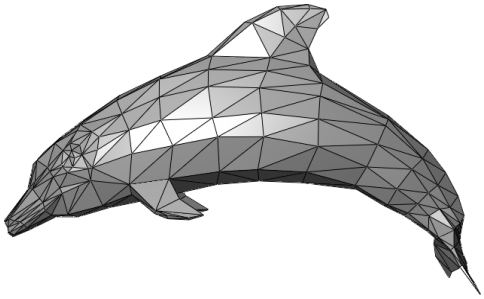
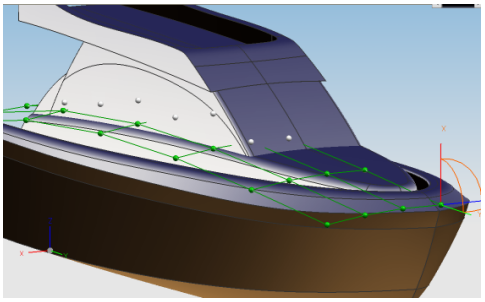
1
Introdução

A representação de formas é um problema fundamental em diversas áreas da ciência da computação, como por exemplo, as áreas de computação gráfica, visão computacional, física computacional e visualização. O principal foco é a representação de objetos tridimensionais, com o objetivo de reproduzir ou simular, em uma configuração discreta digital, os objetos encontrados no mundo real.

As características exigidas da representação de um objeto 3-D variam muito de acordo com a aplicação. Em termos gerais, uma representação tem por objetivo capturar a aparência, a geometria e/ou as propriedades físicas de um objeto. Neste escopo, a comunidade científica produziu inúmeras representações, e o aperfeiçoamento e criação de novos métodos ainda é uma intensa linha de pesquisa.



1.1(a): Malha de triângulos aproximando a forma de um golfinho.

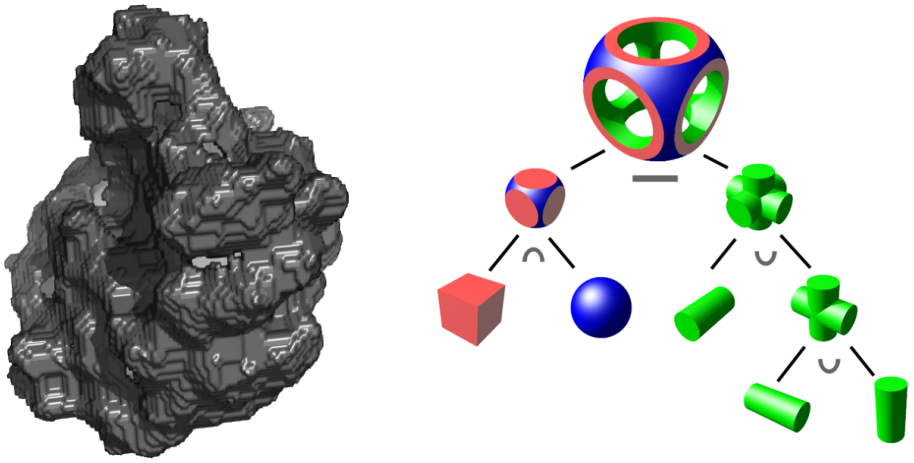


1.1(b): Barco modelado com superfícies paramétricas (neste caso, NURBS).

Figura 1.1: Objetos representados por superfícies poligonais e paramétricas. (Ambas as imagens são de domínio público e foram extraídas da Wikipédia.)

O conceito de aparência está relacionado principalmente com o que é visível do exterior de um objeto. Por esta razão, é comum em computação gráfica que os objetos sejam representados por meio de uma ou mais superfícies, sem necessariamente caracterizar um *manifold* fechado. A representação mais comum para superfícies, por sua vez, são as malhas poligonais — muito populares na indústria de entretenimento, especialmente no formato de malhas de triângulos (Figura 1.1(a)). Em casos onde as curvaturas precisam ser definidas de forma extremamente suave, os objetos podem ser representados

por retalhos de superfícies paramétricas (Figura 1.1(b)). Neste esquema, superfícies são aproximadas ou interpoladas a partir de um conjunto de pontos, e podem garantir, opcionalmente, continuidade C^1 (ou superior). As superfícies paramétricas são populares em pacotes de desenho industrial (*e.g.* CAD – *computer-aided design*), por serem fáceis de manipular.



1.2(a): Sólido definido por uma grade de voxels binários.

1.2(b): Árvore de operações booleanas para formar um sólido através de CSG.

Figura 1.2: Sólidos representados por grades de voxels e CSG. (Imagens extraídas da Wikipédia.)

Muitas aplicações exigem um modelo geométrico mais robusto para seus objetos 3-D. Nestes casos, os sistemas costumam trabalhar com o conceito de *sólidos*. Um objeto sólido 3-D pode ser descrito matematicamente como o conjunto de pontos delimitado por um 2-*manifold* fechado e orientável. É possível identificar três macro abordagens para a representação de sólidos:

Representação de fronteira: o sólido é descrito por um conjunto de superfícies que delimitam o espaço interior e exterior do objeto. A parte interior é "preenchida" implicitamente para formar um sólido.

Representação volumétrica discreta: o espaço onde se encontra um sólido é subdividido em uma grade tridimensional regular. O sólido, então, é especificado pelo conjunto das células que o interceptam (Figura 1.2(a)). O espaço também pode ser decomposto em uma grade não-regular, de forma que se adapte melhor à geometria do objeto.

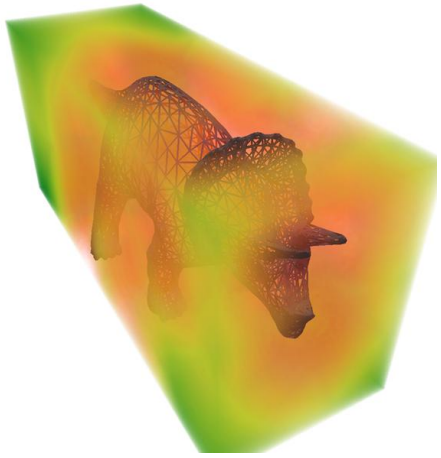
Representação implícita: o sólido é definido implicitamente por uma função $f : \mathbb{R}^3 \rightarrow \mathbb{R}$. Os pontos que satisfazem $f(\mathbf{p}) \leq 0$ pertencem ao sólido, e os pontos onde $f(\mathbf{p}) > 0$ estão fora do sólido. A geometria construtiva de sólidos (CSG – *constructive solid geometry*) é um exemplo de representação implícita. Objetos complexos são compostos através

de operações booleanas (Figura 1.2(b)) com base nas funções de sólidos primitivos (cubóides, cilindros, prismas, pirâmides, esferas e cones).

A representação de sólidos pode facilitar diversos tipos de processamento, como a aplicação de operações booleanas [11], a dilatação ou erosão de objetos [30], e a extração de esqueletos [5].

As representações volumétricas [19] são as mais adequadas quando, além da geometria, é necessário representar propriedades físicas associadas ao interior dos objetos. A técnica mais comum consiste em amostrar campos escalares ou vetoriais em espaços regulares, e então armazená-los em grades 3-D. Mais tarde, os campos podem ser reconstruídos utilizando, por exemplo, interpolação trilinear. O uso de volumes regulares simplifica a representação de objetos deformáveis e a aplicação de métodos numéricos — como por exemplo, o cálculo de derivadas por diferenças finitas. Dados volumétricos podem ser obtidos por amostragem de fenômenos ou objetos reais, produzidos por simulação computacional (*e.g.* aplicações envolvendo análise numérica), ou gerados a partir de modelos geométricos. Exemplos de aplicações que trabalham com amostras volumétricas são os visualizadores médicos (*e.g.* tomografia computadorizada e ultra-sonografia) e geocientíficos (*e.g.* amostras sísmicas). Há também aplicações que geram dados volumétricos a partir de simulações em supercomputadores, como por exemplo, programas para previsão meteorológica, simulação química e análise hidro ou aerodinâmica por CFD (*computational fluid dynamics*).

1.3(a): Campo de distância do robô Hugo, amostrado em uma grade de $73 \times 45 \times 128$.



1.3(b): Campo de distância de um triceratope, amostrado em uma grade de $128 \times 56 \times 42$.

Figura 1.3: Visualização volumétrica de campos de distância 3-D. Imagens provenientes do trabalho [31].

Os *campos de distância* são provavelmente a representação volumétrica mais apropriada para a geometria de objetos [17] — e no âmbito geral, são

uma das representações geométricas mais expressivas e versáteis, com grande número de aplicações [17]. Ao contrário dos volumes binários (como mostrado na Figura 1.2(a)), volumes de distância não sofrem tanto com *aliasing*, e oferecem uma série de vantagens. É possível extrair múltiplas iso-superfícies, testar se objetos estão a uma certa distância um do outro, e até mesmo transmutar a geometria de objetos, através de técnicas simples. A Figura 1.3 mostra volumes de distância calculados a partir de modelos poligonais. As cores indicam a distância até a superfície do modelo, e crescem do vermelho para o verde.

A principal limitação dos campos de distância amostrados regularmente está no alto consumo de memória. Um volume de 512^3 com voxels de 32 bits requer 512MB de memória, e oferece uma resolução de apenas $1 : 2^{-9}$ (ou seja, aproximadamente $1 : 0.004$). Além disso, a amostragem em espaços regulares gera muito desperdício quando a geometria possui características em escalas muito distintas. Por exemplo, o modelo de uma porta simples pode ser representado facilmente em um volume de 32^3; mas se a porta tiver maçaneta, fechadura e chave, seria preciso um volume muito maior — por exemplo, 256^3 — para representar detalhes que ocupam apenas uma pequena região da porta. Neste exemplo, o ideal seria que a parte plana da porta fosse representada em baixa resolução, e apenas as regiões da maçaneta e fechadura fossem representadas em maior resolução, à medida que necessário.

Com o intuito de superar a maior parte das limitações atribuídas aos campos de distância regulares, Frisken *et al.* [10] propuseram os *campos de distância amostrados adaptativamente* (ADFs – *adaptively sampled distance fields*). O uso de amostragem adaptativa significa que regiões com detalhes refinados podem ter freqüências de amostragem mais altas, enquanto regiões onde o campo de distância varia suavemente usam taxas de amostragem mais baixas. Este esquema, além de utilizar memória de forma mais eficiente, permite obter precisão arbitrária no campo reconstruído.

ADFs podem ser implementadas por meio de diferentes técnicas. Uma das técnicas mais simples, apresentada por Frisken *et al.* no primeiro artigo sobre ADFs [10], consiste em armazenar amostras de distância nos vértices das células de uma *octree*, e utilizar interpolação trilinear para reconstruir o campo de distância em uma célula. Com esta técnica, as células de nível 12, por exemplo, ofereceriam uma resolução de $1 : 2^{-12}$ (aproximadamente $1m : 240\mu m$). Para obter-se a mesma precisão em um volume regular, seria necessário representar 69 bilhões de voxels. Outras implementações de ADFs também podem ser obtidas trocando-se a *octree* por outra estrutura espacial, ou a interpolação trilinear por outro método de reconstrução mais sofisticado,

como sugerido no Capítulo 6.

Por utilizar estruturas mais complexas, o custo de reconstrução das ADFs é naturalmente maior do que o de campos de distância regulares. Isto pode ser um problema para determinadas aplicações, especialmente as que precisam extrair iso-superfícies, planejar caminhos ou detectar colisões com ADFs em tempo real (*e.g.* aplicações interativas). Nesses casos, o custo de reconstrução de uma ADF pode ser alto demais para atender ao número de consultas por segundo exigido pela aplicação. A solução freqüentemente adotada na indústria e na academia é o uso de processadores paralelos, *clusters* — ou mais recentemente [22], placas gráficas programáveis (GPUs).

O objetivo deste trabalho é investigar técnicas que aumentem a eficiência e acelerem a reconstrução de ADFs para aplicações interativas de computação gráfica. Uma das principais metas é investigar estruturas e algoritmos que permitam representar e reconstruir ADFs eficientemente em GPUs. Além disso, buscam-se técnicas para renderizar ADFs com iluminação suave, e melhorar a reconstrução dos campos de distância para que aproximem melhor superfícies curvas. Como prova de conceito, o resultado desejado é uma aplicação interativa para a visualização de iso-superfícies de ADFs.

O escopo do trabalho restringe-se às ADFs *estáticas* — ou seja, aquelas cujo campo de distância permanece constante após a construção. Apesar da representação de geometrias deformáveis ser uma das principais aplicações para os campos de distância regulares, o uso de ADFs com este fim é fortemente limitado pelo custo e a complexidade de se reconstruir parcialmente a estrutura espacial da ADF. Portanto, este trabalho foca na representação eficiente de ADFs estáticas, que servem como representações caixa-preta para funções de distância complexas.

As principais contribuições desta pesquisa são:

i) Um método para representar e reconstruir ADFs estáticas de forma eficiente em GPUs modernas.

ii) Um método, baseado no artigo de Lefebvre e Hoppe [20], para construir funções de dispersão espacial perfeita (*perfect spatial hashing*) próprias para GPUs, para conjuntos de dados muito esparsos.

iii) Uma técnica simples e eficiente, baseada em traçado por esferas (*sphere tracing*), para renderizar ADFs em GPU.

iv) Uma maneira eficiente de superar as descontinuidades C^1 presentes nas ADFs e renderizar iso-superfícies com sombreamento suave.

v) Um novo método de reconstrução para ADFs, denominado *pseudo-tricúbico*, capaz de aproximar melhor o valor do campo de distância em regiões com superfícies curvas.

Esta dissertação está organizada em 7 capítulos. O próximo capítulo aborda trabalhos relacionados e apresenta, de forma sucinta, os conceitos mais importantes sobre campos de distância, ADFs e suas técnicas de renderização. O Capítulo 3 descreve o método proposto neste trabalho para representar e reconstruir ADFs estáticas em placas gráficas modernas. No Capítulo 4, são apresentadas técnicas para renderizar ADFs em GPU, com sombreamento suave. Alguns exemplos de aplicações, e os principais resultados deste trabalho, são mostrados no Capítulo 5. O Capítulo 6 aborda a reconstrução *pseudo-tricúbica*: a primeira seção explica a motivação para a criação deste novo método, a segunda seção explica a implementação do método, e a terceira seção analisa os resultados. Finalmente, o Capítulo 7 conclui a pesquisa e aponta trabalhos futuros.