementação $\phi = 0,7$, ver Subseção 6.2.1 para uma análise mais detalhada). Se o resultado do teste for verdadeiro o ramo é removido. O esqueleto resultante deste processo é utilizado como base para o cálculo da superfície de convolução. A Figura 5.6 ilustra a técnica.

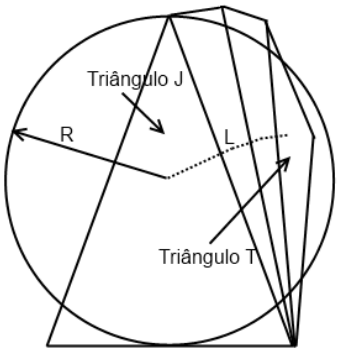

Figura 5.6: Critério de remoção de ramos irrelevantes. Se $\phi L < R$, o ramo de comprimento L (representado pela linha pontilhada) é removido.

5.5 Geração da superfície de convolução

5.5.1 Definição

Conforme visto no capítulo 3, uma superfície de convolução é um tipo de superfície implícita definida por uma integral que contém duas funções: $g(\mathbf{p})$, que é a função geométrica, e $h(\mathbf{p})$, que é a função *kernel*. $F(\mathbf{p})$ é a convolução entre $g(\mathbf{p})$ e $h(\mathbf{p})$. Desta forma, as funções $g(\mathbf{p})$ e $h(\mathbf{p})$ devem ser definidas para que se possa calcular a integral. Para o método proposto, a função geométrica utilizada é um segmento de reta e a função *kernel* utilizada é a *quasi-Cauchy*.

5.5.2 O kernel quasi-Cauchy

O *kernel quasi-Cauchy* foi escolhido porque ele é o *kernel* que apresenta o melhor resultado visual [24] e possui um *blending* suave entre suas primitivas geométricas [1].

O *kernel quasi-Cauchy* para uma primitiva geométrica representada por um ponto $\mathbf{p}$ é definida como:

$$h(\mathbf{p}) = \frac{1}{(1 + s^2 r^2)^2}, \tag{5.3}$$

onde $r = ||\mathbf{p}||$ e s é um parâmetro que define a largura do kernel.

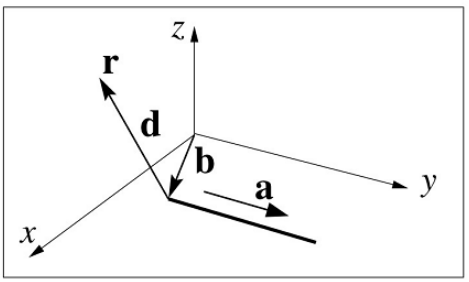

Figura 5.7: Representação vetorial da convolução de um segmento de reta

Entretanto, para utilizar segmentos de reta com primitivas geométricas, é necessário estender este modelo (veja Figura 5.7). Sendo assim, um segmento de reta com comprimento l é definido por:

$$p(t) = \mathbf{b} + t\mathbf{a}, 0 \leq t \leq l \tag{5.4}$$

onde **b** é o vetor base e **a** é o eixo de segmento normalizado.

O quadrado da distância entre um ponto arbitrário **r** e um ponto no segmento de reta é:

$$r^2 = d^2 + t^2 - 2t\mathbf{da} \tag{5.5}$$

onde **d** é um vetor definido por $\mathbf{d} = \mathbf{r} - \mathbf{b}$, e $d = ||\mathbf{d}||$. Então, para obter a função de convolução de um segmento de reta, é necessário substituir r^2 na equação geral (3.3) e resolver a integral.

Logo, o *kernel quasi-Cauchy* para uma primitiva de segmento de reta é definido por:

$$\begin{aligned} F_{line}(\mathbf{r}) &= \int_0^l \frac{dt}{(1 + s^2 r^2(t))^2} \\ &= \frac{1}{2p^2}\left[\frac{h}{s^2h^2 + p^2} + \frac{l-h}{s^2(l-h)^2 + p^2}\right] \\ &+ \frac{1}{2sp^3}\left[\tan^{-1}\left(\frac{sh}{p}\right) + \tan^{-1}\left(\frac{s(l-h)}{p}\right)\right], \end{aligned} \tag{5.6}$$

onde $h = \mathbf{d} \cdot \mathbf{a}$ e p é um termo de distância definido por:

$$p^2 = 1 + s^2(d^2 - h^2). \tag{5.7}$$

A equação (5.6) define uma superfície de convolução uniforme ao redor de um segmento de reta, porque o segmento de reta influencia a superfície com o mesmo peso, o que tipicamente produz superfícies com a forma de uma cápsula.

Entretanto, para que se possa manipular essas superfícies de uma maneira mais flexível, é necessário que se tenha um modelo matemático no qual seja possível atribuir pesos diferentes às extremidades do segmento. Isso pode ser feito usando um modelo de convolução com pesos polinomiais.

5.5.3 Modelo de convolução com pesos polinomiais

Como descrito em [27], aplicando um peso $w(t)$ à integral (5.6), temos:

$$F(\mathbf{r}) = \int_0^l \frac{w(t)}{(1 + s^2 r^2(t))^2} dt \tag{5.8}$$

e, usando uma distribuição de pesos linear, variando de w_0 a w_1, temos:

$$F(\mathbf{r}) = w_0 F_1(\mathbf{r}) + \frac{w_1 - w_0}{l} F_t(\mathbf{r}) \tag{5.9}$$

onde $F_1(\mathbf{r})$ tem a mesma forma definida na equação (5.6) e $F_t(\mathbf{r})$ é definido por:

$$F_t(\mathbf{r}) = \frac{1}{2s^2}\left[\frac{1}{s^2h^2 + p^2} + \frac{1}{s^2(l-h)^2 + p^2}\right] + hF_1(\mathbf{r}), \tag{5.10}$$

onde p é o mesmo definido na equação (5.7).

5.5.4 Calculando e ajustando pesos

No caso do modo de desenho por silhueta, para que se tenha uma superfície que se ajuste bem à silhueta desenhada, os pesos w_0 and w_1 devem ser obtidos baseados nessa silhueta. Além disso, é necessário definir um modelo em que seja possível calcular os pesos com base na distância entre o esqueleto e a silhueta (veja Figura 5.8). No modo de desenho por esqueleto, é apenas definido um peso fixo para os segmentos.

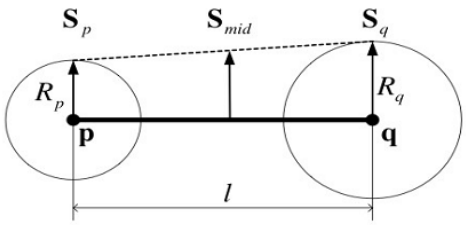

Figura 5.8: Superfície de convolução de um segmento de reta. Imagem extraída de [27].

Para cada segmento de reta, com comprimento l e com extremidades definidas pelos pontos $\mathbf{p}$ e $\mathbf{q}$, é possível inferir os pesos w_0 e w_1 de forma que a superfície se ajuste bem à silhueta. Reescrevendo a equação (5.9) na forma simétrica temos:

$$F(\mathbf{r}) = w_0 F_{l-t}(\mathbf{r}) + w_1 F_t(\mathbf{r}), \tag{5.11}$$

onde:

$$F_{l-t}(\mathbf{r}) = lF_1(\mathbf{r}) - F_t(\mathbf{r}). \tag{5.12}$$

Então, os valores da função de convolução são definidos para os dois pontos selecionados — o ponto $\mathbf{s_p}$, a uma distância R_p acima de $\mathbf{p}$ e o ponto $\mathbf{s_q}$, a uma distância R_q acima de $\mathbf{q}$ — de forma que sejam iguais ao valor de *threshold* T da superfície implícita.

$$\begin{cases} T = F(\mathbf{s_p}) = w_0 F_{(l-t)}(\mathbf{s_p}) + w_1 F_t(\mathbf{s_p}) \\ T = F(\mathbf{s_q}) = w_0 F_{(l-t)}(\mathbf{s_q}) + w_1 F_t(\mathbf{s_q}) \end{cases} \tag{5.13}$$

Considerando que $F_t(\mathbf{s_p})$ e $F_{(l-t)}(\mathbf{s_q})$ são relativamente pequenos, pois são referentes à influencia da extremidade oposta do segmento, assumimos que eles são desprezíveis e os pesos são calculados de forma aproximada, da seguinte forma:

$$\begin{cases} w_0 = \dfrac{T}{F_{(l-t)}(\mathbf{s_p})} \\ w_1 = \dfrac{T}{F_t(\mathbf{s_q})} \end{cases} \tag{5.14}$$

Para aprimorar o resultado, mais um fator, chamado k, é calculado para garantir que a superfície passa através do ponto médio $\mathbf{s_{mid}}$, que é o ponto à distância $(R_p + R_q)/2$ acima do ponto $(\mathbf{p} + \mathbf{q})/2$. O ajuste é feito multiplicando-se os pesos w_0 e w_1 pelo fator k, que é definido por:

$$k = \frac{T}{w_0 F_{(l-t)}(\mathbf{s_{mid}}) + w_1 F_t(\mathbf{s_{mid}})} \tag{5.15}$$

É também importante ressaltar que para pontos do esqueleto que sejam junções de múltiplos segmentos (pontos associados a triângulos S ou J), é necessário, antes de calcular k, dividir o peso do ponto pelo número de segmentos associados a ele (2 ou 3), para que as múltiplas contribuições sejam canceladas.

5.5.5 Combinando primitivas usando uma função de *blending*

Uma vez que a função de convolução para uma primitiva esqueletal (um segmento de reta) é definida, é necessário definir uma composição dessas primitivas para que se

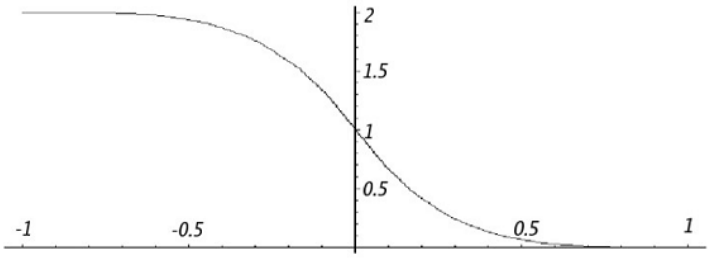

Figura 5.9: Função de *blending*

obtenha a superfície completa. Para conseguir isso, a operação de composição do modelo CSG descrito em [1] é utilizada. Essa operação é composta por uma função de *blending* e uma função de composição. A função de *blending* é uma função de *Stolte* (veja Figura 5.9) definida por:

$$g(x) = \begin{cases} 0 & \text{, if } x > R \\ 2 & \text{, if } x < -R \\ \frac{1}{R^{mn}}(R^m - x^m)^n & \text{, if } x \in [0, R] \\ 2 - \frac{1}{R^{mn}}(R^m + x^m)^n & \text{, if } x \in [-R, 0], \end{cases} \tag{5.16}$$

onde $\forall m \in \mathbb{N}$, m é ímpar e $\forall n \in \mathbb{N}$, n é par.

O parâmetro n define a precisão do *blending* e m é uma constante ($m = 1$ e $n = 2$ na nossa implementação, ver Subseções 6.2.2 e 6.2.3). R é o raio de influência da função e deve ser escolhido de forma que $R = f(\mathbf{p})$, onde $\mathbf{p}$ é um ponto na superfície.

Por fim, a função que define a união entre as primitivas esqueletais é definida como:

$$F(\mathbf{p}) = 1 - \sum_{i=0}^{n} g_i(-f_i(\mathbf{p}) + T)) \tag{5.17}$$

onde f_i é a função de convolução da i-ésima primitiva esqueletal e g_i é a função de *blending* associada a esta i-ésima primitiva.

5.5.6 Visualização da superfície usando Marching Cubes

Para visualizar a superfície gerada, nós usamos uma versão adaptativa do algoritmo *Marching Cubes*, conforme descrito em [20]. O algortimo *Marching Cubes* é um método bastante popular usado para renderizar superfícies implícitas [15]. O algoritmo funciona da seguinte forma: o espaço é dividido recursivamente em 8 cubos até atingir um certo nível de recursão n. Quanto maior o valor de n, maior a resolução de visualização. A cada subdivisão, o algoritmo verifica se o cubo intersecta a superfície e gera os triângulos de forma que eles representem a superfície o mais fielmente possível. Para melhorar a performance, nós utilizamos um *cache* de pontos avaliados para evitar que um ponto seja avaliado mais de uma vez. Na Subseção 6.2.4 é feita uma análise sobre o desempenho do *Marching Cubes*.

5.6 Interface do sistema

5.6.1 Introdução

O sistema protótipo foi implementado usando o *framework* jMonkeyEngine SDK 3.0 [11], baseado na linguagem Java. Devido ao nome do framework utilizado, o sistema protótipo recebeu o nome de *SketchMonkey*. O uso da linguagem Java traz o benefício de tornar o sistema multiplataforma (Windows, Linux e Mac). Para ilustrar o aspecto multiplataforma, neste capítulo usamos capturas de tela extraídas de sistemas Windows e Mac. A Figura 5.10 mostra uma captura de tela da interface do sistema, onde é possível observar um menu, uma barra de ferramentas, opções à esquerda e o *canvas* à direita.

Neste capítulo, apresentamos a interface do sistema desenvolvido, detalhes da implementação, além de exemplos de utilização e limitações do sistema.

5.6.2 Elementos da interface

5.6.2.1 Menu *File*

- Opção *Import*. Importa um modelo no formato *OpenCTM*.
- Opção *Export*. Exporta um modelo para o formato *OpenCTM* [19]. A ferramenta do *OpenCTM* permite conversão para outros formatos.
- Opção *Exit*. Fecha a aplicação.

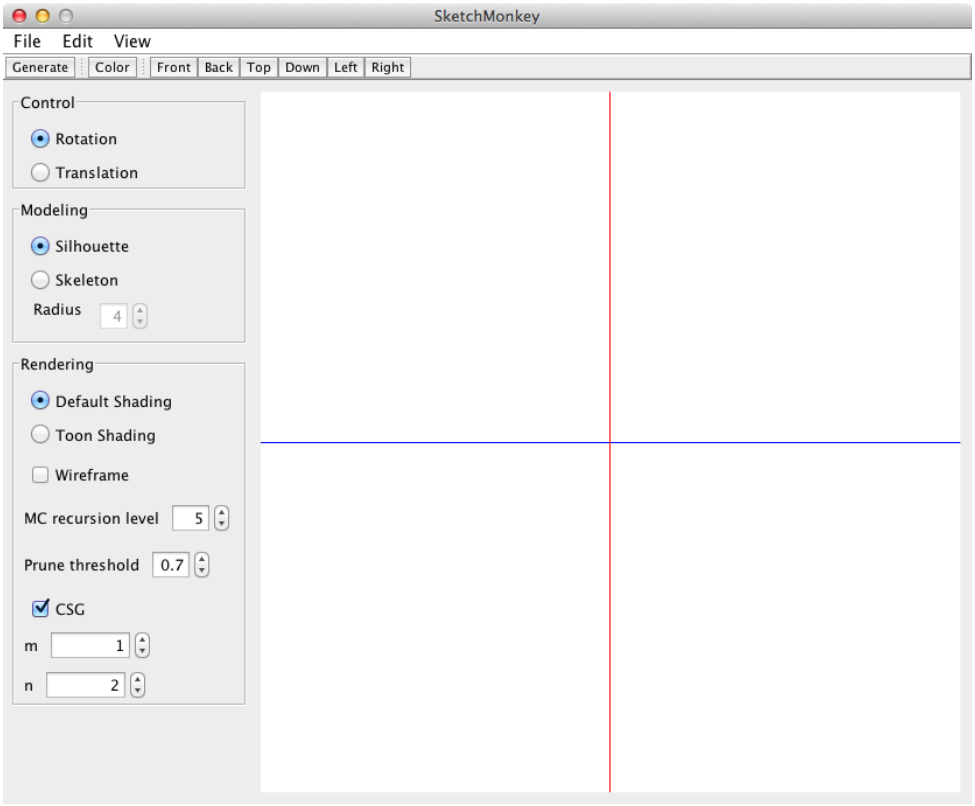

Figura 5.10: Interface do sistema SketchMonkey

5.6.2.2 Menu *Edit*

- Opção *Undo*. Desfaz o último traço feito.
- Opção *Redo*. Refaz o último traço desfeito.
- Opção *Clear*. Limpa o canvas.

5.6.2.3 Menu *View*

- Checkbox *Debug*. Habilita/Desabilita visualização de *debug* de triangulação e esqueleto.
- Checkbox *Drawing*. Habilita/Desabilita visualização dos traços feitos pelo usuário.
- Checkbox *Mesh*. Habilita/Desabilita visualização da malha.

5.6.2.4 Barra de ferramentas

- Botão *Generate*. Gera o modelo de acordo com o conteúdo atual desenhado no canvas.

- Botão *Color*. Abre uma janela de seleção de cor para ser aplicada ao modelo.
- Botão *Front*. Rotaciona o modelo de forma que se tenha uma visão frontal.
- Botão *Back*. Rotaciona o modelo de forma que se tenha uma visão traseira.
- Botão *Top*. Rotaciona o modelo de forma que se tenha uma visão superior.
- Botão *Down*. Rotaciona o modelo de forma que se tenha uma visão inferior.
- Botão *Left*. Rotaciona o modelo de forma que se tenha uma visão à esquerda.
- Botão *Right*. Rotaciona o modelo de forma que se tenha uma visão à direita.

5.6.2.5 Grupo de opções *Control*

- Radiobutton *Rotation*. Define o modo de controle do botão direito do mouse para rotação.
- Radiobutton *Translation*. Define o modo de controle do botão direito do mouse para translação.

5.6.2.6 Grupo de Opções *Modeling*

- Radiobutton *Silhouette*. Define o modo de modelagem para modelagem por silhueta.
- Radiobutton *Skeleton*. Define o modo de modelagem para modelagem por esqueleto.
- Spinner *Radius*. Define o raio utilizado pela modelagem por esqueleto.

5.6.2.7 Grupo de opções *Rendering*

- Radiobutton *Default Shading*. Define o shader utilizado como o padrão (*Phong*).
- Radiobutton *Toon Shading*. Define o shader utilizado como *Toon Shading*.
- Checkbox *Wireframe*. Alterna o modo de renderização do modelo entre renderização normal das faces e *wireframe*.
- Spinner *MC recursion level*. Define o nível de recursão utilizado pelo algoritmo *Marching Cubes*.
- Spinner *Prune threshold*. Define o coeficiente de *threshold* ϕ utilizado pelo algoritmo de remoção (*prune*) de ramos irrelevantes do esqueleto.

- CSG. Alterna o modo de composição das primitivas de convolução entre composição CSG e soma total dos campos de influência.
- Spinner *m*. Define o coeficiente *m* usado na função de *blending* do CSG.
- Spinner *n*. Define o coeficiente *n* usado na função de *blending* do CSG.

5.6.3 Workflow de utilização do sistema

A interface do nosso sistema foi projetada para oferecer um *workflow* flexível e fácil de usar. O sistema pode ser visto como um paradigma de folhas de desenho: ao rotacionar um traçado já feito, o usuário passa a desenhar em uma nova folha paralela a câmera, e assim sucessivamente. Além disso, outras funcionalidades como "Undo" e "Redo" de traços, "Clear" do canvas e controle de visualização de "Debug", "Drawing" (traçado) e "Mesh" (malha) são úteis para visualizar o que o sistema gera e controlar efeitos indesejados. Por fim, a função de "Export" permite exportar a malha para importação em outras ferramentas, aumentando a utilidade da ferramenta. O *workflow* básico do nosso sistema (ver Figura 5.11) é o seguinte:

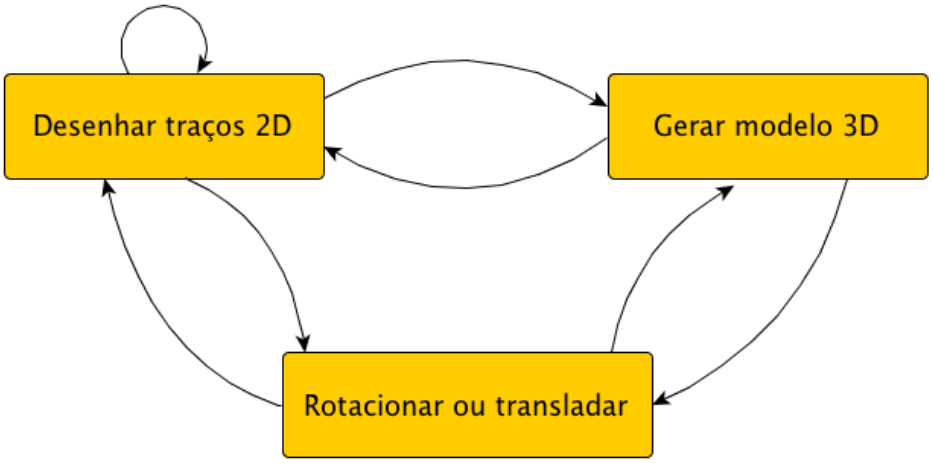

Figura 5.11: Diagrama do workflow do sistema

1. Desenhar traços: O usuário desenha um ou mais traços na tela. Os traços podem ser do tipo silhueta ou esqueleto. Além disso, o usuário pode usar os comandos "Undo" e "Redo" para desfazer e refazer traços. Note que esses traços não podem ter auto-interseções, ou o sistema irá falhar. Então o usuário pode ir para o item 2 ou o item 3.

2. Gerar modelo 3D: O usuário clica no botão "Generate", e então espera o modelo ser gerado. Os traços desenhados pelo usuário serão combinados para gerar uma superfície de convolução. Então o usuário pode ir para o item 1 ou o item 3.

3. Rotacionar ou Transladar: O usuário pode rotacionar, usando uma interface *arc-ball*, o modelo e/ou os traços na tela para obter um outro ângulo de visão para a modelagem. O usuário também pode usar o recurso de translação dos traços que ainda não geraram malhas. Então o usuário pode ir para o item 1 ou o item 2.

5.6.4 Integração do 2D com o 3D

A interface consiste de um plano, chamado de plano de desenho (ou "folha de desenho"), que interage diretamente com o espaço 3D definido por um cubo. A visualização é feita usando uma projeção ortogonal — dessa forma, o plano de desenho se projeta naturalmente sobre o espaço 3D para gerar a forma 3D.

Quando o usuário está desenhando sobre esse plano de desenho, o sistema captura continuamente as coordenadas 2D do cursor do *mouse* e as armazena numa lista de pontos 2D até que o usuário termine o desenho, quando então a lista de pontos 2D é armazenada como uma curva. Então, outro plano de desenho é gerado, cuja coordenada z é definida por um processo de *RayCasting*. Este processo consiste em lançar um raio, perpendicular ao plano de desenho atual, com origem na posição do cursor e obter a interseção mais próxima com um dos planos de desenho já existentes ou com a malha 3D. Essa interseção define a coordenada z do novo plano. A figura Figura 5.12 ilustra a técnica.

Após a criação deste novo plano, o mesmo pode ser transladado com um mecanismo de *drag and drop*. Além disso, todos os objetos podem ser rotacionados usando o *arc-ball*. A rotação é sempre relativa ao centro do eixo de coordenadas cartesianas.

Quando o usuário gera um modelo, esse modelo é visualizado na mesma interface, e o usuário pode rotacioná-lo e também pode continuar desenhando sobre o modelo gerado, diferentemente de outros sistemas, onde o modelo é automaticamente gerado quando o usuário termina um traço. Quando o usuário gera o modelo novamente, o sistema irá gerar novamente a superfície inteira, usando todos os traços que foram feitos, incluindo os que já definiam a superfície que havia sido gerada. Isso é feito dessa forma, porque, ao invés de gerar várias superfícies de convolução que se intersectam, o objetivo é gerar apenas uma superfície com primitivas que se combinam suavemente.

Todos os elementos na interface são organizados numa hierarquia de grafo de cena.

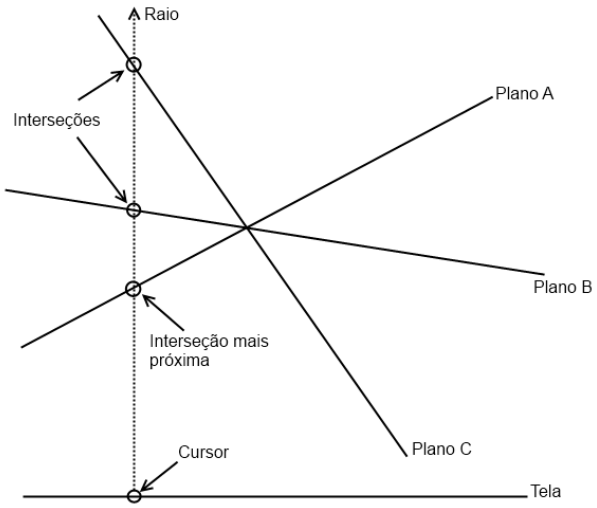

Figura 5.12: RayCasting. O raio, representado pela linha pontilhada, tem interseções com os planos A, B e C. Entretanto, a interseção mais próxima é com o plano A. Portanto, o ponto de interseção entre o raio e o plano A define a coordenada z do novo plano de desenho.

Todos os planos de desenho são filhos de um nó raiz, e todos os traços são definidos como curvas que são filhas dos planos de desenho. Ver Figura 5.13.

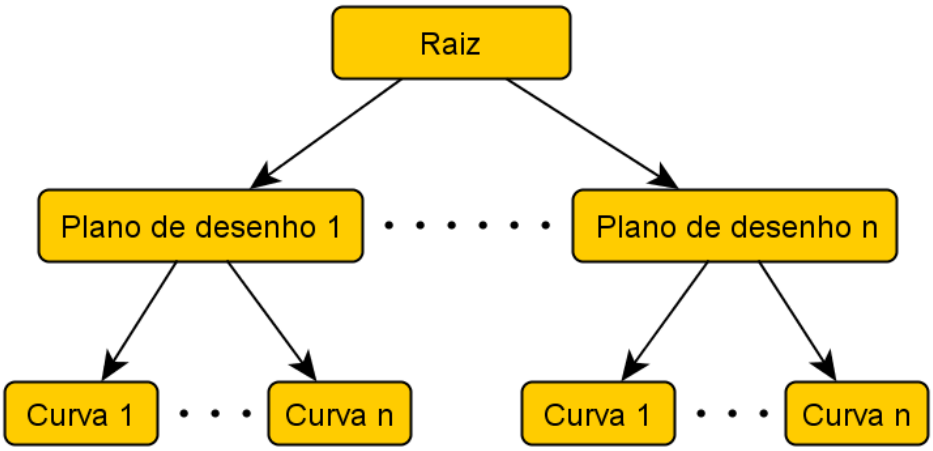

Figura 5.13: Hierarquia do grafo de cena.

Capítulo 6

Resultados

Neste capítulo apresentamos os resultados obtidos com o sistema. Primeiro ilustramos a utilização do sistema com alguns exemplos, depois mostramos os resultados de alguns experimentos com variações de parâmetros e análise de desempenho e por último abordamos as limitações encontradas.

6.1 Exemplos de utilização

Nesta seção serão apresentados alguns exemplos de uso do nosso sistema, mostrando capturas de tela do processo de modelagem e do resultado final. O exemplo 1 (Figura 6.1, Figura 6.2 e Figura 6.3) mostra um exemplo com o passo a passo de utilização do sistema, exemplificando o uso de diversos recursos. Os exemplos 2 (Figura 6.4) e 3 (Figura 6.5 e Figura 6.6) mostram mais exemplos de formas geradas com o sistema.

Figura 6.1: Exemplo 1 (parte 1): Jogador de futebol. Passo a passo de modelagem. É possível perceber a utilização de traçado sobre malha, composição apenas com traçados e desenhos de múltiplas formas desconexas.

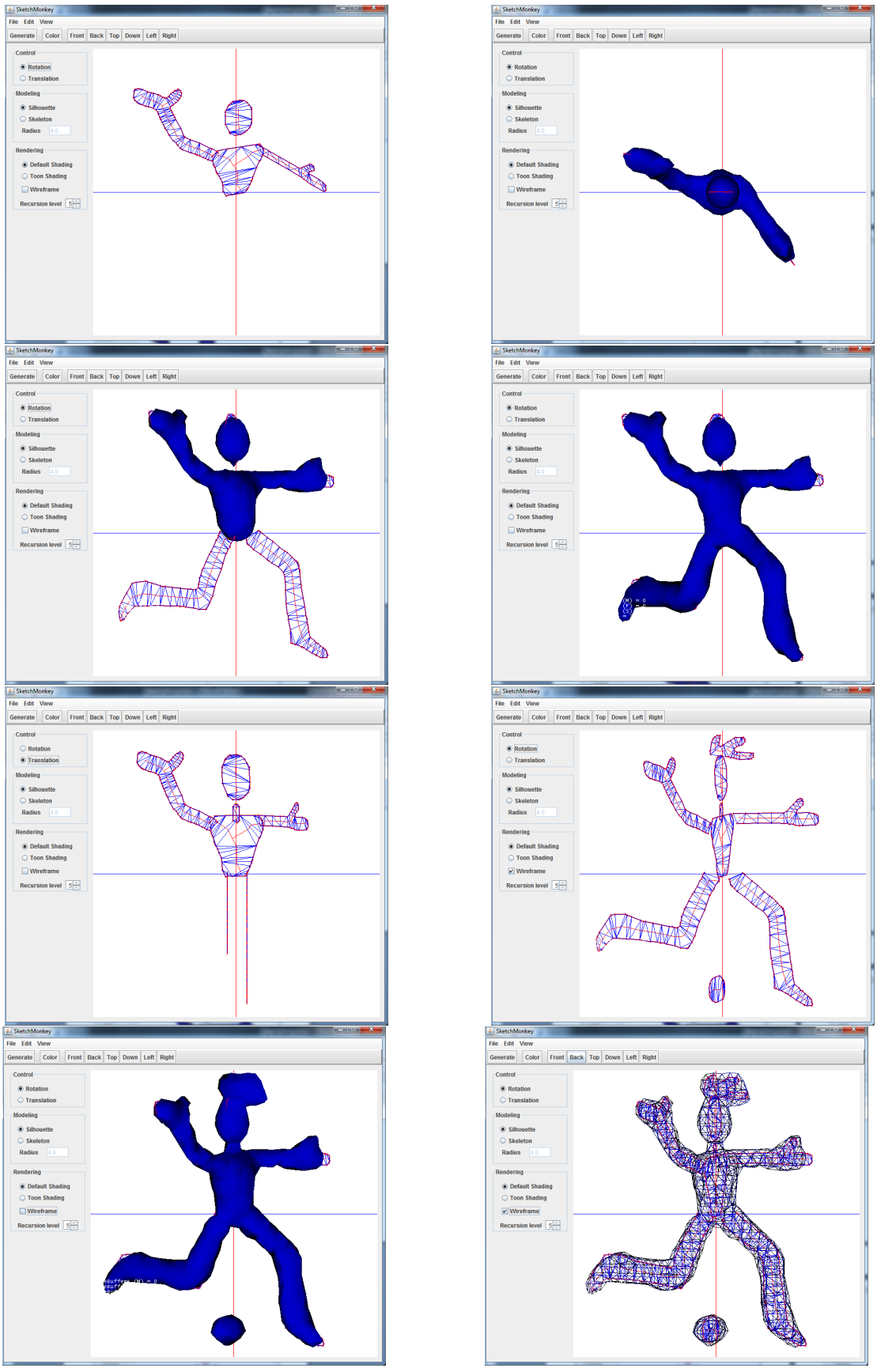

Figura 6.2: Exemplo 1 (parte 2): Jogador de futebol. Passo a passo de modelagem. É possível perceber a utilização de traçado sobre malha, composição apenas com traçados e desenhos de múltiplas formas desconexas.

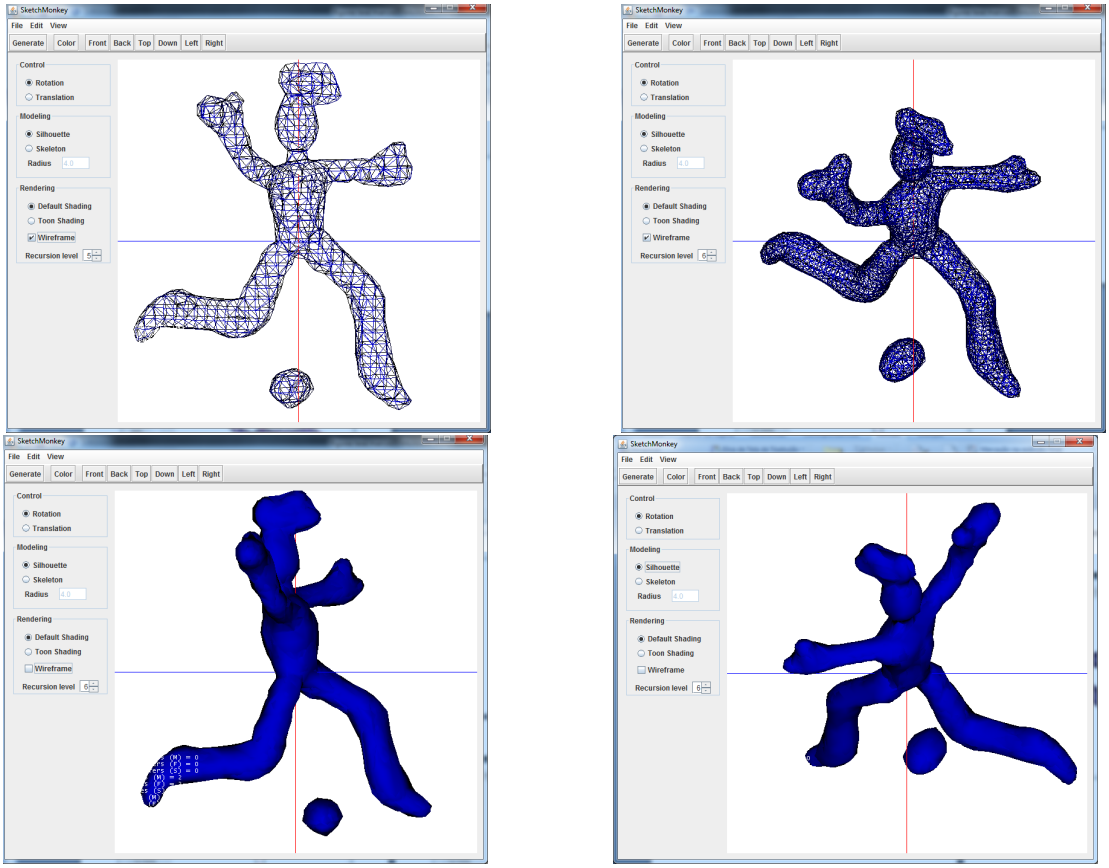

Figura 6.3: Exemplo 1 (parte 3): Jogador de futebol. Passo a passo de modelagem. É possível perceber a utilização de traçado sobre malha, composição apenas com traçados e desenhos de múltiplas formas desconexas.

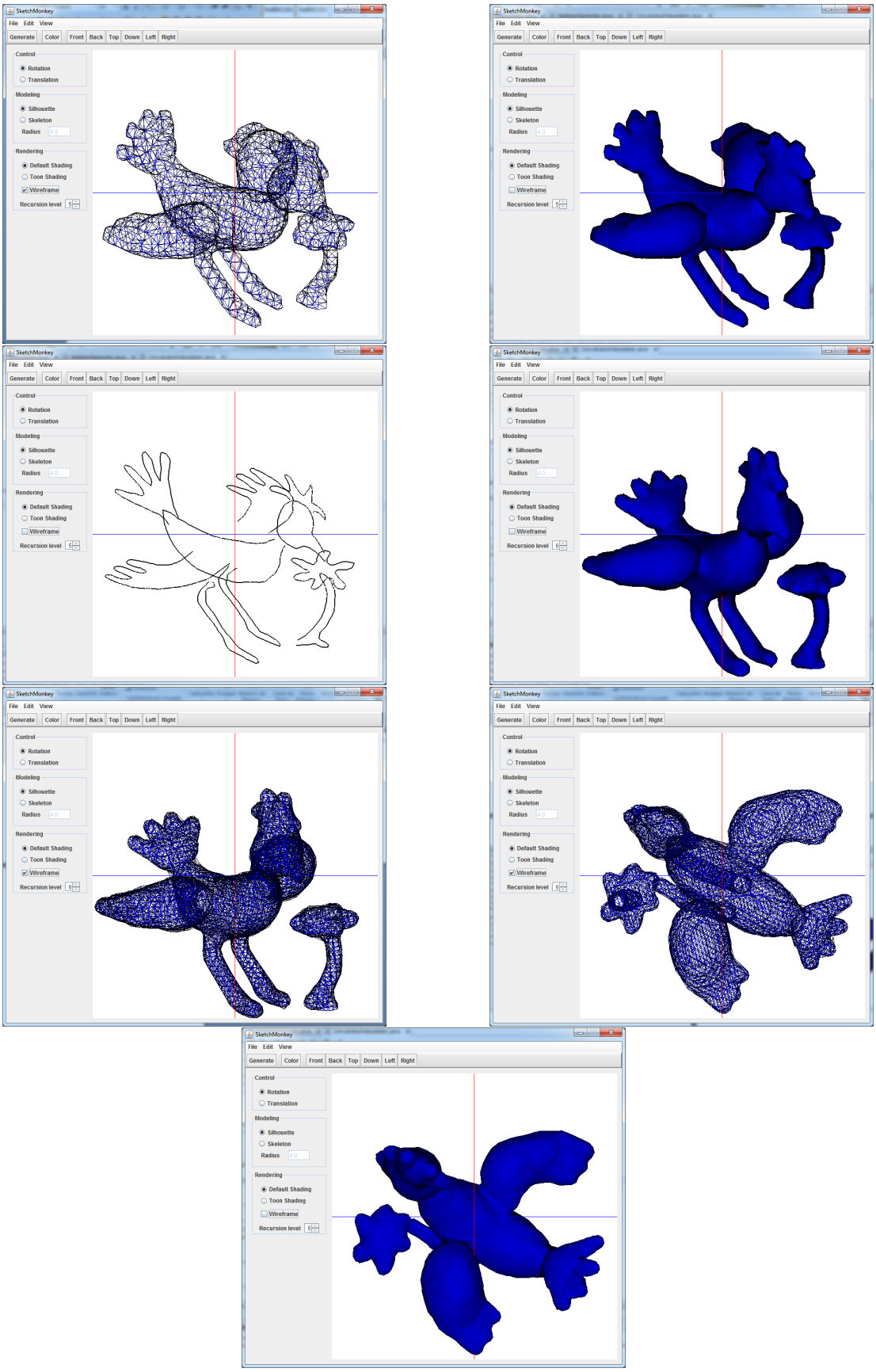

Figura 6.4: Exemplo 2: Pássaro.

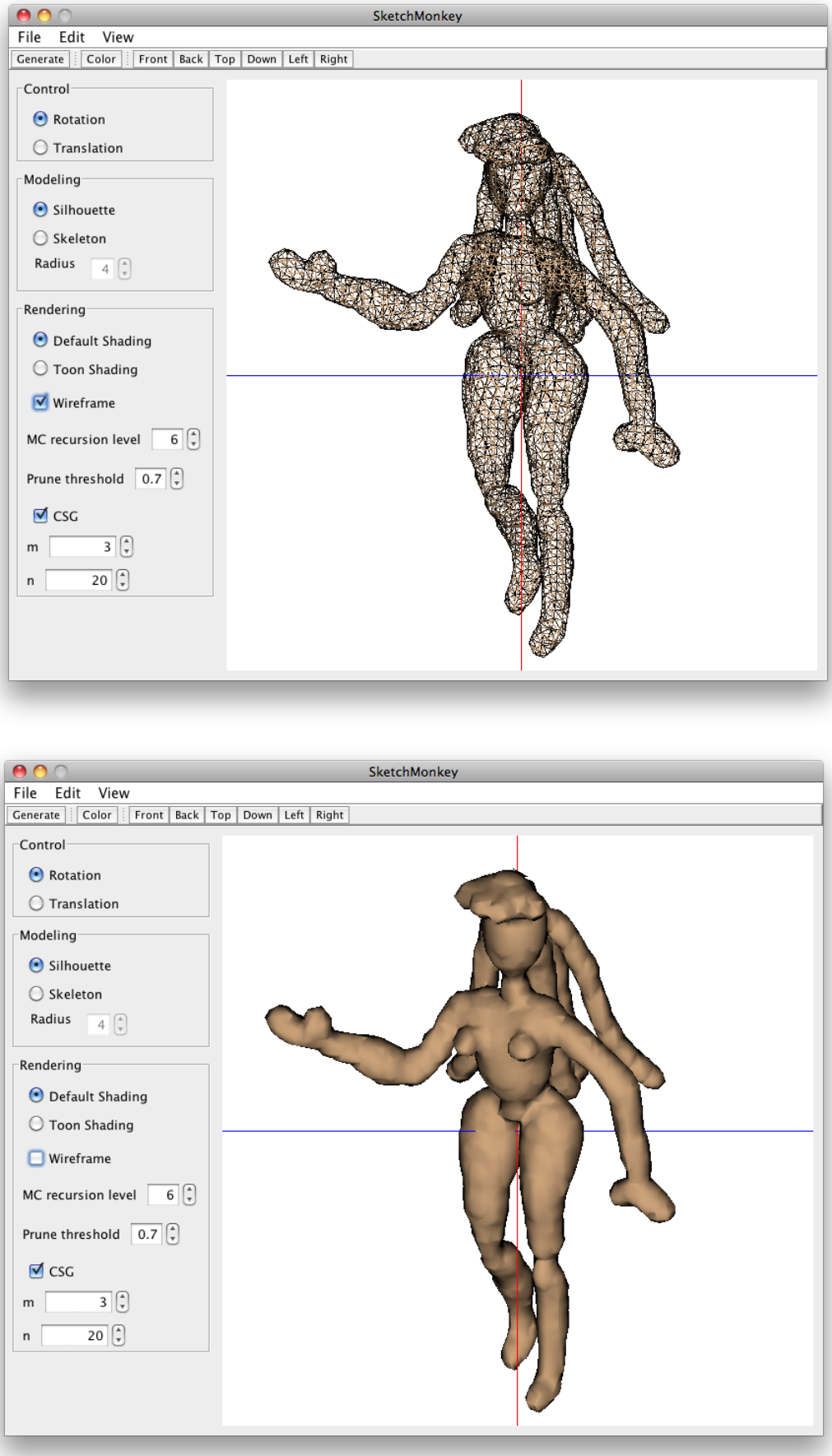

Figura 6.5: Exemplo 3: Mulata (parte 1). Usando parâmetros n=3 e m=20 na composição CSG e n=6 no Marching Cubes.

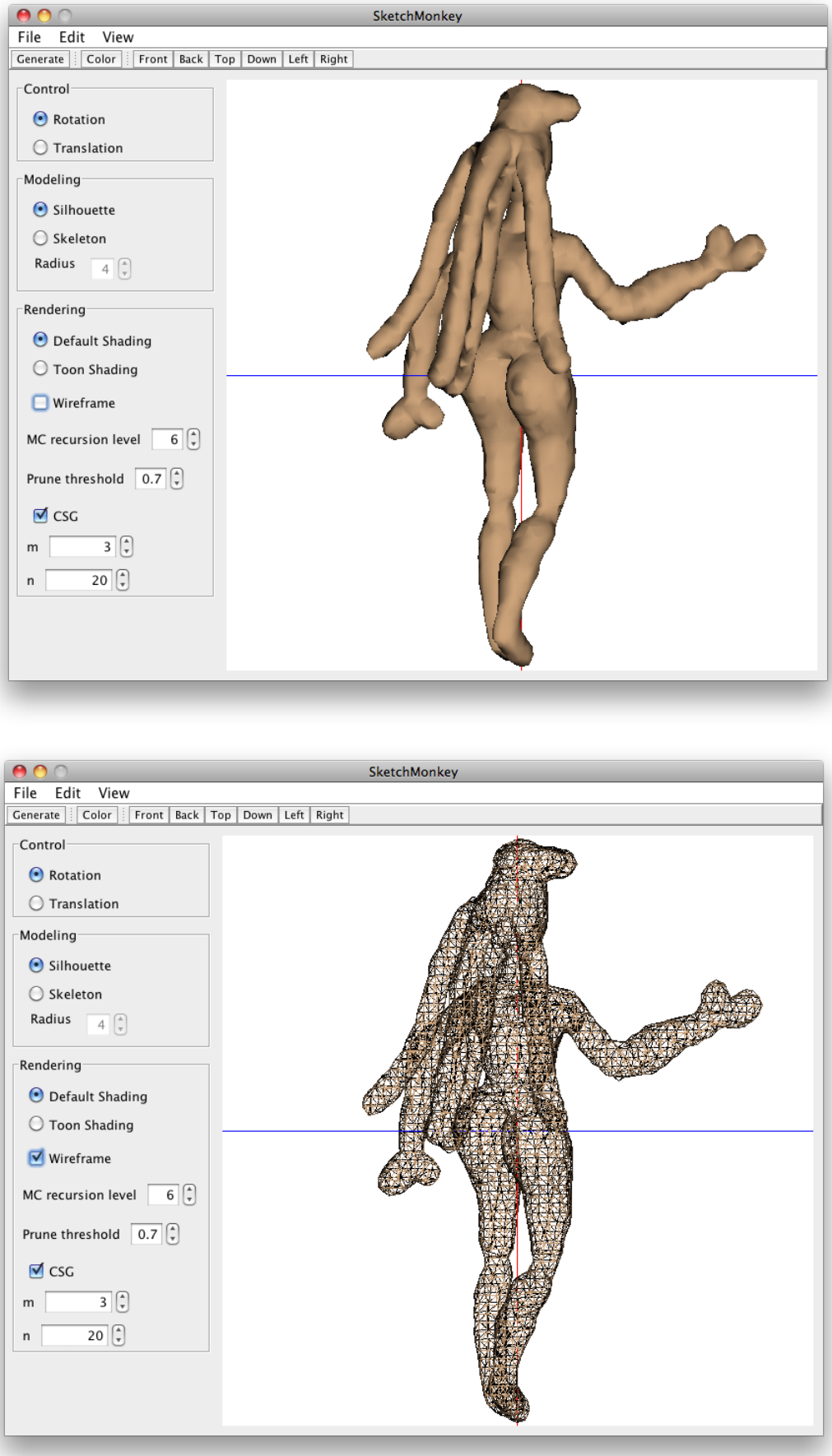

Figura 6.6: Exemplo 3: Mulata (parte 2). Usando parâmetros n=3 e m=20 na composição CSG e n=6 no Marching Cubes.

6.2 Experimentos

Nesta seção são apresentados alguns experimentos que demonstram alguns aspectos importantes deste trabalho.

6.2.1 Análise sobre a remoção de ramos irrelevantes do esqueleto

O algoritmo de remoção de ramos irrelevantes (ver Seção 5.4) é essencial para os bons resultados obtidos pelo nosso método. A Figura 6.7 mostra exemplos de esqueletos gerados com diferentes valores do parâmetro ϕ. É possível observar que ao aumentar demais o valor de ϕ, muito ramos ficam no esqueleto, de fato, se aumentarmos ϕ para um valor suficientemente grande, nenhum ramo será removido. Por outro lado, se diminuirmos demais o valor de ϕ, muitos ramos são removidos, eliminando partes morfologicamente significativas da figura.

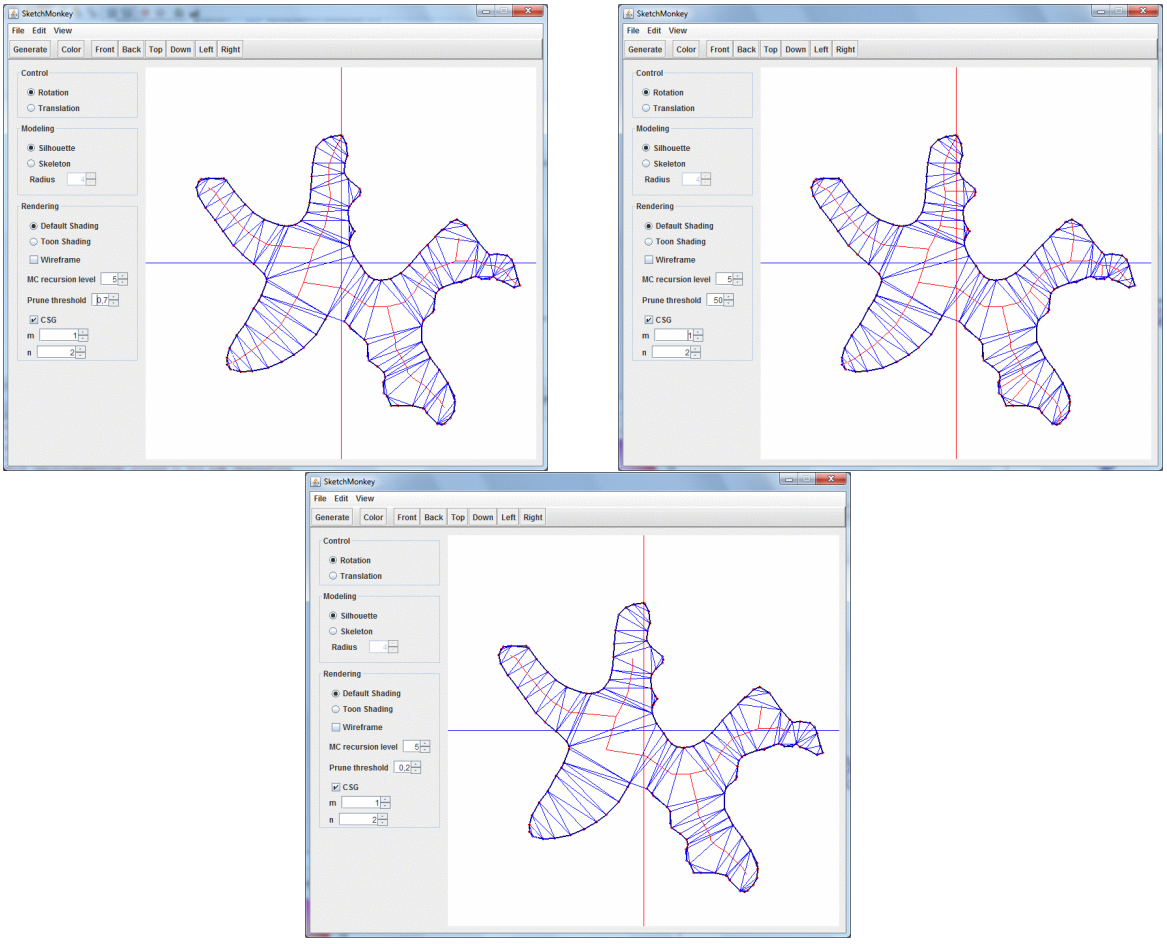

Figura 6.7: Variações de valores do parâmetro ϕ. Da esquerda para a direita: $\phi = 0,7$, $\phi = 50,0$ e $\phi = 0,2$

Empiricamente, escolhemos o valor $\phi = 0.7$ por mostrar um bom resultado prático.

6.2.2 Comparação entre composição CSG e composição por somatório simples

A composição por CSG (ver Subseção 5.5.5) é fundamental para os resultados obtidos, pois ela restringe a distância de influência dos campos gerados pelos segmentos do esqueleto. A Figura 6.8 ilustra essa importância. É possível perceber como o uso da CSG reduz o efeito de "vazamento" da superfície em relação à silhueta.

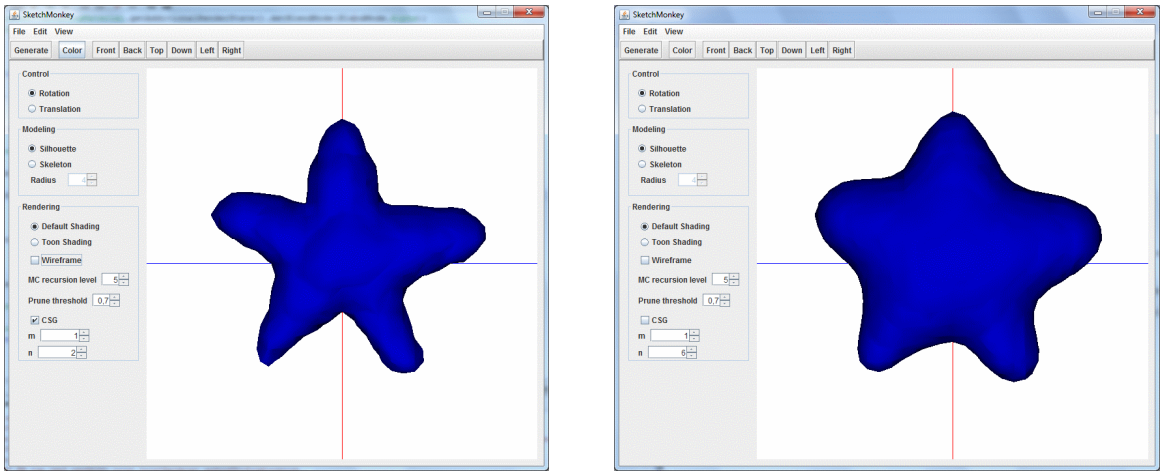

Figura 6.8: Modelo de estrela de cinco pontas. À esquerda modelo gerado usando composição CSG; à direita modelo gerado com somatório simples.

6.2.3 Comparação entre diferentes variações de parâmetros da composição CSG

A composição CSG possui parâmetros para controlar a suavidade da transição entre campos de influência. O parâmetro n controla a suavidade da transição entre os campos gerados pelas primitivas, e o parâmetro m controla a intensidade da influência que os campos causam no espaço. A Figura 6.9 ilustra o impacto causado pela variação do parâmetro n. É possível observar que a transição entre os campos perde bastante a suavidade com o valor de $n = 6$.

A Figura 6.10 ilustra o caso padrão do sistema usando os parâmetros $m = 1$ e $n = 2$, a Figura 6.11 ilustra como o aumento do parâmetro n reduz a suavidade da transição entre os campos, porém aumenta o ajuste à silhueta. Já a Figura 6.12 ilustra como aumentar o parâmetro m aumenta a influência dos campos. A Figura 6.13 ilustra mais uma vez que o aumento do parâmetro n torna mais abrupta a transição entre os campos.

A Figura 6.14 ilustra um caso extremo: definindo $n = 50$, é produzida uma malha com transições excessivamente abruptas entre os campos.

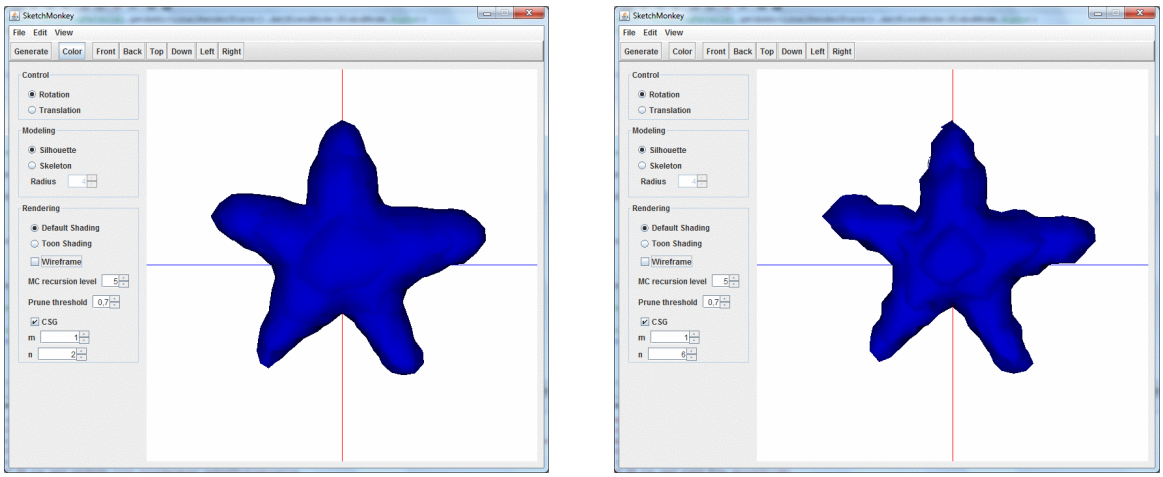

Figura 6.9: Modelo de estrela de cinco pontas. À esquerda modelo gerado usando $n = 2$; à direita modelo gerado usando $n = 6$.

A Figura 6.15 mostra outro caso extremo: definindo $m = 9$, os campos passam a influenciar o espaço em demasia, gerando formas excessivamente arredondadas e extrapolando bastante a silhueta. Para efeito de comparação, a Figura 6.16 mostra o mesmo modelo gerado através de composição por somatório simples. É possível observar que a forma extrapola menos a silhueta do que a forma da Figura 6.15, isso demonstra que para valores suficientemente grandes de m, a influência dos campos chega a ser até mesmo maior do que quando é usada a composição por somatório simples.

A Figura 6.17 ilustra um exemplo de forma gerada usando o modo de desenho por esqueleto, com $m = 1$ e $n = 50$. Nesse exemplo é possível observar claramente as transições bruscas entre os campos que são geradas por valores altos do parâmetro n.

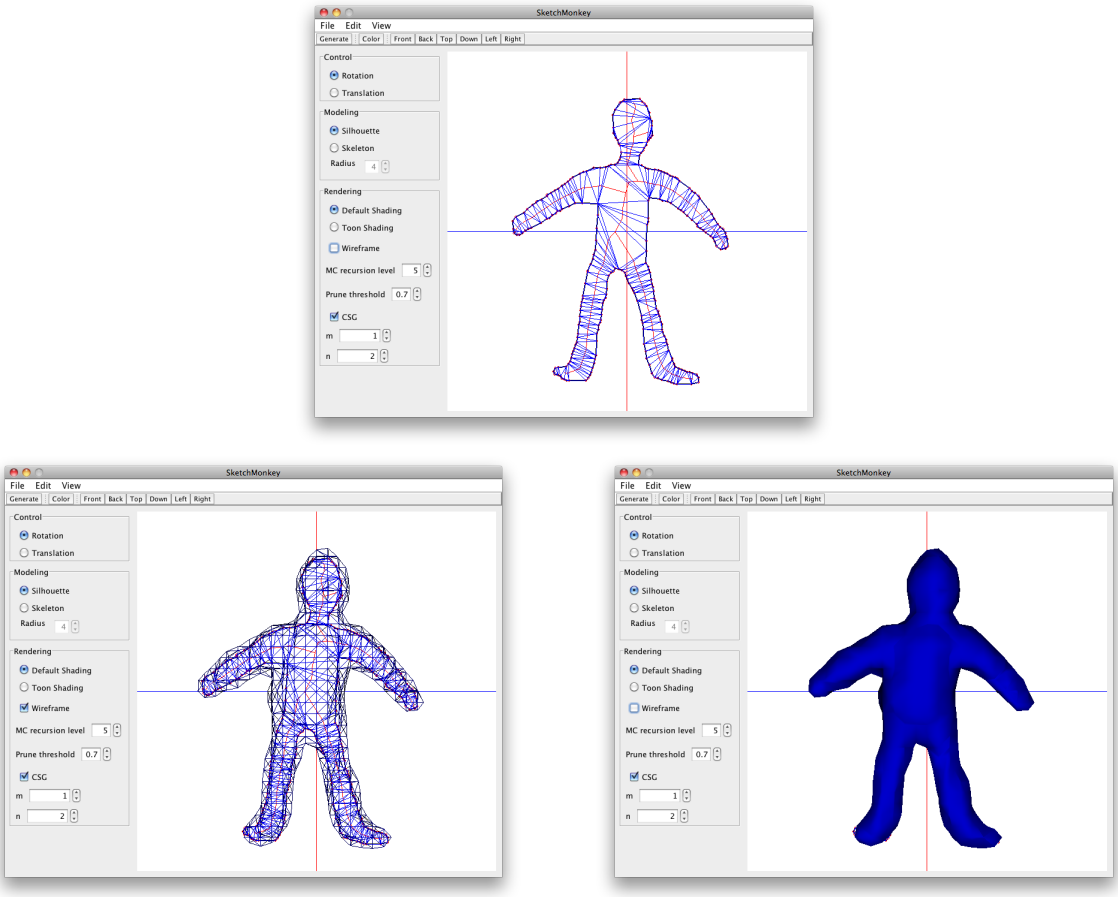

Figura 6.10: Modelo de silhueta humana. Na primeira linha: silhueta desenhada; na segunda linha: modelo gerado usando $m = 1$ e $n = 2$.

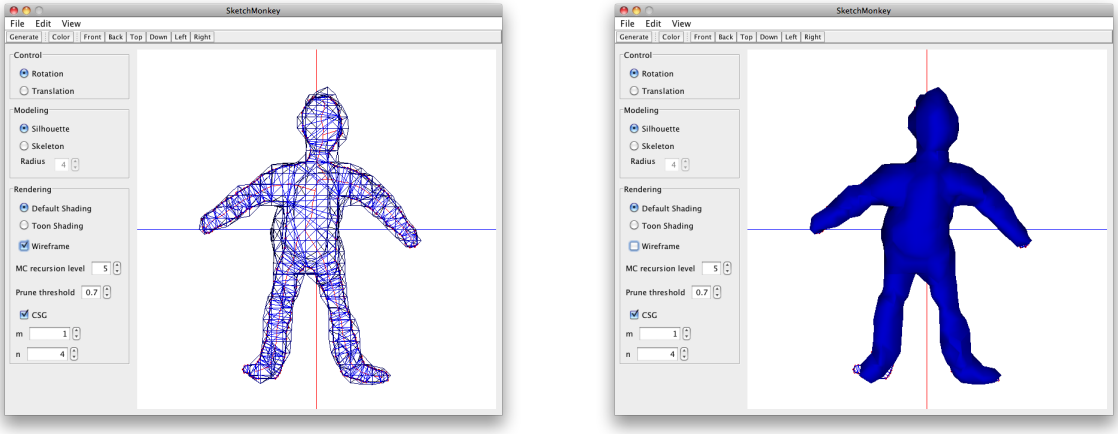

Figura 6.11: Modelo de silhueta humana. Modelo gerado usando $m = 1$ e $n = 4$.

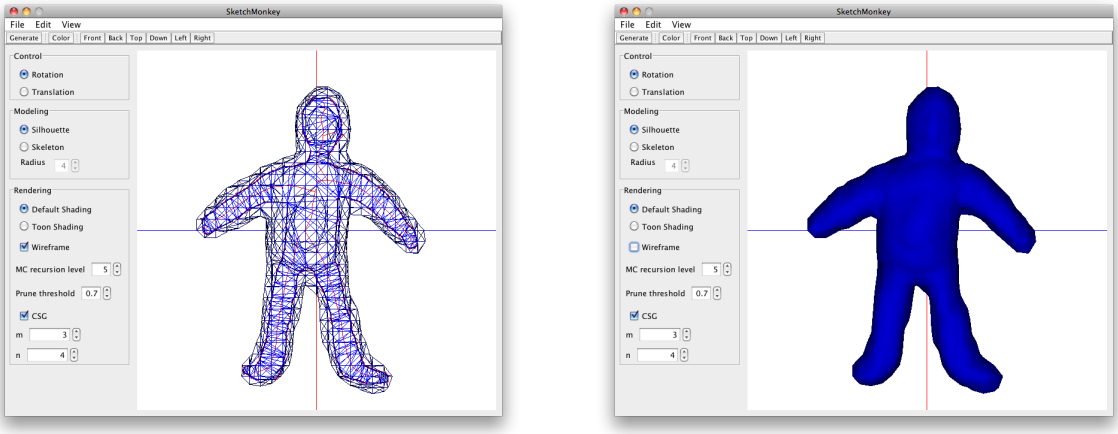

Figura 6.12: Modelo de silhueta humana. Modelo gerado usando $m = 3$ e $n = 4$.

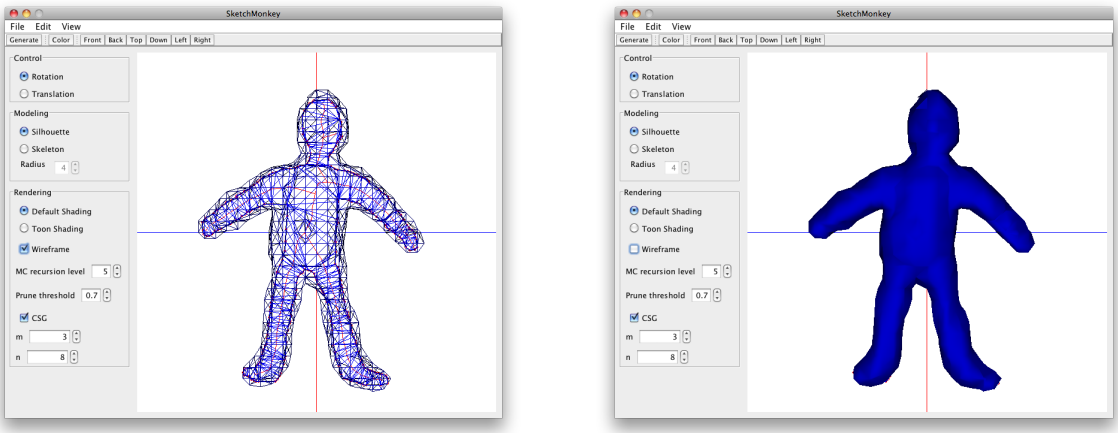

Figura 6.13: Modelo de silhueta humana. Modelo gerado usando $m = 3$ e $n = 8$.

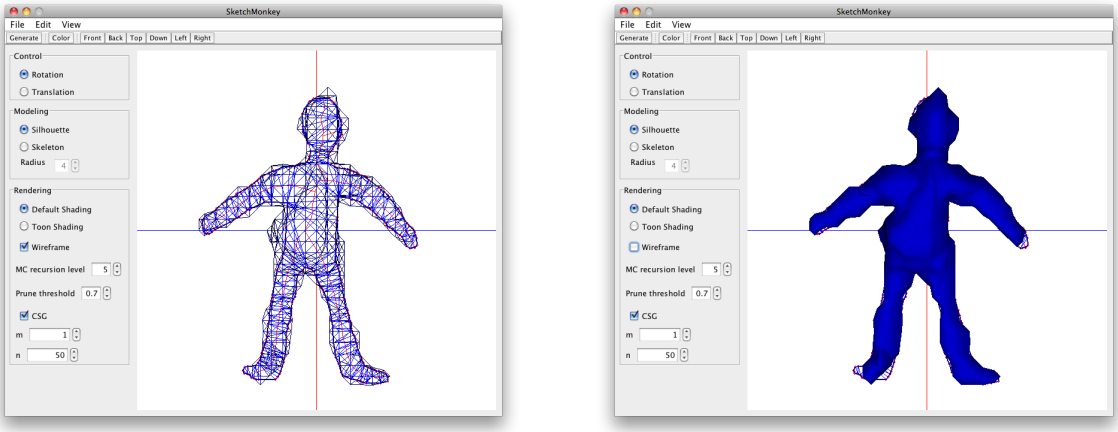

Figura 6.14: Modelo de silhueta humana. Modelo gerado usando $m = 1$ e $n = 50$.

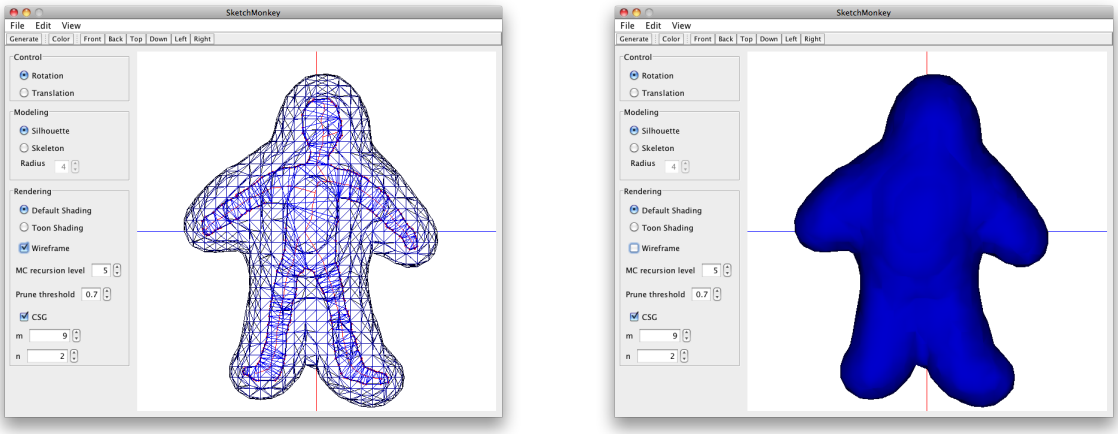

Figura 6.15: Modelo de silhueta humana. Modelo gerado usando $m = 9$ e $n = 2$.

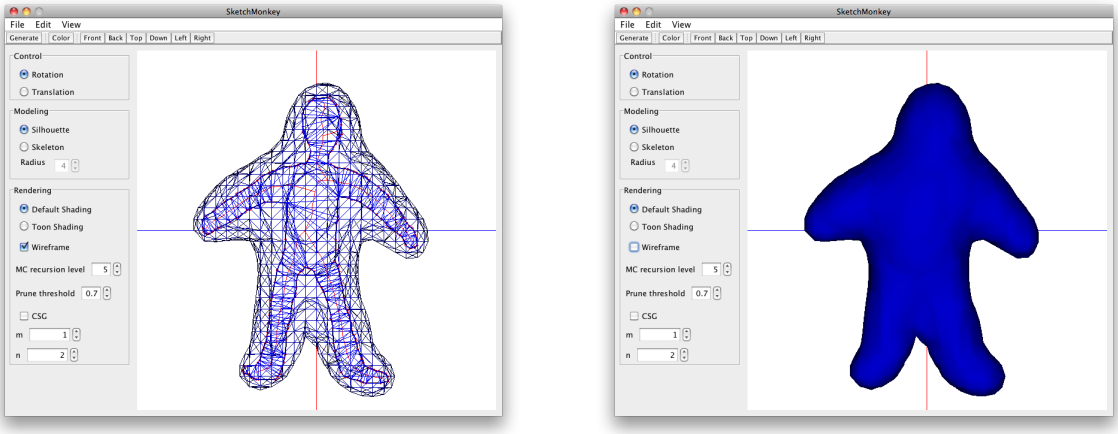

Figura 6.16: Modelo de silhueta humana. Modelo gerado usando somatório simples.

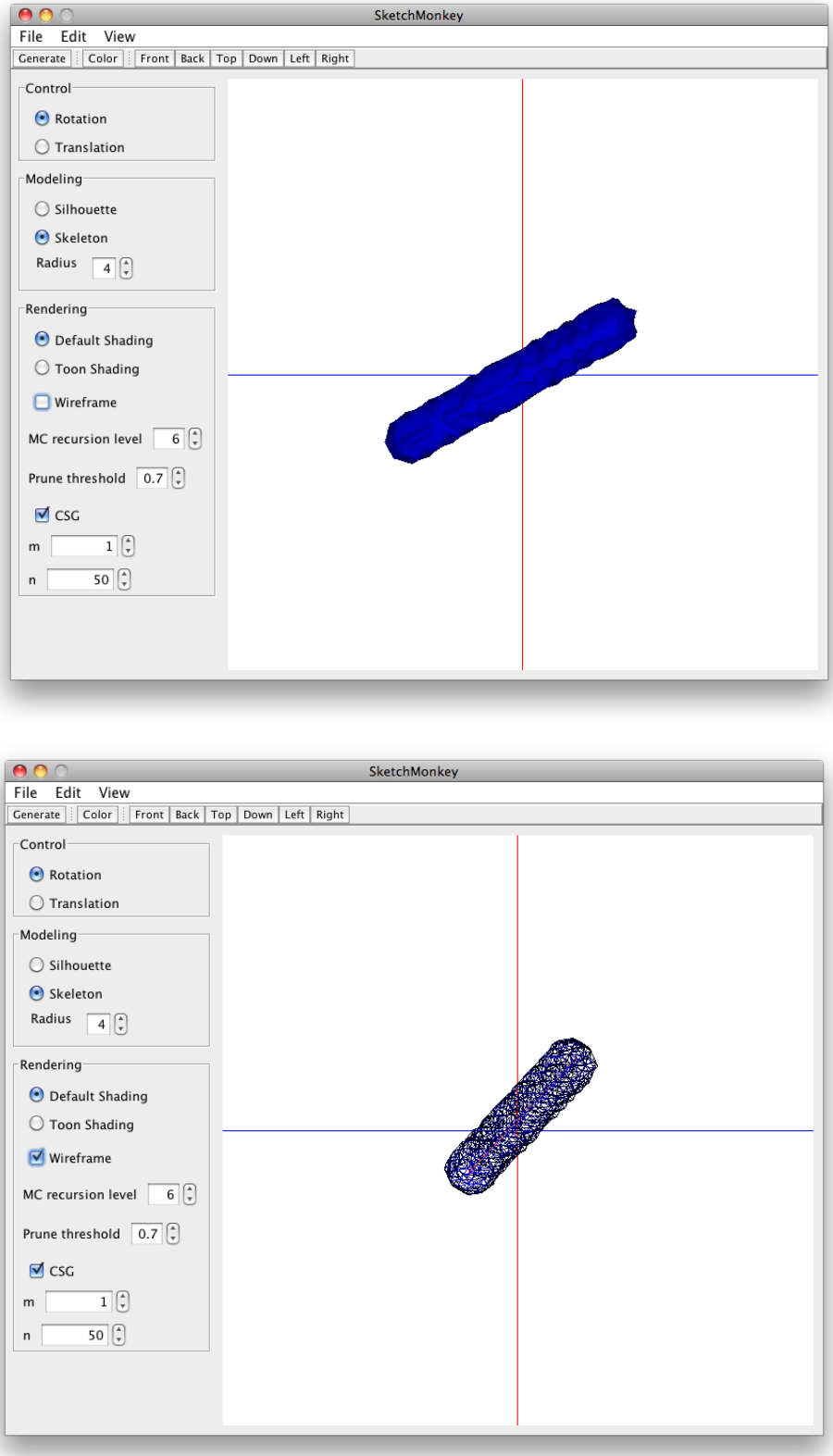

Figura 6.17: Modelo de segmento de reta. Modelo gerado usando $m = 1$ e $n = 50$.

6.2.4 Análise sobre o desempenho do sistema

O fator mais impactante no desempenho do sistema é o tempo do algoritmo de poligonização *Marching Cubes* (ver Subseção 5.5.6). A seguir ilustramos alguns exemplos de figuras geradas com níveis de recursão $n = 5$ (Figura 6.18) e $n = 6$ (Figura 6.19). É nítida a qualidade superior da malha $n = 6$ em relação à malha $n = 5$. Entretanto, o tempo de geração para $n = 6$ é significativamente mais alto que o de $n = 5$.

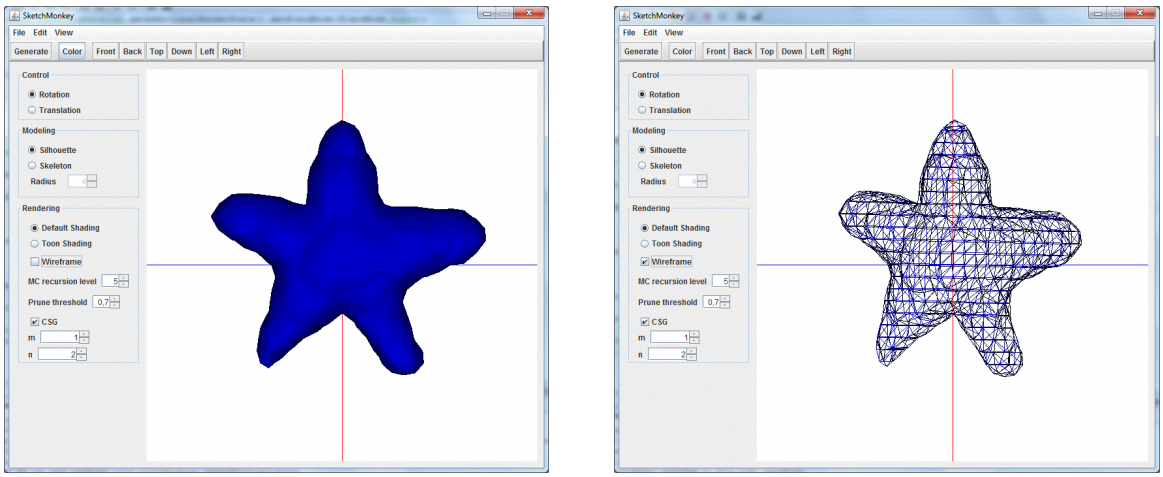

Figura 6.18: Modelo de estrela de cinco pontas gerado com nível de recursão $n = 5$. Tempo de geração: 17,6 ms

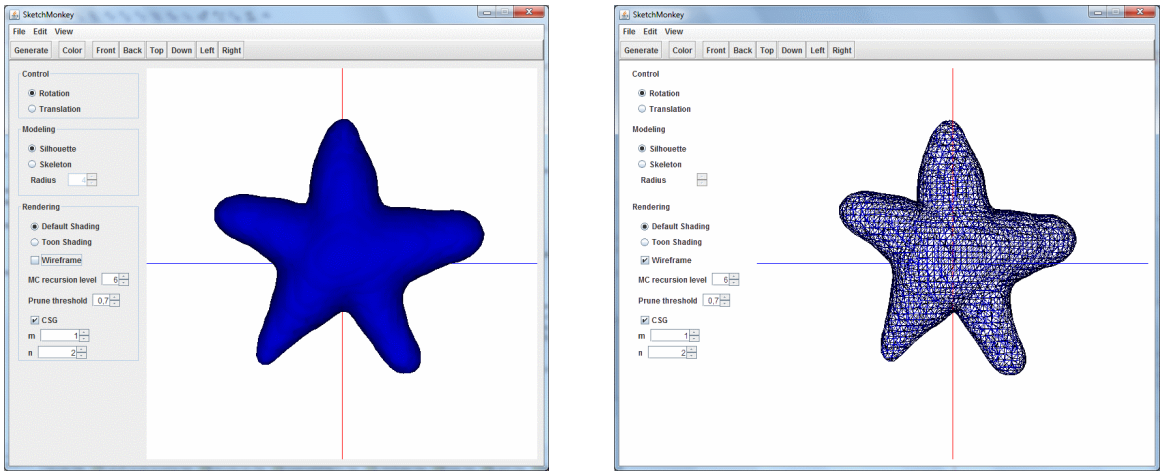

Figura 6.19: Modelo de estrela de cinco pontas gerado com nível de recursão $n = 6$. Tempo de geração: 112,6 ms

Com base nesses experimentos, podemos afirmar que o tempo gasto com $n = 5$ é razoável para um processo interativo de modelagem, e $n = 6$ é razoável para gerar uma malha mais detalhada, ao fim do processo.

6.2.5 Análise da representação de formas redondas

Nesse experimento é possível observar a imprecisão do método na representação de formas redondas. Ao desenhar formas aproximadamente redondas, isto é, formas predominantemente isotrópicas, o algoritmo de esqueletonização gera um esqueleto que não captura corretamente a forma do objeto, o que ocasiona na geração de formas distorcidas, em relação à silhueta original (ver Figura 6.20). Esse problema foi descrito por Levet [13] como sendo inerente à técnica de extração do esqueleto por CDT.

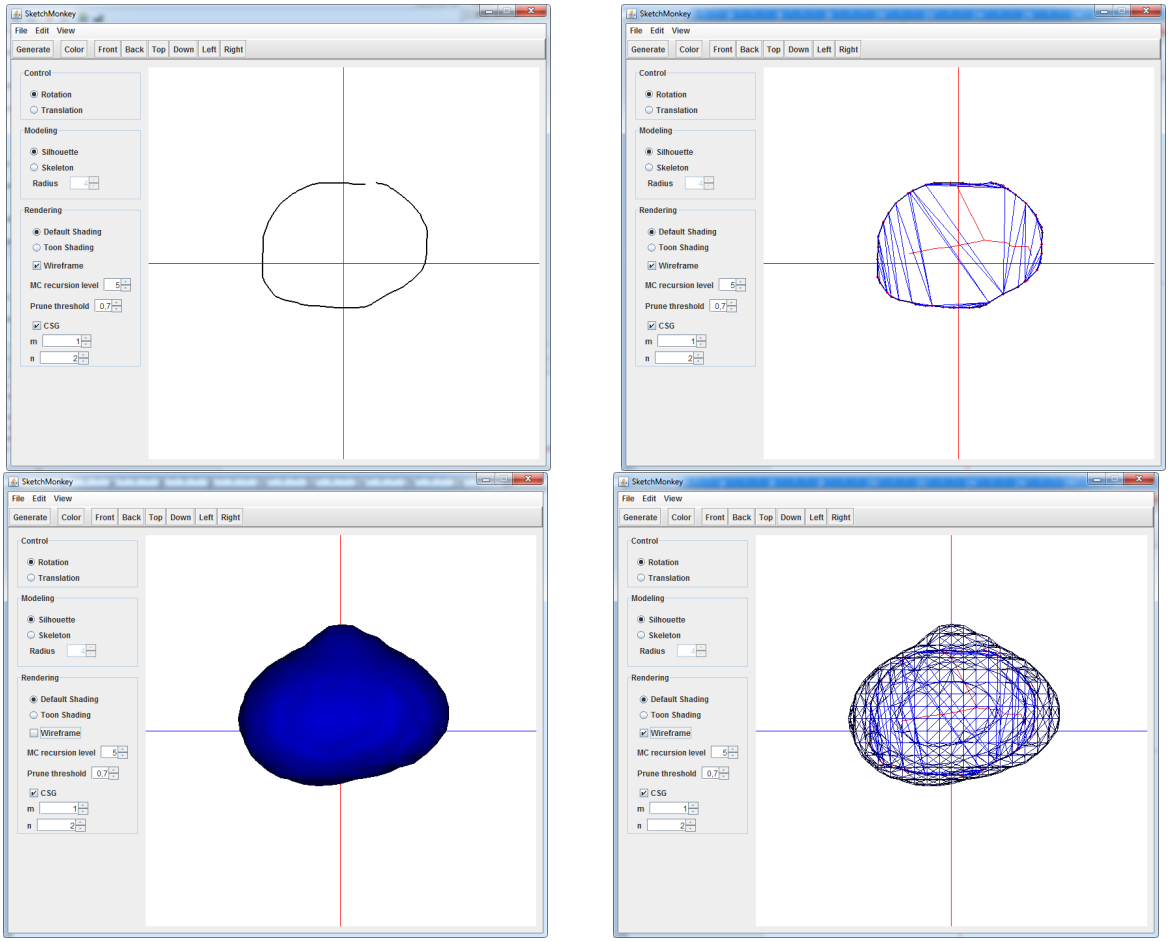

Figura 6.20: Imprecisão na representação de formas redondas. Da esquerda para a direita: Silhueta, esqueleto, modelo 3D, *wireframe* do modelo.

Capítulo 7

Conclusão e trabalhos futuros

Neste trabalho, nós apresentamos um sistema de *sketch-based modeling* usando uma superfície de convolução para representar o modelo 3D. O modelo de superfície de convolução é uma forma de representação intuitiva e flexível.

Diferentemente de métodos de resolução única que não usam representação implícita, nosso método, assim como qualquer representação implícita, é uma representação contínua e compacta. Usando um método para visualização de superfície implícita como o *Marching Cubes*, é possível obter qualquer nível de resolução desejado, o que permite usar malhas de baixa resolução nos estágios iniciais do processo de modelagem, e altas resoluções ao final do processo.

Uma primitiva de uma superfície de convolução pode ser qualquer tipo de função, mas por simplicidade e eficiência, nós escolhemos segmentos de reta como primitivas. Uma desvantagem de usar segmentos de reta como primitivas é que isso cria protuberâncias nas junções, mas nós resolvemos esse problema com a utilização de pesos atribuídos aos campos de contribuição, como em [27]. Outra forma de resolver esse problema seria incluir polígonos no esqueleto e aplicar certas restrições [4], mas derivar uma representação de forma fechada do modelo de convolução por pesos polinomiais seria difícil e possivelmente só obtida por meio de método numérico.

Outro problema com as superfícies de convolução é o método de composição de primitivas. Tipicamente, isso seria resolvido como uma simples soma dos campos de contribuição de todas as primitivas, mas esse método tem a grande desvantagem de que todas as primitivas influenciam o campo global, o que pode causar resultados inesperados. Para resolver este problema usamos a função de *blending* do modelo de composição CSG descrito em [1], mais flexível e preciso, que limita o raio de influência de cada primitiva a uma área

próxima a sua posição e também faz com a que a transição entre os campos seja suave.

Para solucionar o problema do ajuste da malha à silhuetas de formas redondas, devem ser investigados outros métodos de extração de esqueleto e remoção de ramos irrelevantes, como por exemplo os citados em [13], [17] e [28]. Outro problema a ser solucionado é o do tempo de poligonização, portanto também devem ser investigados métodos para reduzir o tempo gasto nesta etapa, como por exemplo o método descrito em [26]. Além disso, outros temas que devem ser abordados são: geração adaptativa de superfícies implícitas, animação por *sketch*, *morphing* de *sketches*, e uso de ruídos controlados por *sketches* para geração de malhas com relevos.

Referências

1 Alexe, A.; Barthe, L.; Cani, M.; Gaildrat, V. Shape modeling by sketching using convolution surfaces. In *ACM SIGGRAPH 2007 courses* (2007), ACM, p. 39.

2 Bajaj, C. *Introduction to implicit surfaces*. Morgan Kaufmann, 1997.

3 Bloomenthal, J. Polygonization of implicit surfaces. *Computer Aided Geometric Design 5*, 4 (1988), 341–355.

4 Bloomenthal, J. Bulge elimination in convolution surfaces. In *Computer Graphics Forum* (1997), vol. 16, Wiley Online Library, pp. 31–41.

5 Cherlin, J.; Samavati, F.; Sousa, M.; Jorge, J. Sketch-based modeling with few strokes. In *Proceedings of the 21st spring conference on Computer graphics* (2005), ACM, pp. 137–145.

6 Cordier, F.; Seo, H. Free-form sketching of self-occluding objects. *IEEE Computer Graphics and Applications* (2007), 50–59.

7 De Araujo, B.; Jorge, J. Blobmaker: Free-form modelling with variational implicit surfaces. In *Proceedings of 12 º Eurographics Portuguese Chapter (12 º Encontro Português de Computação Grafica)* (2003), vol. 12, pp. 17–26.

8 Domiter, V.; Žalik, B. Sweep-line algorithm for constrained delaunay triangulation. *International Journal of Geographical Information Science 22*, 4 (2008), 449–462.

9 Eggli, L.; Hsu, C.; Bruederlin, B.; Elber, G. Inferring 3d models from freehand sketches and constraints. *Computer-Aided Design 29*, 2 (1997), 101–112.

10 Igarashi, T.; Matsuoka, S.; Tanaka, H. Teddy: a sketching interface for 3d freeform design. In *ACM SIGGRAPH 2007 courses* (2007), ACM, p. 21.

11 JMonkeyEngine. Java opengl game engine. `http://www.jmonkeyengine.org`. [Online; accessado em 28/08/2012].

12 Kúkelová, Z.; Durikovic, R.; Paprzycki, M. Sketch-based modeling system with convolution and variational implicit surfaces.

13 Levet, F.; Granier, X. Improved skeleton extraction and surface generation for sketch-based modeling. In *Proceedings of Graphics Interface 2007* (2007), ACM, pp. 27–33.

14 Levoy, M. Efficient ray tracing of volume data. *ACM Transactions on Graphics (TOG) 9*, 3 (1990), 245–261.

15 Lorensen, W.; Cline, H. Marching cubes: A high resolution 3d surface construction algorithm. In *ACM Siggraph Computer Graphics* (1987), vol. 21, ACM, pp. 163–169.

16 Nealen, A.; Igarashi, T.; Sorkine, O.; Alexa, M. Fibermesh: designing freeform surfaces with 3d curves. In *ACM Transactions on Graphics (TOG)* (2007), vol. 26, ACM, p. 41.

17 Ogniewicz, R.; Kübler, O. Hierarchic voronoi skeletons. *Pattern recognition 28*, 3 (1995), 343–359.

18 Olsen, L.; Samavati, F.; Sousa, M.; Jorge, J. A taxonomy of modeling techniques using sketch-based interfaces. *Eurographics State of the Art Reports* (2008).

19 OpenCTM. Open compressed triangle mesh file format. `http://openctm.sourceforge.net/`. [Online; accessado em 28/08/2012].

20 Parari, A. Modelagem e visualização de objetos à mão livre usando superfícies implícitas variacionais. Dissertação de Mestrado, UNIVERSIDADE FEDERAL DO RIO DE JANEIRO, 2004.

21 Rose, K.; Sheffer, A.; Wither, J.; Cani, M.; Thibert, B. Developable surfaces from arbitrary sketched boundaries. In *Proceedings of the fifth Eurographics symposium on Geometry processing* (2007), Eurographics Association, pp. 163–172.

22 Schmidt, R.; Wyvill, B.; Sousa, M.; Jorge, J. Shapeshop: Sketch-based solid modeling with blobtrees. In *ACM SIGGRAPH 2006 Courses* (2006), ACM, p. 14.

23 Schweikardt, E.; Gross, M. Digital clay: deriving digital models from freehand sketches. *Automation in Construction 9*, 1 (2000), 107–115.

24 Sherstyuk, A. *Convolution surfaces in computer graphics.* Tese de Doutorado, Monash University, Australia, 1999.

25 Shewchuk, J. Triangle: Engineering a 2d quality mesh generator and delaunay triangulator. *Applied Computational Geometry Towards Geometric Engineering* (1996), 203–222.

26 Singh, J.; Narayanan, P. Real-time ray tracing of implicit surfaces on the gpu. *Visualization and Computer Graphics, IEEE Transactions on 16*, 2 (2010), 261–272.

27 Tai, C.; Zhang, H.; Fong, J. Prototype modeling from sketched silhouettes based on convolution surfaces. In *Computer Graphics Forum* (2004), vol. 23, Wiley Online Library, pp. 71–83.

28 Thibault, D.; Gold, C. Terrain reconstruction from contours by skeleton construction. *GeoInformatica 4*, 4 (2000), 349–373.

29 Wang, H.; Markosian, L. Free-form sketch. In *Proceedings of the 4th Eurographics workshop on Sketch-based interfaces and modeling* (2007), ACM, pp. 53–58.

30 Williams, L.; Hanson, A. Perceptual completion of occluded surfaces. In *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR'94., 1994 IEEE Computer Society Conference on* (1994), IEEE, pp. 104–112.

31 Yang, C. *Sketch-based modeling of parameterized objects*. Tese de Doutorado, The University of British Columbia, 2006.

32 Zeleznik, R.; Herndon, K.; Hughes, J. Sketch: An interface for sketching 3d scenes. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques* (1996), ACM, pp. 163–170.