UNIVERSIDADE FEDERAL DO CEARÁ
CENTRO DE CIÊNCIAS
DEPARTAMENTO DE COMPUTAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

RAFAEL FERNANDES IVO

ADAPTAÇÃO DE MODELOS BASEADOS EM SPLATS PARA SUPERFÍCIES E ARESTAS CURVAS

FORTALEZA

2020

RAFAEL FERNANDES IVO

ADAPTAÇÃO DE MODELOS BASEADOS EM SPLATS PARA SUPERFÍCIES E ARESTAS CURVAS

Tese apresentada ao Programa de Pós-Graduação em Ciência da Computação do Centro de Ciências da Universidade Federal do Ceará, como requisito parcial à obtenção do título de doutor em Ciência da Computação. Área de Concentração: Computação Gráfica

Orientador: Prof. Creto Augusto Vidal, Ph.D.

Coorientador: Prof. Joaquim Bento Cavalcante-Neto, Dr.

FORTALEZA

2020

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Biblioteca Universitária
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

I1a Ivo, Rafael Fernandes.
Adaptação de modelos baseados em splats para superfícies e arestas curvas / Rafael Fernandes Ivo. – 2020.
97 f. : il. color.

Tese (doutorado) – Universidade Federal do Ceará, Centro de Ciências, Programa de Pós-Graduação em Ciência da Computação , Fortaleza, 2020.
Orientação: Prof. Dr. Creto Augusto Vidal.
Coorientação: Prof. Dr. Joaquim Bento Cavalcante-Neto.

1. Computação gráfica. 2. Renderização. 3. Surface splatting. 4. Arestas e cantos. 5. Silhuetas. I. Título.
CDD 005

RAFAEL FERNANDES IVO

ADAPTAÇÃO DE MODELOS BASEADOS EM SPLATS PARA SUPERFÍCIES E ARESTAS CURVAS

Tese apresentada ao Programa de Pós-Graduação em Ciência da Computação do Centro de Ciências da Universidade Federal do Ceará, como requisito parcial à obtenção do título de doutor em Ciência da Computação. Área de Concentração: Computação Gráfica

Aprovada em: 07/02/2020.

BANCA EXAMINADORA

Prof. Creto Augusto Vidal, Ph.D. (Orientador)
Universidade Federal do Ceará (UFC)

Prof. Joaquim Bento Cavalcante-Neto, Dr. (Coorientador)
Universidade Federal do Ceará (UFC)

Profa. Emanuele Marques dos Santos, Ph.D.
Universidade Federal do Ceará (UFC)

Profa. Maria Andréia Formico Rodrigues, Ph.D.
Universidade de Fortaleza (UNIFOR)

Prof. Anselmo Cardoso de Paiva, Dr.
Universidade Federal do Maranhão (UFMA)

Dedico esta tese aos meus pais, irmãos, minha esposa e a mim mesmo.

AGRADECIMENTOS

Em primeiro lugar, agradeço a Deus por ter me dado forças para superar todas as dificuldades que ocorreram neste longo percurso do doutorado e me permitir concluí-lo.

A minha esposa, Ariadne, que não me deixou desistir, que me deu força em todos os momentos, nos bons e ruins. Agradeço a ela pela compreensão da minha ausência quando precisei me dedicar a esse trabalho.

Aos meus pais, pelo sacrifício que tiveram que fazer por tantos anos para dar a melhor educação que eu pude ter e pelo apoio nas minhas decisões. Devo minha vida e tudo que alcancei a eles.

Aos meus irmãos, Rafaele e Roberto, que sei que torcem muito por mim e aos quais sirvo de exemplo como irmão mais velho.

Aos meus orientadores, em especial, que participaram de minha carreira acadêmica desde o principio da graduação até hoje. Ao meu coorientador Joaquim Bento pela ajuda na pesquisa, na formulação de escrita de documentos acadêmicos e até mesmo na vida. E ao meu orientador Creto Vidal, sem o qual eu certamente não teria finalizado esta tese. Agradeço a sua paciência e dedicação a ajudar minha pessoa a concluir este doutorado. Vocês são minha inspiração para minha carreira como professor universitário.

A todos os colegas e amigos do grupo de pesquisa VCG (Visual Computing Lab) em ISTI-CNR-Pisa. Especialmente aos pesquisadores Roberto Scopigno e Fabio Ganovelli por toda a estrutura e apoio na evolução da minha pesquisa, e a Francesco Banterle, Marco Potenziani, Luca Benedetti e Francesca De Mitry por terem tornado minha estadia na Italia tão especial e maravilhosa.

Aos meus colegas da Universidade Federal do Ceará do Campus de Russas, que auxiliaram com bons horários de disciplinas, me permitindo ter tempo de trabalho para o doutorado.

Aos meus amigos do CRAb, que me fizeram ter todo o crescimento que tive na área de computação gráfica até hoje com conhecimento e experiências.

Aos meus amigos do Projeto Seis de Março, aos quais devo a descoberta da minha vocação para professor universitário e, dessa forma, na minha continuação na carreira acadêmica.

Ao apoio financeiro da Coordenação de Pessoal de Nível Superior (CAPES).

A todos, meu muito obrigado.

"O sonho é que leva a gente para frente. Se a gente for seguir a razão, fica aquietado, acomodado."

(Ariano Suassuna)

RESUMO

A técnica de *surface splatting* é comprovadamente uma boa abordagem para renderizar modelos baseados em pontos. Os artefatos como aliasing são efetivamente evitados por filtros no espaço de objeto e de imagem. Renderizações com alto desempenho são obtidas pelas implementações em GPU e sombreamentos de alta qualidade, com qualidade similar a *Phong*, pelo uso de campos normais. No entanto, splats são discos no espaço do objeto e não podem representar adequadamente arestas e cantos de um modelo. Além disso, quando o modelo é observado em uma visualização aproximada ou quando o modelo é de baixa amostragem, artefatos tornam-se visíveis, como saliências na silhueta. Apresentamos um novo método para recortar splats em modelos com arestas curvas e cantos. Cada splat próximo de uma aresta é uma elipse recortada contra uma curva de Bézier racional bidimensional. Foi desenvolvido e implementado um cálculo automático das curvas de recorte e uma melhoria nos métodos de amostragem próximos a arestas. Esta tese propõe também o uso de *quadric splats*, uma primitiva de renderização que melhora a renderização de silhuetas e arestas de modelos amostrados por splats, e um algoritmo para calcular o conjunto de amostras presente nas silhuetas de uma nuvem de quadric splats. Os métodos anteriores extraem silhuetas de uma nuvem de pontos usando limiarização de vetores normais, o que pode levar a sobre e sub-detecção simultânea de amostras próximas das silhuetas. Outros métodos usam informações de vizinhança e/ou variacionais para essa finalidade. O quadric splat e a curva de recorte são definidos por dados simples e uniformes, portanto, nossas primitivas de renderização podem ser implementados completamente com base em vertex e fragment shaders das GPUs atuais.

Palavras-chave: Computação gráfica. Renderização. Surface splatting. Arestas e cantos. Silhuetas.

ABSTRACT

Surface splatting has been proven to be a good approach to render point-based models. Aliasing artifacts are effectively avoided by filtering. Renderings with high performance are achieved by GPU implementations and high quality shadings by normal fields. However, splats are disks in object space and cannot represent properly sharp features, as edges and corners. Besides, when the model is observed in a close-up view or when the model is low sampled, geometric artifacts become visible, such as bumps on silhouette. We present a new method for clipping splats in models with sharp features. Each splat near an edge is an ellipse clipped against a bidimensional rational Bézier curve. We designed and implemented an automatic computation of the clipping curves and an improvement on sampling methods near sharp features. We also present *quadric splat*, a rendering primitive that improves the rendering of model silhouettes and sharp features, and an algorithm to compute the silhouette set of a quadric splats cloud. Previous methods extract point set silhouettes by thresholding point normal, which can lead to simultaneous over and under-detection of silhouettes. Other methods use neighboring and variational info for this purpose. The quadric splat and the clipping curve are defined by simple and uniform data, thus, our rendering primitives can be implemented completely based on vertex and fragment shaders of current GPUs.

Keywords: Computer graphics. Rendering. Surface splatting. Sharp features. Silhouette.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

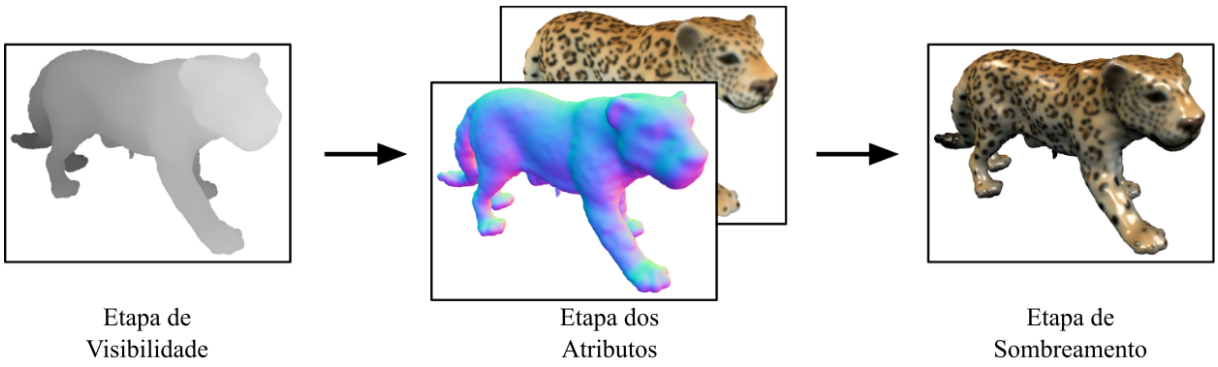
1 INTRODUÇÃO

Muitas aplicações gráficas representam objetos através de suas superfícies, cujos modelos usam um conjunto discreto de primitivas simples. Atualmente as primitivas mais comuns são polígonos, especialmente triângulos. Entretanto, nas últimas décadas, um novo tipo de primitiva tem sido bastante utilizada: o ponto. A alternativa de representar superfícies por meio de nuvens de pontos tem recebido crescente atenção por duas principais razões: a flexibilidade do processamento geométrico e a popularização de sistemas de escaneamento e fotografias digitais 3D (GROSS; PFISTER, 2007). De fato, em algumas aplicações, as nuvens de pontos são livres das sobrecargas de gerenciamento, processamento e manipulação de informações de conectividade comuns em malhas poligonais muito grandes. Por sua vez, os scanners a laser produzem modelos detalhados através de nuvens de pontos densas que precisam ser renderizadas. Um grande exemplo desse tipo de aplicação foi desenvolvido há cerca de duas décadas no Projeto Michelângelo Digital (LEVOY *et al.*, 2000). Hoje em dia, o problema ainda é relevante para a comunidade científica uma vez que tecnologias como LIDAR (*LIght Detection And Ranging*), sensores do Microsoft Kinect, câmeras iPhone TrueDepth, e Google Tango produzem nuvens de pontos para diversos tipos de aplicações (AKENINE-MöLLER *et al.*, 2018).

Pontos não armazenam qualquer informação de conectividade e topologia. Essa característica permite a utilização de nuvens de pontos na realização de operações geométricas complexas como: CSG (*Constructive Solid Geometry*) (ADAMS; DUTRÉ, 2003), renderização progressiva (WU *et al.*, 2005), reamostragem (GUENNEBAUD *et al.*, 2008), simulação física de fraturas (PAULY *et al.*, 2005) etc. Entretanto, a renderização direta de uma representação baseada em pontos, normalmente, resulta em espaços visíveis entre as amostras. Para preencher esses espaços, geralmente, trocam-se os pontos por discos ou elipses denominados *splats*. Assim como outras representações lineares de superfícies, tais como malhas triangulares, esses discos fornecem uma aproximação de primeira ordem da superfície, pois eles têm orientação e extensão. Dessa forma, usam-se splats maiores e em menor quantidade nas regiões mais planas, e splats menores e em maior quantidade nas regiões mais curvas. Apesar da similaridade com malhas triangulares, a nuvem de splats não define uma superfície contínua e conectada, mas mantém a simplicidade conceitual das representações baseadas em pontos.

O fato dos splats serem unidades autônomas tem o inconveniente de não impedir que eles se cruzem e gerem sobreposições, o que não ocorre com os elementos de uma malha. Por esse motivo, a técnica de *surface splatting* (ZWICKER *et al.*, 2001) combina splats vizinhos

Figura 1 – Pipeline de splatting em GPU. A etapa de visibilidade preenche o depth map, o qual permite que alguns fragmentos tenham seus atributos, como cor e normais, combinados e depois armazenados em buffers separados. A etapa de sombreamento recebe todos os buffers anteriores para calcular a cor final de cada pixel da imagem.

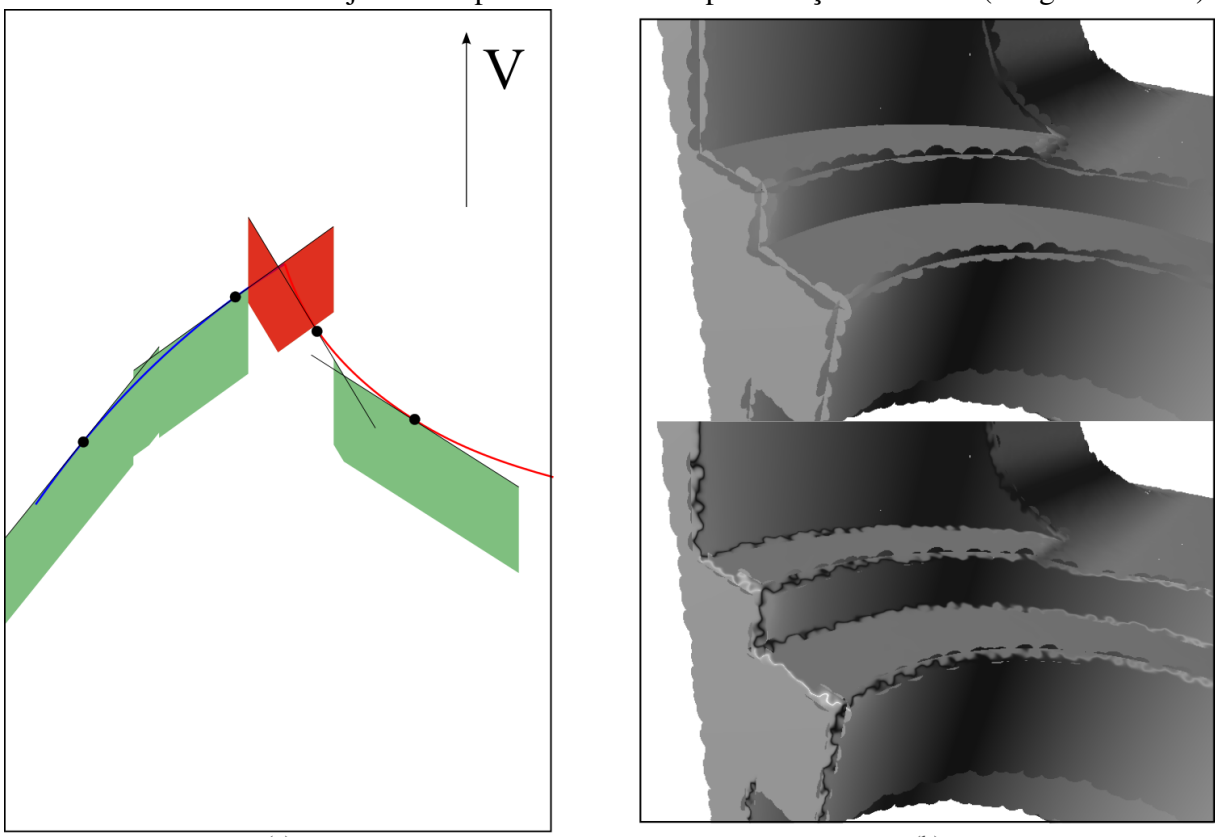


Fonte: (BOTSCH *et al.*, 2005)

e sobrepostos em espaço de imagem usando filtros passa-baixa, o que gera imagens de alta qualidade. No entanto, a implementação do algoritmo original de surface splatting é baseada em CPU, o que torna a renderização de modelos massivos, como é o caso da maioria dos modelos baseados em splats, extremamente lenta. Em termos práticos, essa é uma limitação muito severa e inaceitável na maioria dos casos. Devido à falta de suporte nativo a splats em hardware e em bibliotecas gráficas atuais, explorar a aceleração oferecida por GPUs não é uma tarefa simples. A abordagem mais utilizada atualmente foi proposta há quinze anos em Botsch *et al.* (2005). A renderização se dá em três etapas: duas etapas no espaço de objeto (etapa de visibilidade e etapa de atributos) e uma etapa no espaço de imagem (etapa de sombreamento) (Figura 1). Na ***etapa de visibilidade***, todos os splats são rasterizados sobre o frame buffer com o z-buffer habilitado. O propósito dessa etapa é obter os valores de profundidade dos splats visíveis em cada pixel, criando um *depth map*. Na ***etapa de atributos***, todos os splats são rasterizados novamente, mas com z-buffer desabilitado e a combinação aditiva de fragmentos habilitada. O propósito dessa etapa é obter os vetores normais e os valores de cores dos splats visíveis, combiná-los e salvá-los em dois diferentes buffers, o *normal map* e o *color map*. O *depth map* da etapa anterior é usado para evitar misturar fragmentos de splats que estão distantes do splat mais próximo do observador em uma dada direção. Na ***etapa de sombreamento***, um retângulo do tamanho da imagem é enviado para a GPU com os buffers gerados nas etapas anteriores. Com essas informações, computam-se sombreamentos por pixel no *fragment shader*.

Pesquisas em percepção visual demonstram que a renderização apropriada das características curvas de um modelo, especialmente em suas arestas e silhuetas, causam uma melhora significativa da visualização e da percepção da forma que está sendo representada

Figura 2 – Artefatos presentes nas arestas de modelos representados por splats. (a) Vetor **V** indica a direção do observador. A área verde atrás dos fragmentos mais próximos do observador compreende todos os fragmentos que serão combinados para a computação da cor de um determinado pixel. A área vermelha destaca uma região onde combinar ou não os fragmentos acarreta em artefatos. (b) Não combinar os fragmentos acarreta a detecção de discos cruzando uns aos outros (imagem superior). Combinar acarreta a mistura indesejada de superfícies e a má representação da aresta (imagem inferior).



(a) (b)

Fonte: o autor.

(KOENDERINK, 1984).

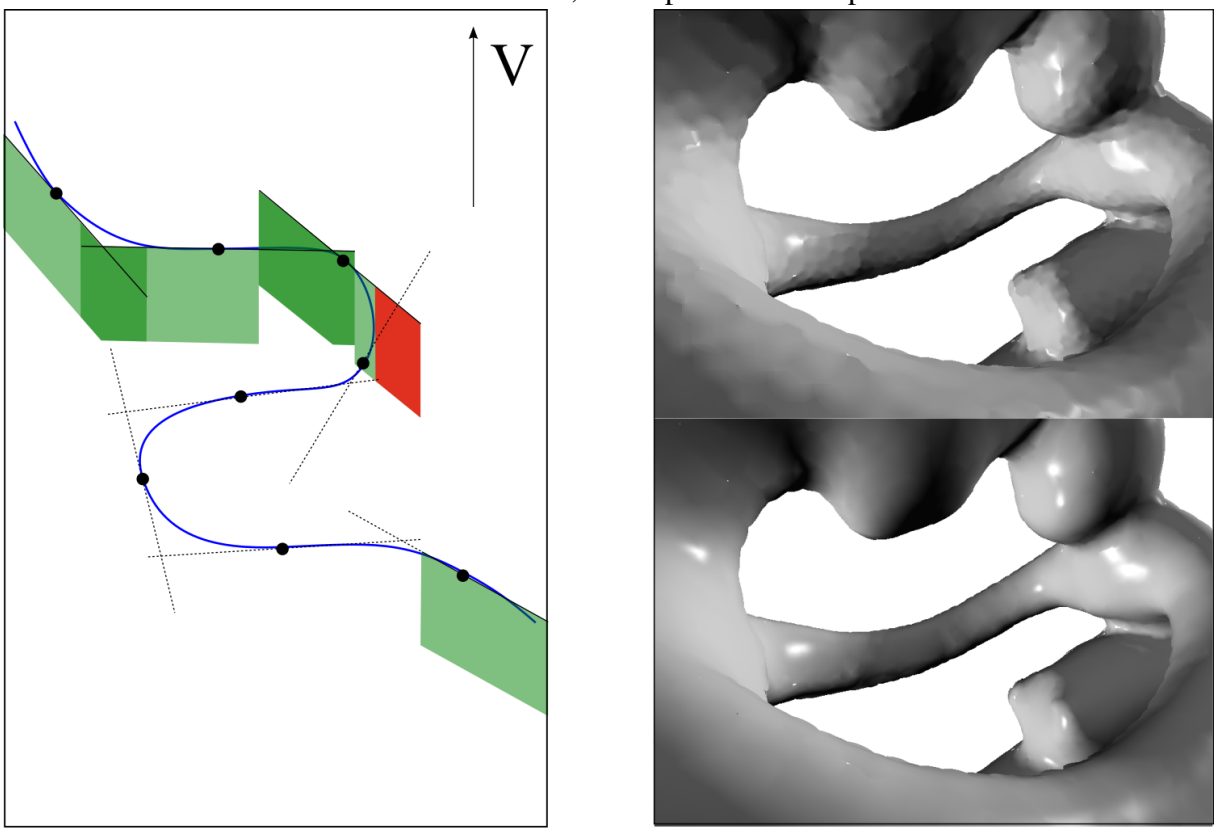
Apesar do fato de nuvens de splats apresentarem a mesma ordem de aproximação de malhas poligonais, as arestas e silhuetas representadas por malhas são pelo menos linhas contínuas, enquanto estas características representadas por splats são descontínuas. A mistura de amostras vizinhas através de filtros, a aplicação de técnicas de sombreamento especiais ou a aplicação de texturas podem elevar a qualidade da imagem renderizada a partir da nuvem de splats, mas não são capazes de esconder os artefatos que, via de regra, aparecem nessas regiões problemáticas.

A combinação de fragmentos através de filtros passa-baixa imprime uma boa renderização final, pois normalmente deseja-se uma transição suave entre as primitivas discretas que representam a superfície. Entretanto, essa transição suave não é desejável sobre as arestas ou nos

cantos de um modelo. Combinar splats nessas regiões pode acarretar a mistura indesejada de superfícies com características potencialmente muito distintas, como cores, texturas e sombreamento. Em contrapartida, não combiná-los acarreta a exibição de discos que se cruzam (Figura 2). Normalmente, quando o número de primitivas de um modelo discreto aumenta, a superfície desejada é representada de forma mais acurada. Assim, os artefatos causados por splats cruzando as linhas das arestas podem ser minimizados se a quantidade das amostras for aumentada e os tamanhos dos splats forem reduzidos. Entretanto, arestas são comumente interpretadas como regiões com curvaturas infinitas em uma certa direção. Por esse motivo, uma quantidade finita de amostras não é suficiente para representá-las apropriadamente. Assim, arestas devem ser representadas explicitamente. A forma mais comum de se fazer isso é utilizar retas para recortar splats e, assim, representar as arestas do modelo. Entretanto, quando as amostras possuem diferentes distribuições e tamanhos no entorno de uma aresta curva, aproximá-la através de recortes retos resulta em artefatos evidentes. Discute-se esse problema em mais detalhes na Seção 2.1.

Artefatos similares ocorrem na silhueta de uma superfície curva. Nessas regiões, splats não serão combinados com seus vizinhos porque eles são excluídos pela etapa de remoção de superfícies ocultas. Dessa forma, na maioria dos casos, splats nas silhuetas possuem regiões onde a combinação com splats vizinhos não ocorre, fazendo com que a estrutura do splat fique aparente na imagem gerada (Figura 3). De acordo com a definição padrão, um ponto **P** está sobre a silhueta de uma superfície suave quando o vetor normal da superfície em **P** é perpendicular ao vetor na direção da câmera partindo de **P**. Detecção de silhueta em malhas é um problema mais fácil de ser solucionado devido à disponibilidade de informações de conectividade entre as amostras. Por outro lado, poucas técnicas foram desenvolvidas para renderizar silhuetas em modelos baseados em pontos e menos ainda para detectar as amostras presentes na silhueta. A abordagem mais comum é utilizar limiarização de vetores normais, onde um ponto **P** pertence à silhueta se o produto escalar entre o vetor normal em **P** e o vetor direcionado à câmera é menor do que um limiar próximo de zero (ZAKARIA; SEIDEL, 2004). Entretanto, esse método pode detectar muitos splats em regiões de baixa curvatura, dependendo de seu ângulo com relação à câmera, ao mesmo tempo que deixa de detectar splats próximos da silhueta em regiões de alta curvatura (Figura 4a). A segunda falha é mais problemática uma vez que as silhuetas de regiões com alta curvatura são as que apresentam maiores artefatos quando renderizadas utilizando splats planos. Em geral, não é possível escolher um limiar fixo adequado que corrija ambos os

Figura 3 – Artefatos presentes nas silhuetas de modelos representados por splats. (a) O vetor **V** indica a direção do observador. A área verde atrás dos fragmentos mais próximos do observador compreende todos os fragmentos que serão combinados para a computação da cor de um determinado pixel. A área vermelha destaca uma região onde o fragmento mais próximo não será combinado com nenhum fragmento de splat vizinho acarretando um artefato na silhueta da superfície. (b) O modelo superior foi renderizado com um cálculo de iluminação por splat, enquanto o inferior foi renderizado com um cálculo de iluminação por fragmento. Ainda que se utilize um melhor método de sombreamento, esse tipo de artefato permanece visível.



(a) (b)

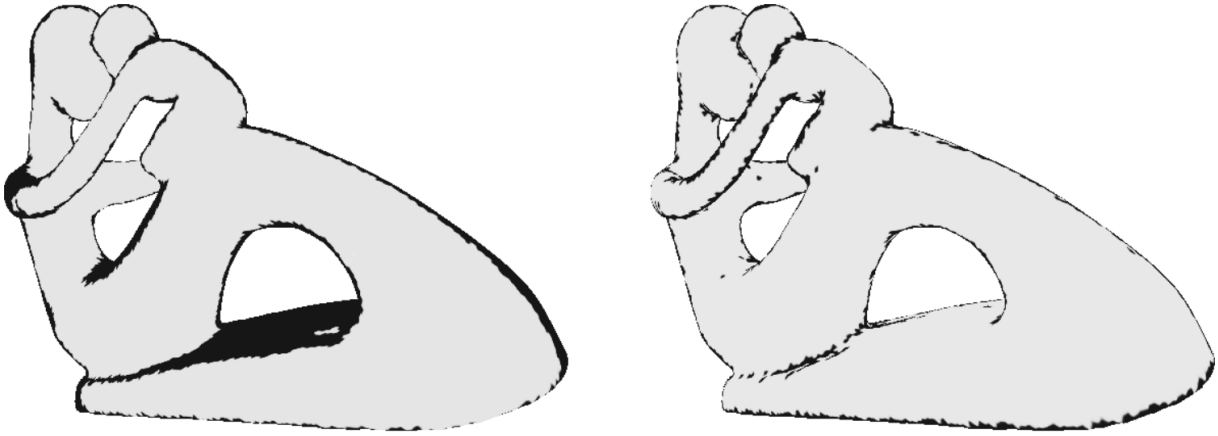
Fonte: o autor.

problemas simultaneamente.

1.1 Objetivos

O principal objetivo dessa tese é reduzir os artefatos decorrentes da renderização de modelos baseados em splats próximos de regiões de arestas e silhuetas, tornando assim a qualidade da renderização mais alta mesmo sob condições de baixa amostragem ou com a proximidade do observador. Nessas situações, um splat ocupa uma área grande em espaço de imagem, causando os artefatos mostrados nas Figuras 2 e 3. Sabe-se que modelos baseados em splats são normalmente gerados a partir de uma amostragem de outras representações, como

Figura 4 – Método de detecção de silhueta. (a) Limiarização de vetores normais pode detectar muitos splats longe da silhueta em regiões de baixa curvatura ao mesmo tempo que deixa de detectar splats próximos de silhueta em regiões de alta curvatura. (b) A detecção de silhueta proposta, usando quadric splats, apresenta um melhor controle independentemente da curvatura da superfície.



(a) (b)

Fonte: o autor.

nuvens de pontos densas, superfícies implícitas, modelos CSG, malhas poligonais etc. A partir dessas representações, é possível extrair informações acerca da superfície que ajudem a adaptar os splats em um grau de aproximação maior do que o já apresentado.

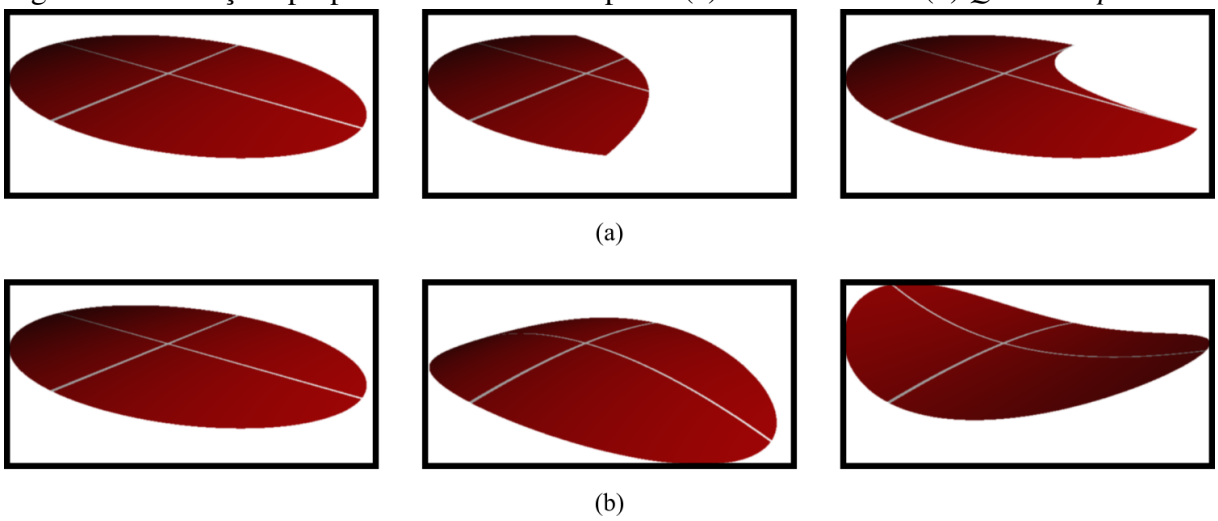
1.2 Contribuições

Este trabalho propõe adaptar o recorte de splats próximos a arestas e a própria forma do splat a graus de aproximação maiores do que as representações lineares atuais com o objetivo de minimizar os artefatos apresentados anteriormente. Essa adaptação visa obter melhor qualidade das imagens geradas a partir de modelos baseados em splats, sem aumentar demasiadamente a quantidade de informação por amostra, explorando o hardware gráfico atual e dando maior liberdade a algoritmos de amostragem para posicionar splats considerando unicamente a curvatura local da superfície e das arestas do modelo.

As técnicas propostas nesta tese orbitam as seguintes contribuições:

- ***Recorte curvo de splats*** (Figura 5a). Cada splat próximo de uma aresta é recortado por uma curva de Bézier racional definida localmente. A qualidade da representação da aresta independe da distância do observador, garantindo a mesma precisão em nível de pixel. A estrutura de dados da curva de recorte é leve e tem tamanho fixo, tornando sua implementação em GPU fácil.

Figura 5 – Alterações propostas na forma dos splats. (a) Recorte curvo. (b) *Quadric Splat*.



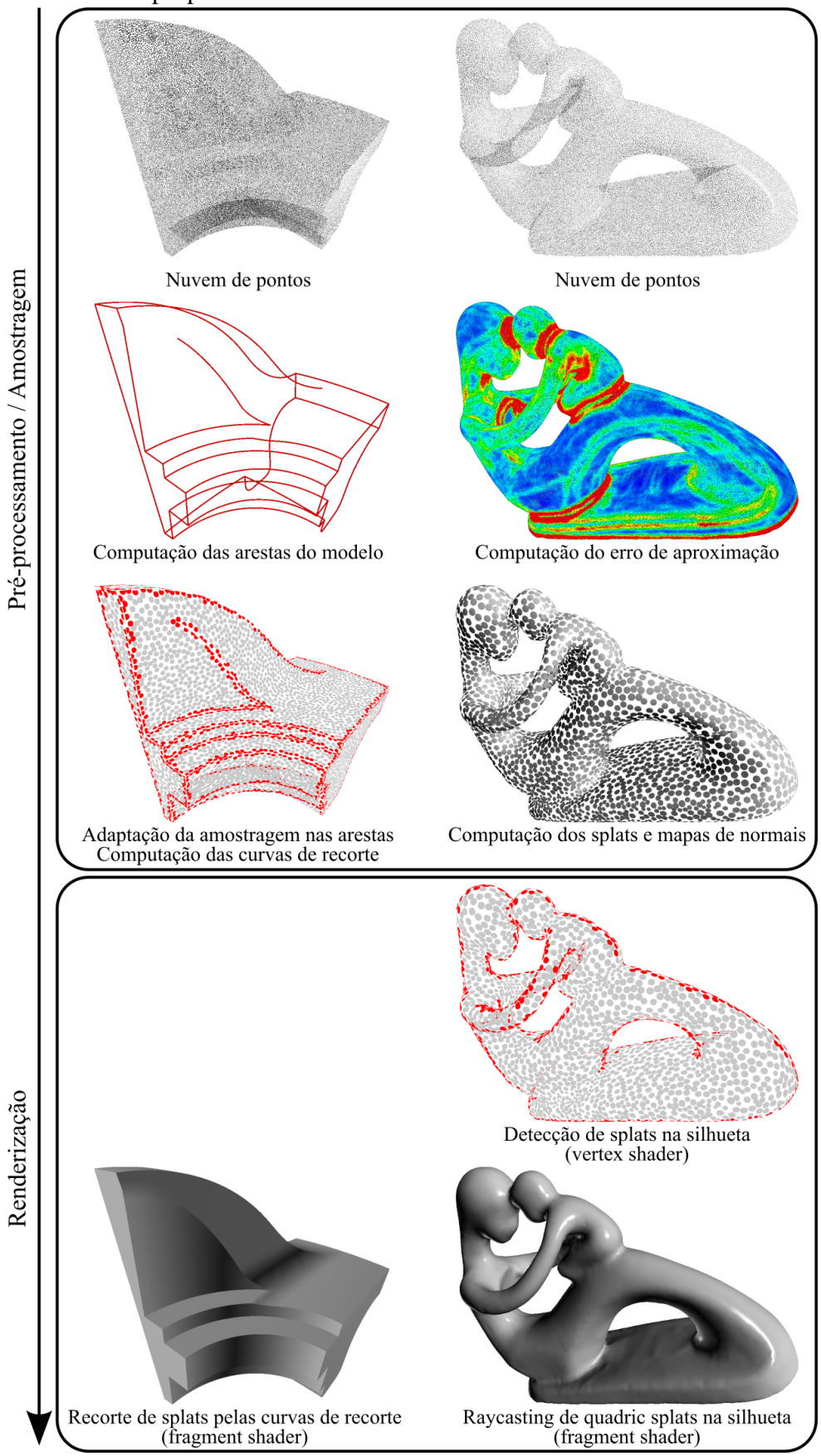
Fonte: o autor.

- ***Quadric splat*** (Figura 5b). Uma primitiva que melhora a qualidade de renderização através da suavização de silhuetas e de splats localizados próximos de arestas curvas.

A Figura 6 mostra as etapas da criação de conteúdo 3D da perspectiva de modelos baseados em pontos onde as técnicas propostas nessa tese se encaixam:

- A partir de uma nuvem de pontos inicial, cada ponto é associado com uma métrica de erro. A métrica proposta nessa tese é calculada com base no erro de aproximação entre um possível quadric splat centralizado no ponto sendo analisado e a nuvem de pontos no seu entorno. (Seção 4.2).
- Durante a geração dos splats, se o modelo possuir arestas, essas devem ser definidas explicitamente por algum método externo: a partir de uma malha poligonal, a partir do usuário, ou a partir de técnicas de detecção de arestas (DANIELS *et al.*, 2008), etc. A maioria dos métodos de geração de splats considera que a superfície é suave, dessa forma propõe-se aqui a adaptar o processo de amostragem próximo de arestas para evitar buracos e sobre-amostragem desnecessária nessas regiões (Seção 3.1).
- Caso o modelo apresente arestas, os splats que possuem interseção com alguma aresta serão detectados e suas curvas de recorte serão computadas de forma automática (Seção 3.3).
- Ao final da geração dos splats utilizando o erro computado inicialmente como critério para as dimensões dos splats, mapas de normais são computados para cada splat. Os mapas de normais podem ser computados de diversas formas, sendo as mais comuns através de mínimos quadrados.
- Durante a renderização, a etapa de vertex shader realiza processamentos individuais para

Figura 6 – Pipeline de criação e renderização de modelos baseados em splats com técnicas utilizadas e propostas nessa tese.



Fonte: o autor.

cada splat. Propomos a detecção de quadric splats próximos à silhueta do modelo em espaço de objeto (Seção 4.4). Isso permite utilizar a renderização mais custosa do quadric splat apenas onde se faz necessário. Além disso, esse método pode ser interessante para renderizações estilizadas de nuvens de splats (Figura 4b).

- Na etapa de processamento de fragmentos da renderização, é realizado o ray casting de quadric splats destacados como pertencentes à silhueta. Os demais são rasterizados como splats planos (Seção 4.3).
- Após a rasterização do splat, caso ele possua uma curva de recorte associada, os fragmentos gerados são classificados com relação a sua posição relativa a essa curva de recorte, removendo partes do splat e adaptando-o às arestas do modelo (Seções 3.2 e 3.4).

1.3 Organização da Tese

O restante desta tese está dividido da seguinte forma:

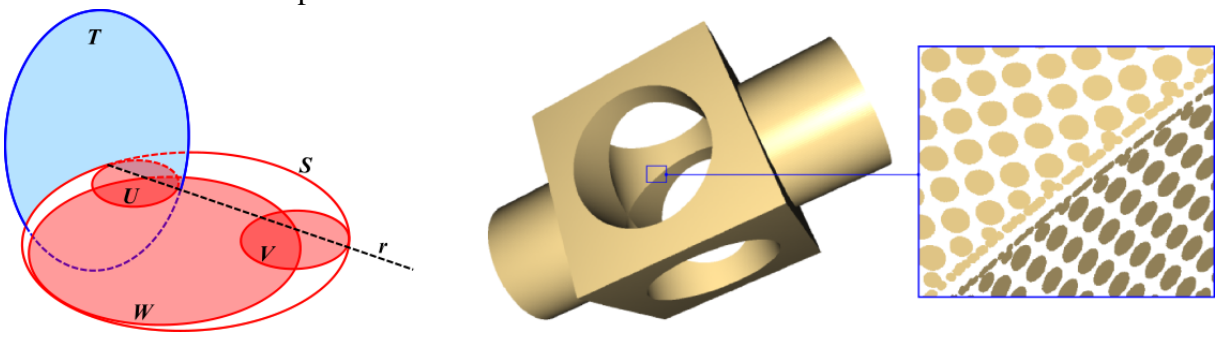
- Capítulo 2 – Descreve e analisa os trabalhos relacionados com os temas abordados nesta tese – renderização de modelos baseados em splats; representação de arestas; representação de superfícies curvas através de amostragens adaptativas e amostras de alta ordem; e algoritmos de detecção de silhuetas.
- Capítulo 3 – Descreve a adaptação de amostragem de splats na proximidade de arestas; define as curvas utilizadas para recorte de splats próximos a arestas, e apresenta o cálculo dessas curvas; e apresenta o método de rasterização de splats afetados por curvas de recorte em GPUs programáveis.
- Capítulo 4 – Apresenta a formulação do *quadric splat* e o método de rasterização desses splats em GPUs; e descreve um método para detectar quadric splats localizados na silhueta do modelo em espaço de objeto.
- Capítulo 5 – Analisa e compara a qualidade e a eficiência das renderizações de nuvens de splats utilizando as técnicas clássicas e as técnicas propostas.
- Capítulo 6 – Apresenta as conclusões da tese, resumindo as técnicas propostas e relatando problemas ainda não solucionados que merecem atenção em possíveis trabalhos futuros.

2 TRABALHOS RELACIONADOS

Pontos foram propostos como primitivas de renderização pela primeira vez em (LEVOY; WHITTED, 1985). Posteriormente, diversas técnicas de renderização baseadas em pontos foram propostas (GROSSMAN; DALLY, 1998; PFISTER *et al.*, 2000; ALEXA *et al.*, 2001; FLEISHMAN *et al.*, 2003). Uma das mais populares dessas técnicas é a *Surface Splatting* (ZWICKER *et al.*, 2001). Splatting é uma técnica simples que obtém imagens de alta qualidade a partir de superfícies amostradas por pontos utilizando anti-aliasing e filtragem de sinais presentes em aplicações de texturas. A ideia básica é estender a amostra de uma nuvem de pontos, que não possui dimensão, a um disco circular ou elíptico em espaço de objeto, denominado *splat*. A sobreposição desses splats preenche os espaços existentes entre cada amostra do modelo. Como splats não são justapostos como os elementos de uma malha, a cor de dois splats vizinhos é misturada para produzir a impressão visual de uma superfície suave. Para realizar essa mistura, cada splat é associado a uma função-peso passa-baixa, normalmente uma gaussiana, e é projetado sobre o *frame buffer*. A cor de cada pixel no frame buffer é determinada pela média ponderada dos splats que o atingiram utilizando a função-peso como parâmetro. Devido à ausência de suporte nativo a splats no hardware gráfico, a renderização baseada em pontos não é direta assim como em malhas triangulares. Entretanto, as GPUs tornaram-se mais e mais flexíveis nas últimas décadas, permitindo a implementação da técnica de Surface Splatting de forma muito mais eficiente (BOTSCH; KOBBELT, 2003; GUENNEBAUD; PAULIN, 2003; BOTSCH *et al.*, 2005; GUENNEBAUD *et al.*, 2006).

A qualidade da renderização apresentada pela técnica de splatting depende diretamente da combinação dos filtros dos splats vizinhos em espaço de imagem. Entretanto, há certas regiões onde a combinação dessas amostras causa problemas. Em certas aplicações, a rasterização de arestas torna-se necessária. Nesse tipo de modelo, a diferença de sombreamento em lados opostos de uma aresta, geralmente, é distinta. Por esse motivo, combinar essas superfícies pelos filtros das amostras acarreta artefatos. A Seção 2.1 relata trabalhos que apresentam propostas de solução para este tipo de problema. As silhuetas são regiões do objeto definidas dinamicamente de acordo com o ponto de vista do observador. Nessas regiões, a renderização da representação por splats é problemática pois a mistura entre amostras vizinhas é reduzida, tornando perceptíveis as amostras que compõem o modelo. Na Seção 2.2, são apresentados trabalhos que propõem abordagens de como representar uma superfície para reduzir esse tipo de artefato. Visto que, nesta tese, é proposta uma abordagem para detectar splats presentes na silhueta, alguns trabalhos

Figura 7 – Representação de arestas proposta em (ADAMS; DUTRÉ, 2003). Splats próximos de arestas são substituídos por outros menores para amenizar os artefatos causados pela mistura de splats nestas áreas.



Fonte: (ADAMS; DUTRÉ, 2003).

relacionados a esse tema são apresentados na Seção 2.3.

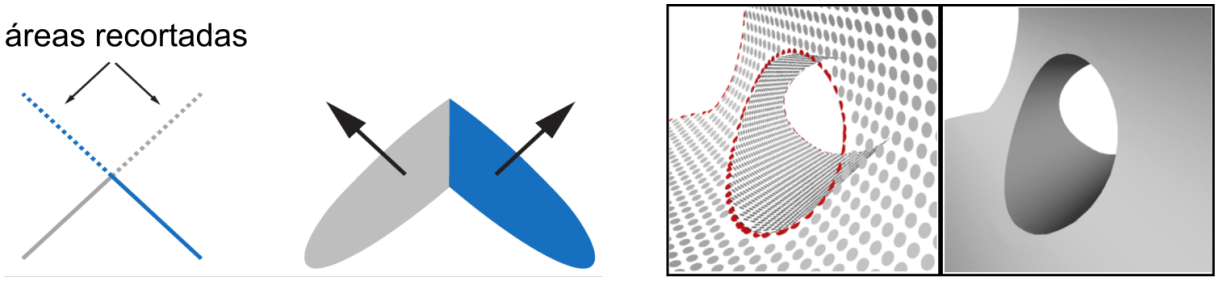
2.1 Representação de Arestas

Uma abordagem comum em modelagem geométrica é construir objetos complexos através da combinação de formas mais simples usando operações booleanas. Representações por nuvens de pontos são atrativas para esse tipo de operação, devido à simplicidade decorrente da ausência de topologia no modelo. Entretanto, modelos resultantes de operações booleanas normalmente apresentam arestas. Por esse motivo, diversos trabalhos que buscam representar arestas em modelos baseados em splats são trabalhos correlacionados à técnica de CSG (*Constructive Solid Geometry*).

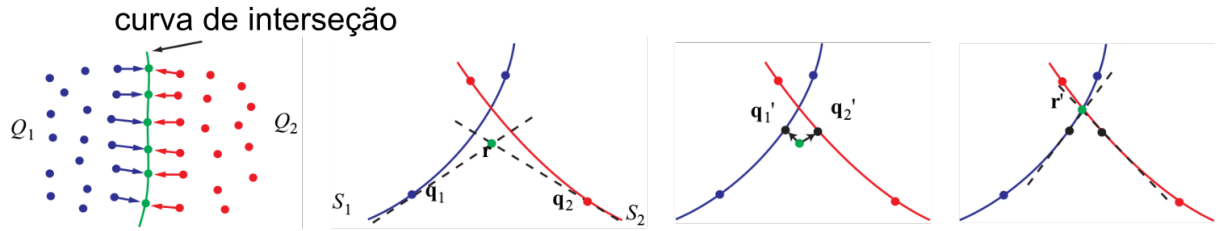
Adams e DutrÉ (2003) utilizam a árvore CSG para sobreamostrar as regiões de interseção, reduzindo assim os artefatos presentes em arestas. As duas superfícies envolvidas em uma determinada operação booleana possuem *octrees* para realizar os testes de áreas internas e externas às superfícies. Splats localizados em regiões de interseção são detectados e substituídos por vários menores. Essa abordagem reduz a área borrada a tamanhos próximos de um pixel de largura. Apesar de ocultos a certa distância, a técnica jamais poderia conciliar a natureza dos splats com as arestas e os artefatos podem se tornar visíveis com a aproximação do observador (Figura 7).

Pauly *et al.* (2003b) inseriram um novo tipo de primitiva específica para representar as curvas de interseção entre duas superfícies envolvidas em uma operação booleana. Essa primitiva é representada por dois splats que compartilham um mesmo centro, mas com normais divergentes representando as orientações das superfícies em torno da curva de interseção. Durante

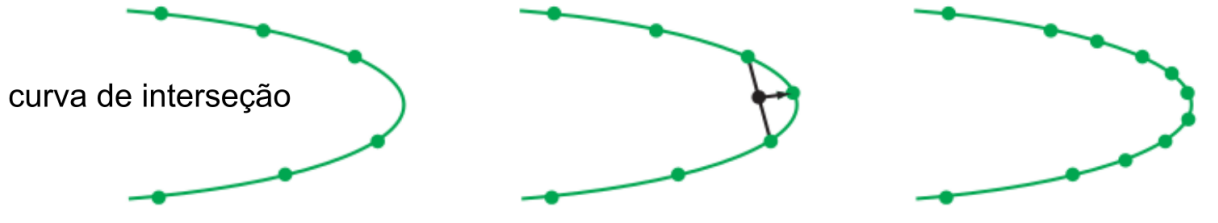
Figura 8 – Representação de arestas proposta em (PAULY *et al.*, 2003b). (a) Splat especial equipado com dois vetores normais são formados por dois discos recortados um contra o outro. (b) Método iterativo para posicionamento das amostras sobre as curvas de interseção. (c) Processo de sobreamostragem adaptativa dessas primitivas sobre a curva de interseção de duas superfícies. (d) Diferentes níveis de refinamento uniforme utilizados para aumentar a qualidade da representação do modelo.



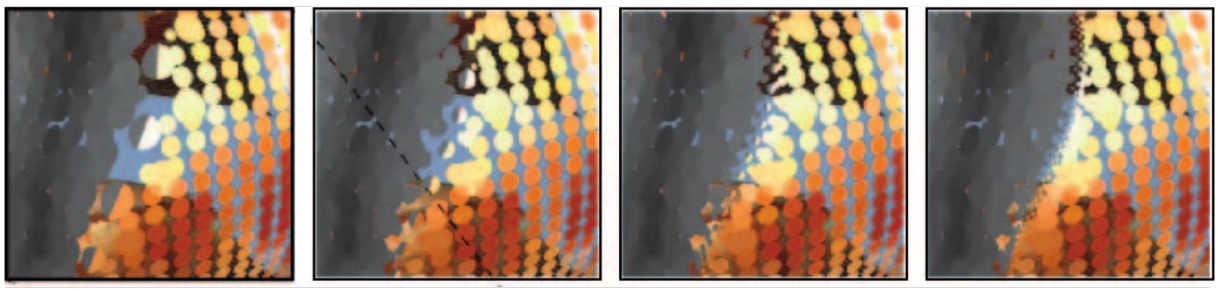
(a)



(b)



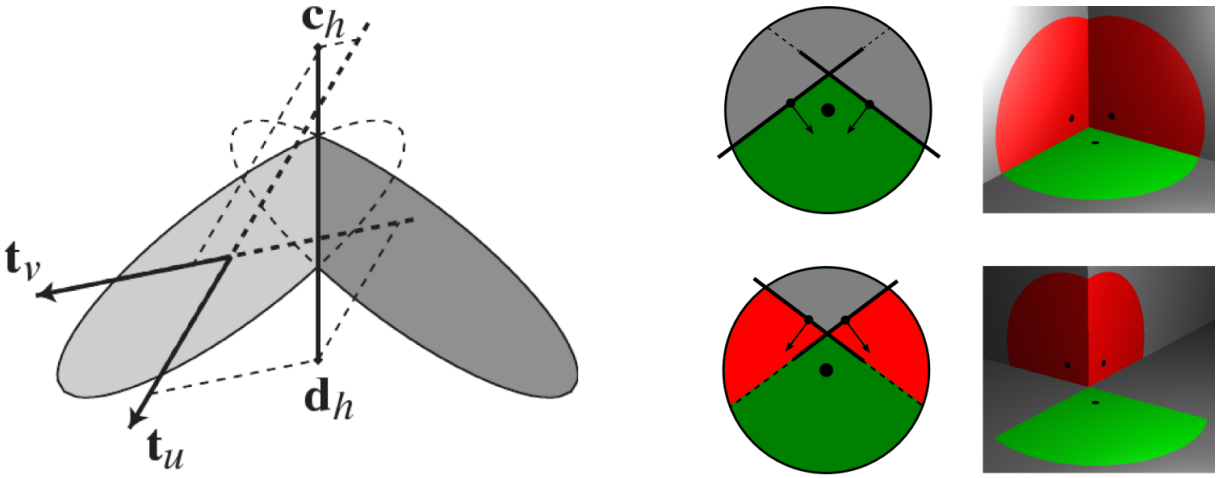
(c)



(d)

Fonte: (a-c) (PAULY *et al.*, 2003b), (d)(GROSS; PFISTER, 2007)

Figura 9 – Representação de arestas proposta em (ZWICKER *et al.*, 2004). (a) *Clipping line* definida na reta de interseção dos planos de splats em lados opostos de uma aresta. (b) Ambiguidade quando mais de uma *clipping line* afeta um determinado splat. Teoricamente, ambos definem a mesma área de recorte, mas os casos são diferentes.



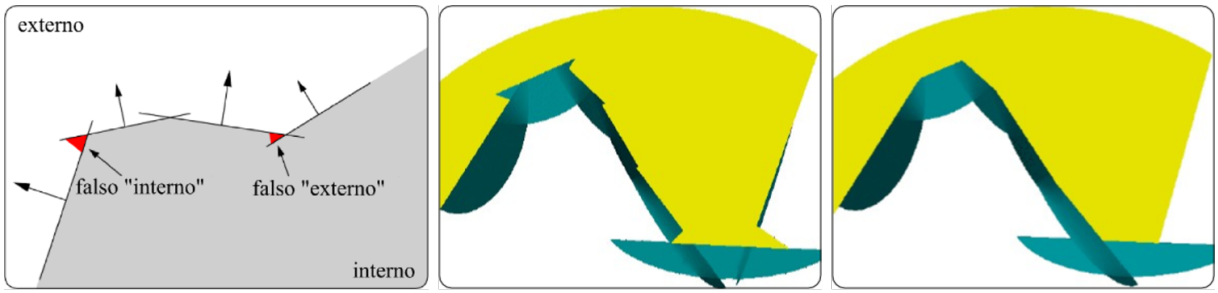
(a) (b)

Fonte: (a) (ZWICKER *et al.*, 2004) e (b) o autor

a rasterização dessa primitiva, cada um dos splats é então recortado pelo plano definido pelo outro, obtendo uma aproximação linear local da curva de interseção em espaço de imagem (Figura 8a). Um problema decorrente dessa abordagem é a necessidade de reamostragem nos entornos da área afetada pela curva de interseção. Splats que cruzam a curva deverão ser removidos e substituídos por outros menores. Como os splats que representam a aresta devem estar exatamente sobre a aresta, um método iterativo é necessário para posicionar uma amostra sobre a curva, utilizando projeções MLS (ALEXA *et al.*, 2001) (Figura 8b). A aproximação linear da aresta do modelo requer uma sobreamostragem da curva de interseção para melhorar a qualidade da representação, principalmente em regiões onde a aresta possui alta curvatura (Figura 8c). Essas arestas podem ocorrer entre superfícies com curvaturas bastante distintas e que podem possuir splats de tamanhos bastante distintos. Entretanto, como os discos com centro compartilhado também compartilham seu tamanho, uma sobreamostragem é feita em uma superfície com curvatura relativamente baixa, ou seja, que poderia aceitar amostras maiores e em menor quantidade (Figura 8d).

Zwicker *et al.* (2004) apresentaram a adição das denominadas *clip lines* – retas locais sobre o plano do splat utilizadas para recortar uma das divisões do splat. Esse método oferece algumas vantagens: a *clipping line* não precisa passar obrigatoriamente pelo centro do splat, não sendo, portanto, necessário posicionar splats recortados com os centros exatamente sobre as arestas do modelo; e não é obrigatório manter as informações de modelagem, exigidas

Figura 10 – Representação de arestas proposta em (WICKE *et al.*, 2004). (a) Utilizar apenas o splat mais próximo para realizar a classificação de um fragmento como interno ou externo a outro objeto pode resultar em alguns erros. (b) Exemplo de aresta apresentando serrilhamento devido à utilização apenas do *clip partner* mais próximo no cálculo de seu recorte. (c) Mesmo caso anterior utilizando dois *clip partners* mais próximos a cada fragmento no processo de classificação, resultando em uma aresta mais contínua.

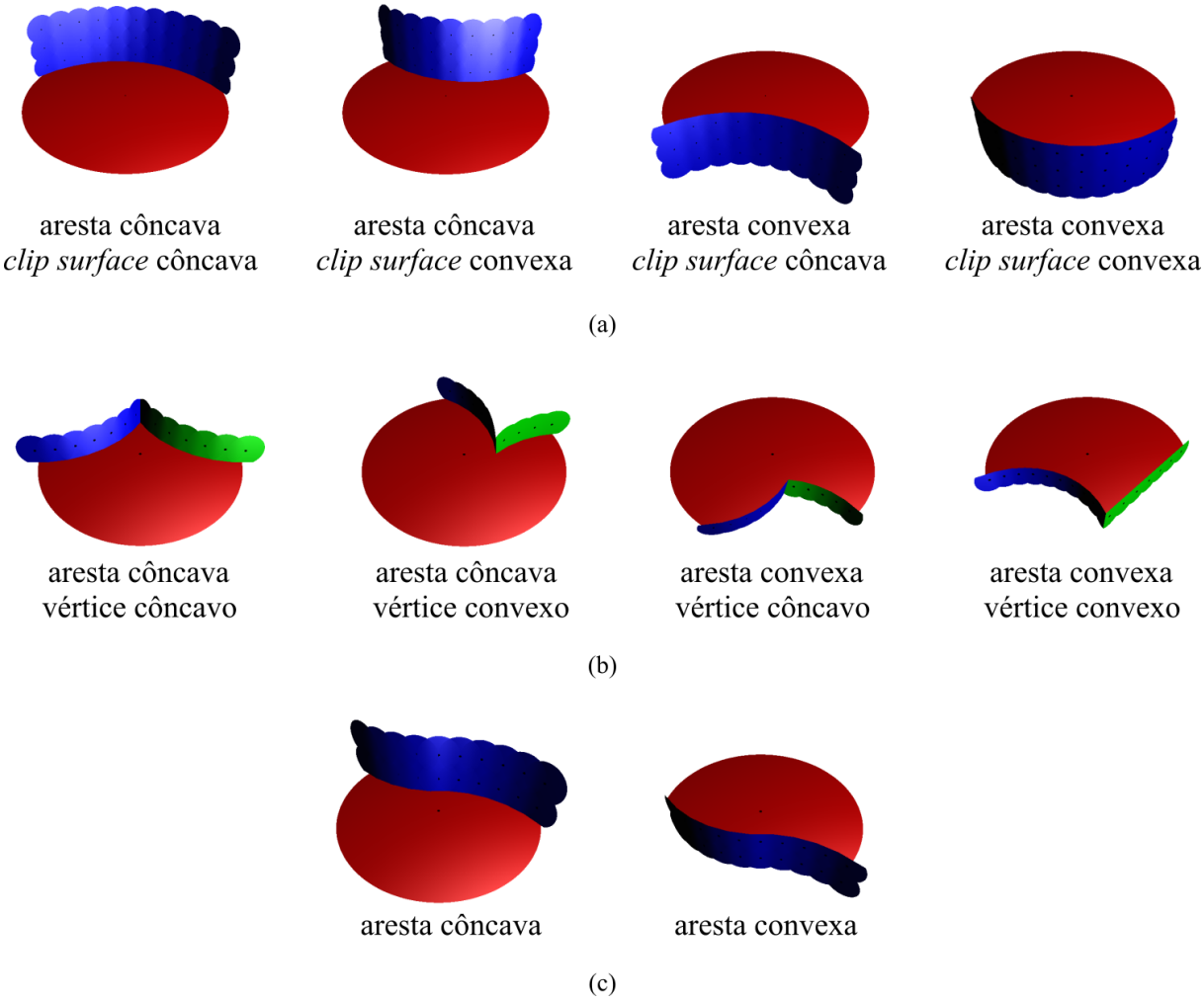


Fonte: (WICKE *et al.*, 2004)

nos trabalhos anteriores. Neste método, um splat detectado como pertencente a uma aresta é recortado pela reta de interseção do seu plano com o plano de outro splat presente no lado oposto da aresta (Figura 9a). Os problemas ocorrem quando as densidades das amostras e as curvaturas das superfícies são diferentes em torno da aresta. Uma aresta curva não pode ser adequadamente aproximada por linhas retas, e splats grandes tornam esse problema mais evidente. Além disso, quando mais de uma *clip line* afeta um determinado splat, como no caso de um canto, os resultados podem ser ambíguos, resultando em buracos e tornando a representação de interseções complexas impossível (Figura 9b).

Wicke *et al.* (2004) não adicionam novos splats ao modelo e solucionam o problema da ambiguidade presente na técnica de (ZWICKER *et al.*, 2004) ilustrada na Figura 9b. Entretanto, a técnica é aplicada somente a modelos representados por uma árvore CSG. Quando um determinado splat S está localizado sobre a curva de interseção de duas superfícies envolvidas em uma operação booleana, todos os splats que possuem alguma interseção com S, mas fazem parte da outra nuvem de splats, são denominados por *clip partners*. Durante a rasterização, as coordenadas de cada fragmento de S são mapeadas para um sistema de coordenadas local. Quando diversos splats intersectam S, os dois *clip partners* mais próximos deste ponto 3D classificam o fragmento como interno ou externo ao objeto oposto. Nas técnicas anteriores, quando essa situação ocorre, apenas o mais próximo é utilizado para realizar o recorte, mas essa escolha torna a aresta serrilhada (Figura 10a-b). No trabalho de Wicke *et al.* (2004), a classificação do fragmento depende de dois fatores: da operação booleana realizada entre os

Figura 11 – Representação de arestas proposta em (IVO *et al.*, 2012). (a) Casos básicos de recorte onde a superfície representada pelos *clip partners* é inteiramente côncava ou convexa. (b) Casos onde um vértice é representado. (c) Casos onde a curva da aresta possui uma parte côncava e outra convexa.

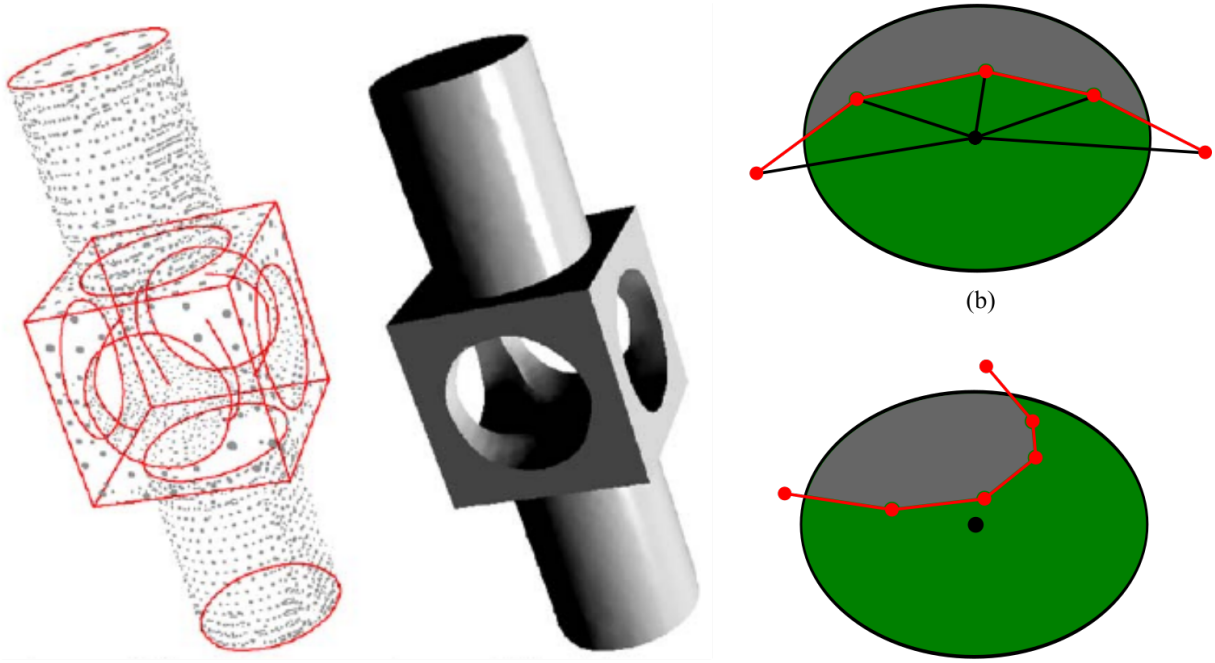


Fonte: (IVO *et al.*, 2012)

objetos; e da orientação relativa entre os clip partners envolvidos,isto é, se eles formam uma área côncava ou convexa entre si. A Figura 10c mostra que as arestas obtidas com este método apresentam menos descontinuidades, gerando menos artefatos. Os problemas dessa técnica são: sua dependência da informação de modelagem para a decisão de remoção de um determinado fragmento; e a exigência de busca pelos clip partners mais próximos em tempo de renderização. Esses problemas dificultam a implementação da técnica em GPU, pois diferentes fragmentos de um mesmo splat podem precisar de informações distintas.

Ivo *et al.* (2012) generalizaram a ideia proposta em (WICKE *et al.*, 2004) sem a necessidade de informação de modelagem. O trabalho propõe que os *clip partners* sejam todos os splats presentes no lado oposto de uma aresta que participarão do recorte de um determinado splat S. Se a superfície α representada pelos *clip partners* de um certo splat S for completamente

Figura 12 – Representação de arestas proposta em (ZHANG; KAUFMAN, 2007). (a) A estrutura de dados híbrida: splats e as polilinhas de recorte. (b) Para permitir a implementação em GPU, um *triangle fan* entre o centro do splat e os vértices da polilinha é renderizado para representar um splat recortado. (c) Essa abordagem exige que o centro do splat esteja no mesmo lado de todos os segmentos, ou seja, não permite casos como este.



(a) (b) (c)

Fonte: (a) (ZHANG; KAUFMAN, 2007) e (b-c) o autor

côncava ou convexa, a união ou a interseção das áreas de recorte de cada *clip partner* é realizada para adaptar S à aresta (Figura 11a). Se α possuir uma aresta ou uma zona de transição entre uma parte côncava e outra convexa, os *clip partners* são divididos em dois grupos. Cada grupo é classificado usando o mesmo método para o caso simples e os resultados são combinados apropriadamente (Figura 11b-c). Como a etapa de classificação não é feita entre um fragmento e os splats, mas entre as próprias amostras em espaço de objeto, essa etapa pode ser realizada em um pré-processamento, removendo um pouco do peso da etapa de renderização. Entretanto, como a quantidade de informação enviada para recortar cada splat é variável, pois a quantidade de *clip partners* é variável para cada splat, a implementação precisa ser feita em software, tornando a renderização muito ineficiente quando comparada com as técnicas que utilizam GPU.

Zhang e Kaufman (2007) propõem o uso de uma estrutura de dados híbrida na representação de objetos com arestas: além dos splats, as arestas são representadas explicitamente por um conjunto de polilinhas que podem ser inseridas diretamente pelo usuário ou por um método de detecção de arestas (GUMHOLD *et al.*, 2001; KOBBELT *et al.*, 2001; PAULY *et al.*,

2003a; DANIELS *et al.*, 2008). Quando um dado splat é interceptado por até dois segmentos da polilinha, o recorte é feito assim como nas técnicas anteriores. Entretanto, quando o splat intercepta mais do que dois segmentos, o splat é renderizado usando um *triangle fan* (Figura 12b). Como a quantidade de segmentos de reta que interceptam um splat pode ser variável, os autores propõem a escrita de um par de shaders para cada quantidade de segmentos utilizados no recorte para evitar a execução da técnica puramente em CPU. Além dessa restrição, a técnica não permite casos onde o centro do splat esteja em lados diferentes de dois segmentos usados para recorte, ou seja, a técnica não trata casos como aqueles mostrados na Figura 12c.

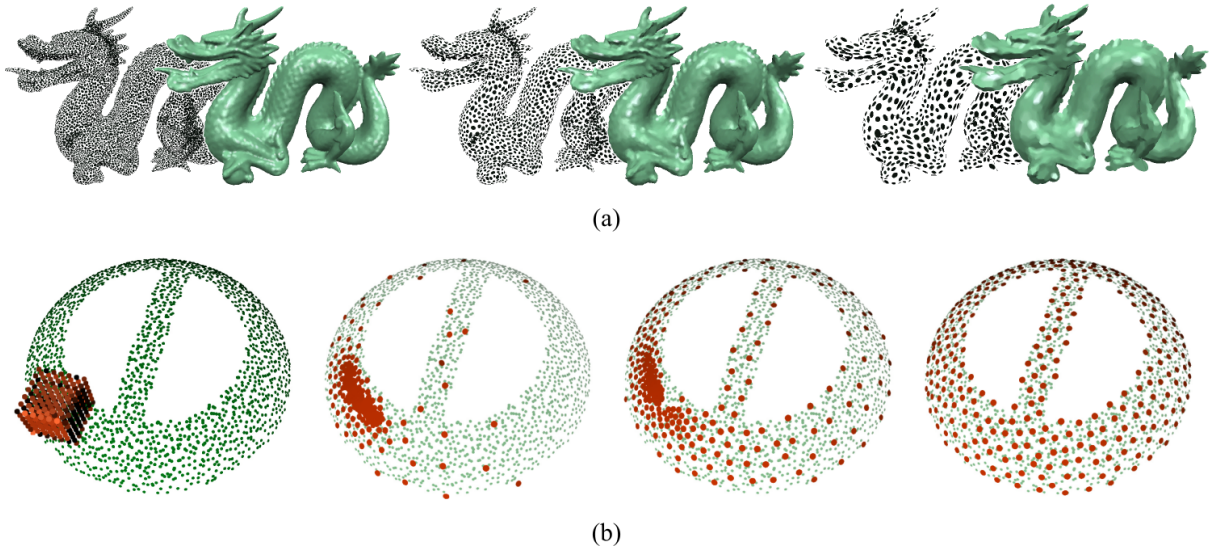
2.2 Representação de Superfícies

A técnica de surface splatting apresenta renderizações de alta qualidade em modelos representados por splats devido à combinação de amostras vizinhas utilizando filtros em espaço de objeto e espaço de imagem. Entretanto, ainda trata-se de uma técnica de renderização de modelos discretos, e, assim como em qualquer outra representação discreta, para obter renderizações de alta qualidade, pode alterar a densidade de amostras do modelo (Seção 2.2.1) ou pode alterar a forma das amostras (Seção 2.2.2). Nessas seções, são apresentados os trabalhos com essas abordagens que tratam do problema de representação de silhuetas.

2.2.1 Amostragem

Nuvens de pontos são representações discretas no sentido de que consistem em um número finito de amostras. Essas amostras devem representar uma superfície contínua, então é muito importante determinar sua densidade apropriada para capturar os detalhes geométricos relevantes de um modelo. Geralmente, algoritmos de amostragem são aplicados antes da técnica de splatting para que os tamanhos dos splats em espaço de imagem sejam menores do que o limiar definido pelo usuário (dois pixels, por exemplo). Técnicas que alteram a amostragem de uma nuvem de pontos são normalmente divididas em técnicas de sobreamostragem, quando aumentam o número de amostras, e de subamostragem, quando diminuem o número original de amostras do modelo. Na literatura, as técnicas de subamostragem receberam maior atenção porque, normalmente, as nuvens de pontos são obtidas a partir de scanners 3D, que geram uma quantidade massiva de amostras.

Figura 13 – Abordagens de sub-amostragem. (a) Remoção iterativa de amostras através de operações de fusão (WU *et al.*, 2005). (b) Distribuição de amostras aleatórias e posicionadas por simulação de repulsão de partículas (LIPMAN *et al.*, 2007).



(a)

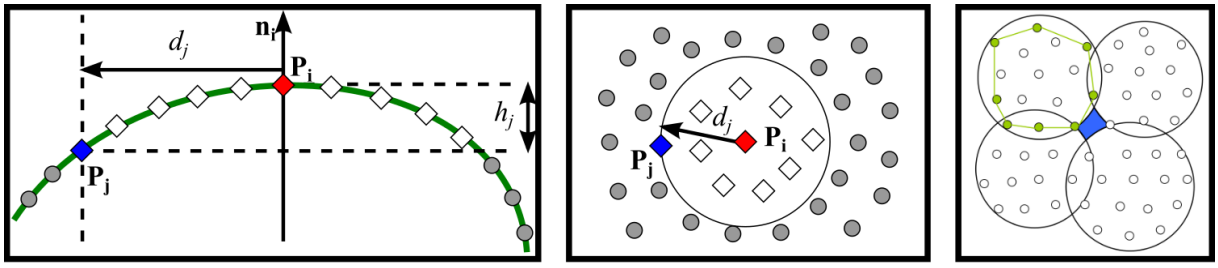
(b)

Fonte: (a) (WU *et al.*, 2005) e (b) (LIPMAN *et al.*, 2007).

Técnicas de subamostragem normalmente implementam níveis de detalhe (*LOD – Level-Of-Detail*) em modelos representados por pontos. *LOD* e técnicas de multirresolução dependente do observador foram estabelecidas para ajustar desempenho e qualidade de exibição de objetos e cenas complexas e grandes aos recursos de renderização disponíveis (LUEBKE *et al.*, 2003). Nuvens de pontos são uma representação bastante útil para abordagens de níveis de detalhe, pois a inexistência de informações topológicas permite implementá-las simplesmente adicionando ou removendo pontos.

Uma exploração sistemática de diferentes abordagens para simplificação de nuvens de pontos foi proposta por Pauly *et al.* (2002) e continuou sendo desenvolvida por diversos outros autores. Os métodos podem ser classificados nas seguintes categorias principais: simplificação iterativa, simulação de partículas e clusterização. Na primeira abordagem, operações de fusão entre pares de pontos reduz o número de pontos (Figura 13a). Normalmente, essas operações são dispostas em uma fila de prioridade de acordo com uma métrica que quantifica o erro causado pela junção dos pontos originais (WU *et al.*, 2005). Na segunda abordagem, um determinado número de amostras é randomicamente espalhado através da superfície e suas posições são ajustadas usando algoritmos de repulsão de partículas (Figura 13b). A movimentação das novas amostras é restrita sobre a superfície definida pela nuvem de pontos original para assegurar uma aproximação mais precisa (LIPMAN *et al.*, 2007).

Figura 14 – Abordagens de subamostragem por clusterização propostas em (WU; KOBBELT, 2004). (a) Um splat centrado em $\mathbf{P_i}$ cresce ao adicionar pontos vizinhos até encontrar o limite no seu raio ou na sua variação vertical. (b) Visão do crescimento do splat a partir de sua normal. (c) Uma amostragem de splats envolvendo todos os pontos da nuvem pode não ser suficiente para garantir uma amostragem livre de buracos.



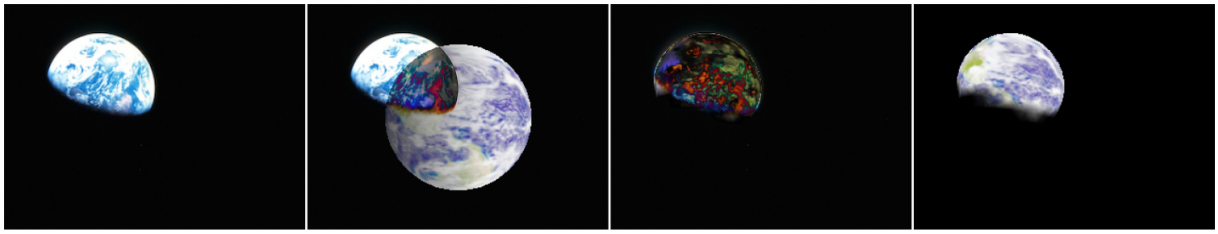
(a) (b) (c)

Fonte: (WU; KOBBELT, 2004)

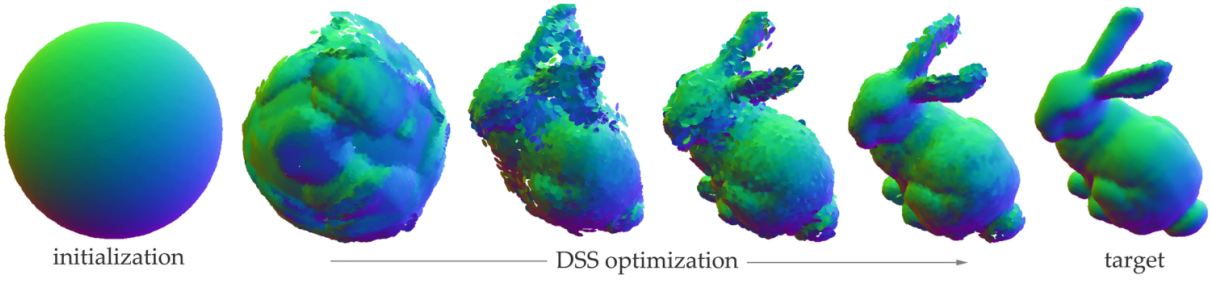
Pode-se dizer que a abordagem de subamostragem mais comum é a clusterização. Nessa abordagem, um conjunto inicial de pontos é subdividido em grupos, cada um dos quais passa a ser representado por uma amostra que aproxime a superfície localmente. Em geral, esses grupos são limitados pelo seu tamanho, em termos de diâmetro, ou pela sua variação, em termos de normal ou análise de covariância. A clusterização hierárquica, por exemplo, agrupa pontos de forma recursiva ao dividir a nuvem de pontos usando alguma estrutura de dados de particionamento como *octree*, árvore BSP (*Binary Space Partitioning*), *k-d tree* etc. Após a construção da estrutura de dados, níveis de detalhe apropriados de acordo com a posição relativa do observador são selecionados (RUSINKIEWICZ; LEVOY, 2000; PAJAROLA, 2003). Mais tarde, a análise hierárquica dessas estruturas de dados foi sequencializada permitindo a implementação delas em GPU (DACHSBACHER *et al.*, 2003). As nuvens de splats utilizadas neste trabalho foram construídas através da abordagem de clusterização por crescimento de região. Nessa abordagem, splats são gerados ao escolher pontos iniciais e pontos vizinhos vão sendo adicionados gradualmente até que algum limite seja alcançado (WU; KOBBELT, 2004) (Figuras 14a-b).

Independentemente do método de subamostragem, uma das tarefas mais difíceis é garantir uma amostragem livre de buracos. Uma condição comumente verificada é se todos as amostras de uma nuvem de pontos estão cobertas por pelo menos um splat. Essa condição, infelizmente, não é suficiente para garantir uma amostragem livre de buracos (Figura 14c). Por esse motivo, Wu e Kobbelt (2004) definem que, de todos os pontos sob um determinado splat, aqueles que pertencem ao fecho convexo são definidos como não cobertos pelo splat. Essa restrição assegura que esses pontos sejam cobertos pelos splats vizinhos, garantindo uma

Figura 15 – Renderização diferencial. (a) Adaptação de modelos e demais atributos da cena para aproximar a imagem renderizada a uma imagem objetivo. (b) Renderização diferencial baseada em pontos permite mudanças topológicas substanciais preservando detalhes geométricos.



(a)



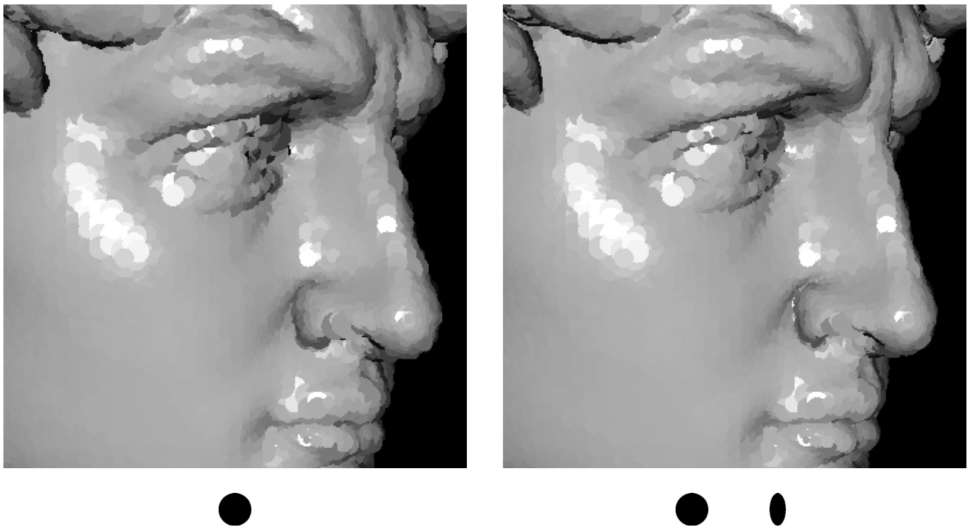
(b)

Fonte: (a)(LOPER; BLACK, 2014) e (b)(YIFAN *et al.*, 2019).

amostragem livre de buracos. Entretanto, a técnica parte da premissa da superfície ser suave. Ou seja, em superfícies com arestas, a amostragem pode ter alguns efeitos colaterais indesejados como buracos ou sobreamostragem. Esse problema é melhor discutido na Seção 3.1, onde uma adaptação a essa condição, que garante uma nuvem de splats sem buracos e sem sobreamostragem mesmo próximo a arestas do modelo, é apresentada.

Nos últimos anos, o processamento diferencial de informações do modelo a partir de uma imagem objetivo está emergindo como um componente fundamental em modelagem 3D. A ideia básica de um renderizador diferencial pode ser descrita como a seguir: um renderizador $\Re$ recebe como entrada informações θ tais como a geometria da cena, iluminação, material e posição da câmera como entrada e retorna como saída uma imagem $\Phi = \Re(\theta)$. Em essência, um renderizador diferencial $\Re$ é desenvolvido para propagar mudanças em espaço de imagem aos parâmetros em espaço de objeto θ. Essa informação pode ser usada para otimizar os parâmetros de forma que a imagem renderizada $\Phi = \Re(\theta)$ aproxime uma imagem de referência Φ^*. Algumas técnicas baseadas nessa premissa foram desenvolvidas para modelos baseados em pontos (INSAFUTDINOV; DOSOVITSKIY, 2018; LIN *et al.*, 2018; ROVERI *et al.*, 2018). O trabalho apresentado por Yifan *et al.* (2019) é o primeiro e ate então único baseado em modelos formados por splats. Apesar de ser uma ótima ferramenta de modelagem, as primitivas utilizadas

Figura 16 – Splats circulares e elípticos em espaço de imagem (RUSINKIEWICZ; LEVOY, 2000). À esquerda, todos os splats são circulares com 20 pixels de largura. À direita, os splats são elípticos e são rotacionados de acordo com a projeção da normal do ponto.



Fonte: (RUSINKIEWICZ; LEVOY, 2000).

continuam sendo lineares e partilham da mesma problemática das demais técnicas de amostragem em regiões próximas de arestas e silhuetas.

2.2.2 Amostras de alta ordem

A escolha da forma do splat para representar uma área local da superfície tem efeitos significantes na qualidade final da imagem. Rusinkiewicz e Levoy (2000) propõem diversos tipos de formas em espaço de imagem para representar as projeções dos pontos. Entre elas, a opção mais simples e rápida é um *OpenGL point*, que é renderizado como um quadrado. Outras formas um pouco melhores são círculos opacos ou difusos. Apesar dessas primitivas serem inferiores aos splats elípticos em espaço de objeto (ZWICKER *et al.*, 2001), o trabalho traz um cuidado especial com a representação de silhuetas ao utilizar formatos diferentes para pontos presentes nessas regiões. Cada ponto pode ser representado por uma elipse em espaço de imagem. O eixo menor da elipse aponta para a projeção do vetor normal do ponto no plano da imagem. Essa escolha melhora a qualidade das silhuetas suaves (Figura 16). Entretanto, a escolha dessas primitivas não garante que a reconstrução do modelo em espaço de imagem seja livre de buracos. Na prática, ocasionalmente, buracos aparecem quando discos elípticos são usados próximos de arestas presentes nas silhuetas do objeto.

As representações em espaço de objeto em combinação com filtros em espaço de

Figura 17 – Pontos Diferenciais (KALAIAH; VARSHNEY, 2003). (a) Definição do ponto diferencial: um retângulo com um mapa de normais variando linearmente na direção de seus eixos. (b) Renderização dos wireframes dos pontos diferenciais. (c) Renderização final usando os pontos diferenciais.

(a) (b) (c)

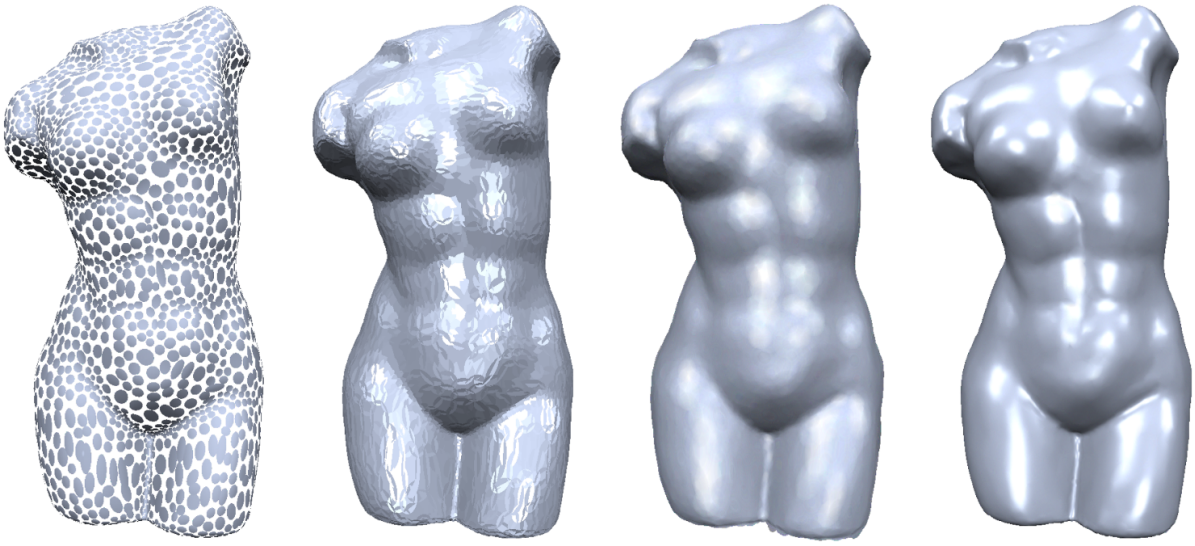
Fonte: (KALAIAH; VARSHNEY, 2003).

imagem (ZWICKER *et al.*, 2001) apresentam a melhor qualidade de imagem e simplicidade, garantindo mais facilmente amostragens livres de buracos. Como dito anteriormente, a distribuição de splats planos é uma das dificuldades encontradas nas representações deste tipo de modelo. Uma alternativa a alterações na amostragem é usar amostras com maior grau de aproximação e que melhor se adaptem à superfície que ela se propõe a aproximar. Essas abordagens normalmente são baseadas em geometria diferencial. Superfícies MPU (*Multi-level Partition of Unity*) (OHTAKE *et al.*, 2003) e SLIM (*Sparse Low-degree Implicits*) (OHTAKE *et al.*, 2005), por exemplo, usam pedaços de superfícies quadráticas ou cúbicas para aproximar ou interpolar superfícies altamente detalhadas. Em ambas as técnicas, superfícies polinomiais são combinadas para gerar uma representação de superfície suave. Entretanto, os algoritmos de renderização dessas técnicas são baseados em software, comprometendo sua eficiência.

Kalaiah e Varshney (2003) introduziram pontos diferenciais que incorporam informações de curvatura locais da superfície para iluminação e sombreamento. Cada amostra armazena as direções principais da curvatura local e as intensidades da variação do vetor normal nas direções das curvaturas locais. Diferentemente das representações de splats convencionais, que usam um único vetor normal por splat, pontos diferenciais são retângulos com um mapa de normais (Figura 17). Os pontos diferenciais são categorizados em 256 variações baseando-se nas combinações relativas das curvaturas principais. Cada variação possui sua própria distribuição de normais. Todas essas distribuições são armazenadas em uma textura para serem acessadas durante o tempo de execução. Pontos diferenciais são excelentes na renderização de superfícies suaves, entretanto, a ausência de interpolação entre os pontos tornou-se uma desvantagem.

Botsch *et al.* (2004) combinaram as amostras diferenciais apresentadas em (KA-

Figura 18 – Phong Splatting (BOTSCH *et al.*, 2004). Da esquerda para direita: distribuição das amostras, sombreamento por splats sem mistura de amostras vizinhas, sombreamento por splat usando mistura de amostras vizinhas, Phong splatting.

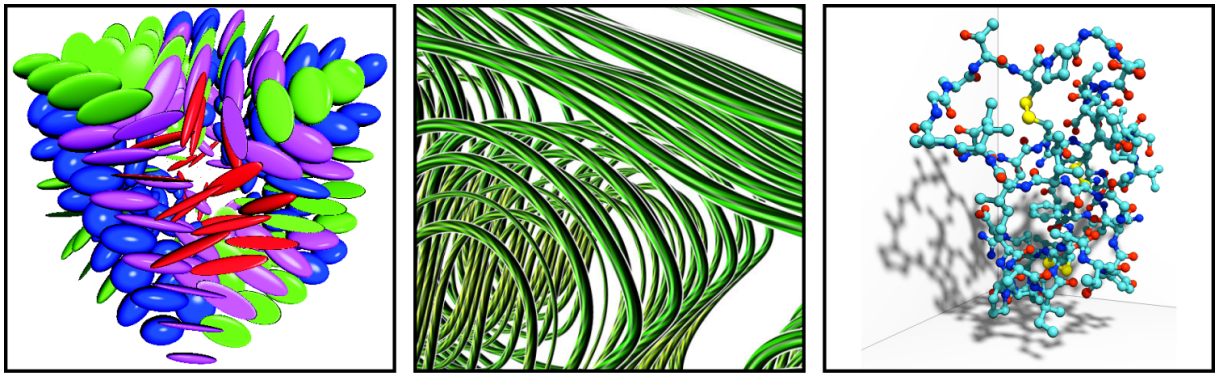


Fonte: (BOTSCH *et al.*, 2004).

LAIAH; VARSHNEY, 2003) com a técnica de surface splatting. Os mapas de normais são computados após um método de subamostragem que cria uma nuvem de splats a partir de uma nuvem de pontos. Cada splat formado após a simplificação é associado com um conjunto de pontos coberto por ele (WU; KOBBELT, 2004). Esses pontos são usados para aproximar uma função quadrática a ser usada como mapa de normais do splat. Essa técnica produz renderizações de alta qualidade mesmo sob baixas amostragens (Figura 18) e torna-se eficiente quando combinada com a implementação em GPU proposta em (BOTSCH *et al.*, 2005). Apesar da melhoria significativa no sombreamento de modelos, os splats continuam sendo rasterizados por superfícies planas, tornando a motivação de utilizar poucas amostras em um problema nas áreas no entorno de silhuetas.

Superfícies quadráticas têm sido primitivas básicas em computação gráfica por bastante tempo e suas propriedades e técnicas para visualização foram exaustivamente estudadas. Algumas técnicas foram desenvolvidas para usar hardware gráfico para visualizar eficientemente superfícies quadráticas ou pedaços delas. Uma das primeiras abordagens foi apresentada em (GUMHOLD, 2003), onde elipsoides eram rasterizados na tela através de *ray casting*. Posteriormente diversos trabalhos fizeram adaptações de implementação para as mais diversas finalidades, como visualização de campos magnéticos (KLEIN; ERTL, 2004; REINA; ERTL, 2005), renderizações estilizadas de linhas (STOLL *et al.*, 2005), visualização de estruturas moleculares (SIGG *et al.*, 2006) etc (Figura 19). A maioria dessas técnicas são específicas

Figura 19 – Splatting de superfícies quadráticas fechadas utilizadas para diversas finalidades. (a) Renderização de campos de tensores por elipsoides (GUMHOLD, 2003). (b) Renderização de linhas estilizadas por agrupamento de cilindros (STOLL *et al.*, 2005). (c) Renderização de moléculas com elipsoides e cilindros adicionando sombras projetadas e *ambient occlusion* (SIGG *et al.*, 2006).



(a) (b) (c)

Fonte: (a) (GUMHOLD, 2003), (b) (STOLL *et al.*, 2005) e (c) (SIGG *et al.*, 2006).

para superfícies quadráticas como elipsoides, cilindros e cones. Ou seja, não são tão facilmente aplicáveis a surface splatting. A ideia de aplicar ray casting em superfícies quadráticas em GPU foi generalizada em (STOLL *et al.*, 2006) de forma eficiente, economizando chamadas ao fragment shader ao realizar *back-face culling* de quádricas no vertex shader usando *bounding spheres* ou *bounding tetrahedrons*.

Bolla e Narayanan (2008) apresentaram os *Algebraic Splats* – splats quadráticos ou cúbicos computados usando um procedimento de *Moving Least Squares (MLS)* (LEVIN, 2003). O objetivo do trabalho é obter boa qualidade e maior velocidade de renderização usando uma quantidade muito menor de primitivas. Apesar da semelhança com a proposta desta tese, algumas diferenças são significativas. Em Bolla e Narayanan (2008), todos os *algebraic splats* são rasterizados por ray casting no fragment shader, causando uma sobrecarga desnecessária, uma vez que os artefatos visíveis estão basicamente na silhueta do modelo. Para splats longe da silhueta, splats planos sombreados com modelo de Phong (BOTSCH *et al.*, 2004) apresentam o melhor custo-benefício. Outra diferença está na amostragem dos splats. A amostragem sugerida em Bolla e Narayanan (2008) garante uma amostragem livre de buracos dependendo de um valor heurístico escolhido pelo usuário. Nesta tese, utiliza-se a amostragem proposta por Wu e Kobbelt (2004), garantindo uma amostragem livre de buracos sem uso de parâmetro, além de adaptá-lo a arestas e cantos do modelo. Por fim, os quadric splats precisam de menos valores para serem representados. Os polinômios usados nos *algebraic splats* usam entre 6 coeficientes (quadráticas) a 15 coeficientes (quárticas). Esses valores podem ser reduzidos ao alinhar os splats com as

Figura 20 – Pipeline da detecção de silhueta em espaço de imagem proposta em (ZHANG *et al.*, 2014).

Fonte: (ZHANG *et al.*, 2014).

direções das curvaturas principais do modelo. Por esse motivo, os *quadric splats* propostos aqui utilizam apenas 4 coeficientes. Essa menor quantidade de dados, contribui para menos gasto de memória gráfica.

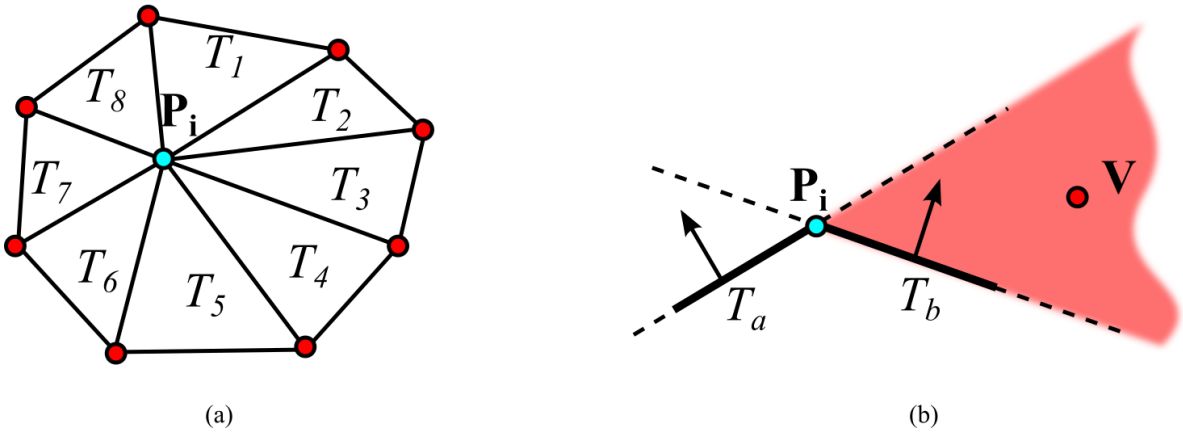
2.3 Detecção de Silhuetas

Muitos dos algoritmos de geração de silhueta existentes são baseados em malhas poligonais. Eles podem ser divididos em três grupos de acordo com onde o algoritmo detecta e desenha a silhueta: em espaço de objeto, em espaço de imagem, e em ambos os espaços (algoritmos híbridos) (HERTZMANN, 1999; ISENBERG *et al.*, 2003).

Xu e seus coautores (2004) desenvolveram um método em espaço de imagem baseando-se no algoritmo do Pintor. O método renderiza a nuvem de splats aumentando levemente o raio das amostras em uma primeira etapa. Em uma segunda etapa, o modelo é renderizado novamente com os raios originais dos splats, mas utilizando o *depth map* da etapa anterior. Dessa forma, os pixels gerados pela primeira etapa não cobertos pelos splats da segunda etapa estão na silhueta. Zhang e seus coautores (2014) renderizam duas imagens usando splatting com parâmetros diferentes e obtêm as linhas da silhueta ao aplicarem um operador de subtração Gaussiana sobre essas duas imagens (Figura 20). Ambas as técnicas operam em espaço de imagem. Apesar de serem técnicas rápidas e gerarem boas renderizações estilizadas, técnicas em espaço de imagem não são adequadas para acelerar o processo de renderização, como proposto aqui, pois é necessário detectar splats presentes na silhueta antes de rasterizá-los, ou seja, em espaço de objeto.

Uma abordagem comum para encontrar amostras presentes na silhueta de uma nuvem de pontos em espaço de objeto é a *limiarização de normais*. Nessa abordagem, um ponto $\mathbf{P}$ está na silhueta se $\mathbf{n_P} \cdot \mathbf{v_P} \in [-\varepsilon, \varepsilon]$, onde $\mathbf{n_P}$ é o vetor unitário normal à superfície em $\mathbf{P}$, $\mathbf{v_P}$ é o

Figura 21 – Detecção de pontos na silhueta proposta em (OLSON *et al.*, 2011). (a) Uma malha triangular local é construída com a vizinhança local de cada ponto da nuvem de splats. (b) Se houver pelo menos dois triângulos, um voltado e outro não, para a câmera (ponto **V**), o ponto é considerado como pertencente à silhueta.



(a) (b)

Fonte: o autor.

vetor unitário apontando para a câmera a partir do ponto **P** e ε uma determinada margem de erro. Zakaria e Seidel (2004) apresentam uma abordagem híbrida que identifica pontos na silhueta usando a limiarização de normais, renderiza-os no frame buffer, e extrai as curvas no espaço de imagem usando um processo que conecta áreas subamostradas e remove pixels em áreas sobreamostradas. Como ilustrado na Figura 4, limiarização de normais pode resultar na detecção de splats longe da silhueta e na não-detecção de splats presentes na silhueta, dependendo da curvatura local da superfície.

A silhueta de uma malha poligonal pode ser definida como o conjunto de arestas compartilhadas por faces de frente para a câmera com faces de costas para a câmera. Olson e seus coautores (2011) adaptam essa definição para nuvens de pontos ao construírem para cada ponto do modelo uma malha local com os vizinhos mais próximos. Um ponto é identificado como pertencendo a uma silhueta se há uma face de frente para a câmera e outra de costas para a câmera nos triângulos em volta do ponto em questão (Figura 21). Apesar de exigir um pré-processamento computacionalmente caro, esse método funciona em espaço de objeto e evita os problemas da limiarização de normais. Entretanto, essa é uma técnica mais adequada para nuvens de pontos puros. Splats permitem determinar muito mais informações sobre a forma local de um superfície dependendo do tipo de amostragem utilizada. Por exemplo, em uma amostragem adaptativa, splats menores normalmente localizam-se em regiões de alta curvatura. O quadric splat proposto nesta tese é uma reconstrução local ainda mais próxima da superfície representada e pode ser visto como um tipo de aproximação local assim como a malha local

utilizada em (OLSON *et al.*, 2011), mas aplicável a nuvens de splats, que tendem a ser bem menos numerosos e renderizados de forma mais eficiente do que nuvens de pontos.

3 RECORTE CURVO DE SPLATS

Um dos principais problemas encontrados pelas demais técnicas de recorte de splats é utilizar linhas retas para aproximar uma aresta localmente. Recortar splats por segmentos de reta pode ser bastante problemático se a curvatura da aresta é alta. Quando curvas são pouco amostradas em malhas, pelo menos elas são contínuas, o que não ocorre em nuvens de splats. Dessa forma, artefatos tornam-se mais visíveis nesse tipo de modelo. Estruturas mais complexas como polilinhas podem ser usadas para recortar um splat, mas a quantidade variável de dados por primitiva torna difícil a implementação destes métodos em GPU, tornando a renderização proibitivamente lenta. Utilizar uma curva para recortar um splat possui algumas vantagens sobre as abordagens anteriores:

1. o erro de aproximação do modelo em relação à superfície original é menor;
2. o recorte é suave independentemente da densidade de splats no lado oposto da aresta e da distância entre observador e modelo; e
3. a estrutura de dados para cada amostra é fixa, tornando simples a implementação em GPU.

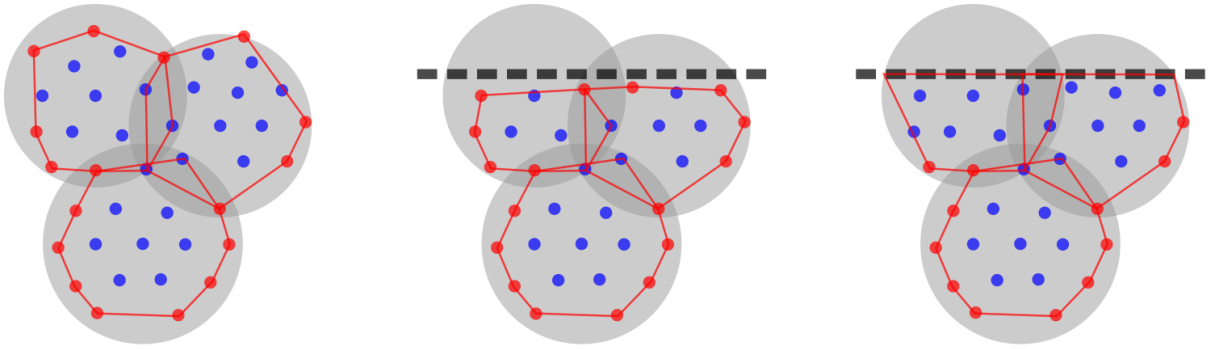
Devido às vantagens mencionadas, o recorte curvo de splats acaba por permitir que menos primitivas possam ser posicionadas nas proximidades de arestas. Por esse motivo, propõe-se uma adaptação ao processo de amostragem por clusterização na Seção 3.1. Em seguida, a curva utilizada para recortar splats é definida na Seção 3.2. A Seção 3.3 explica o processo de computação automática das curvas de recorte. E finalmente, a seção 3.4 apresenta uma implementação possível de recorte curvo com o uso de shaders.

3.1 Amostragem na proximidade de arestas

Como discutido na Seção 2.2.1, dentre os diversos fatores que influenciam a qualidade da geração de uma imagem a partir de um modelo discreto, três se destacam: a quantidade, a distribuição e a forma das amostras. Como o recorte dos splats será curvo, os algoritmos de amostragem possuem maior liberdade para amostrar splats maiores dependendo menos da curvatura da aresta e mais da curvatura da superfície onde o splat está localizado.

Independentemente do método utilizado para a geração de splats, ao final deve-se garantir que a nuvem de splats cobrirá completamente a superfície. A Figura 14 ilustra a amostragem por clusterização proposta em (WU; KOBBELT, 2004). Inicialmente, sua amostragem gera um splat para cada ponto da nuvem de pontos, e, depois, constrói um subconjunto desses splats

Figura 22 – Amostragem livre de buracos na proximidade de arestas. (a) Longe de arestas, descartar os pontos do fecho convexo dos pontos cobertos por um splat, garante uma amostragem sem buracos. (b) Na proximidade de arestas, alguns pontos sempre serão considerados não cobertos. (c) Adaptação dessa abordagem ao utilizar a linha da aresta no cálculo do fecho convexo.



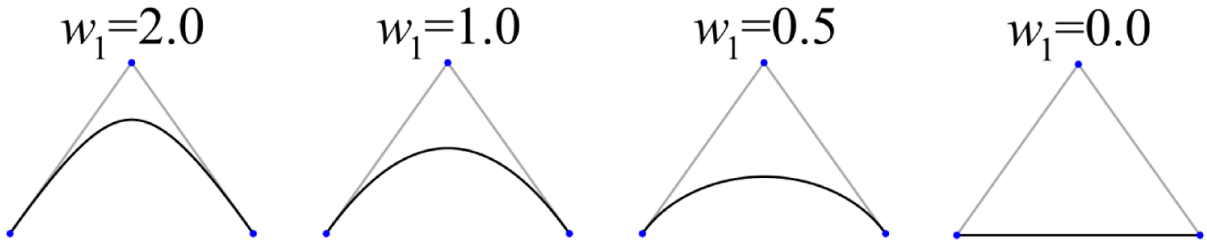
(a) (b) (c)

Fonte: o autor.

selecionando splats iterativamente até que todos os pontos da nuvem original sejam definidos como cobertos. Para garantir que não haja buracos entre os splats, os pontos sob um determinado splat, mas presentes no fecho convexo desse grupo não são denominados como cobertos pelo splat. Essa abordagem mostra-se um tanto conservadora, mas garante uma aproximação livre de buracos. Entretanto, esse método foi proposto para superfície suaves. Na proximidade de arestas pode ocorrer sobreamostragem indesejada.

Para entender esse efeito, na Figura 22 há um conjunto de pontos e três splats gerados pelo processo de amostragem: os pontos em azul são denominados *cobertos* por esse conjunto de três splats e os pontos em vermelho são denominados *não cobertos*. Na Figura 22a, não há aresta interceptando os splats. Alguns pontos pertencentes ao fecho convexo de um splat são internos a de outros splats vizinhos. Essa obrigatoriedade, garante a amostragem livre de buracos. Na Figura 22b, há uma aresta interceptando dois splats. No lado oposto da aresta não há pontos pertencentes à mesma superfície, então não são considerados pela amostragem. Logo, alguns pontos muito próximos da aresta estarão no fecho convexo de todos os splats gerados até então e serão considerados como pontos não cobertos de forma errônea. Já que o processo de amostragem só termina ao determinar todos os pontos como cobertos, splats serão adicionados utilizando esses pontos próximos de arestas como seus centros, acarretando uma sobreamostragem desnecessária nos entornos dessas regiões. Como forma de evitar esse efeito, propõe-se projetar a linha da aresta sobre os splats e utilizar essa linha no cálculo dos fechos convexos. Com essa adição, pontos próximos a arestas passam a ser cobertos por splats nos

Figura 23 – Diferentes curvas formadas pela variação do peso aplicado ao ponto de controle central. Os pesos associados com os pontos de controle das extremidades foram fixados em 1.0.



Fonte: Disponível em: <https://en.wikipedia.org/wiki/Bézier_curve>. Acesso em: 12 fev 2020.

arredores, evitando a sobre-amostragem, mas garantindo a ausência de buracos entre os splats (Figura 22c).

3.2 Definição do Recorte Curvo

Os splats que cruzam uma aresta do modelo precisam ser recortados. Cada splat nessas regiões terá associada aos seus dados uma curva definida sob seu sistema de coordenadas local. A Seção 3.2.1 define a curva de recorte utilizada nesse método e a Seção 3.2.2 define os pontos internos do splat localizados na área de recorte delimitada pela curva de recorte.

3.2.1 Curva de Recorte

As curvas propostas para recortar os splats próximos de arestas são as curvas de Bézier racionais. Essas curvas possuem pesos ajustáveis para cada ponto de controle e permitem aproximações melhores a curvas arbitrárias, incluindo seções cônicas como arcos circulares, o que é bastante útil em modelos de peças mecânicas e de engenharia. Uma curva desse tipo pode ser definida como

$$\mathbf{B}(t) = \frac{\sum_{i=0}^{n} b_{i,n}(t) w_i \mathbf{P}_i}{\sum_{i=0}^{n} b_{i,n}(t) w_i} \tag{3.1}$$

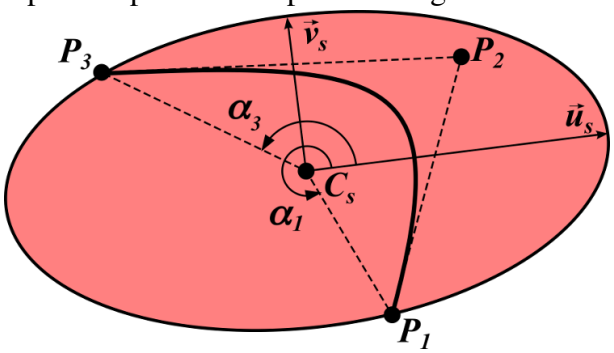
onde

$$b_{i,n}(t) = \binom{n}{i} t^i (1-t)^{n-i}. \tag{3.2}$$

w_i e $b_{i,n}(t)$ são, respectivamente, os pesos e as *blending functions* associadas com cada ponto de controle, nesse caso, denominadas polinômios de Bernstein.

Para permitir maior eficiência em renderizar os splats em GPU, as curvas precisam ser definidas com uma quantidade fixa de dados. Dessa forma, escolhemos apenas uma determinada

Figura 24 – Definição da curva de recorte: uma curva Bézier racional com três pontos de controle descritos sobre o sistema de coordenadas local do splat. As extremidades são fixadas na borda do splat e representadas por seus ângulos centrais.



Fonte: o autor

classe destas curvas, as curvas Bézier racionais com três pontos de controle, $\mathbf{P_1}$, $\mathbf{P_2}$ e $\mathbf{P_3}$. Os pesos das extremidades são fixados em 1.0 e o peso w do ponto de controle central sendo maior que 0.0 (Figura 23). Com essa simplificação, a Equação 3.1 pode ser reescrita como

$$\mathbf{B}(t) = \frac{(1-t)^2\mathbf{P_1} + 2t(1-t)w\mathbf{P_2} + t^2\mathbf{P_3}}{(1-t)^2 + 2t(1-t)w + t^2}. \tag{3.3}$$

Seja S um splat elíptico com centro $\mathbf{C_s}$ e eixos ortogonais $\mathbf{u_s}$ e $\mathbf{v_s}$. Seu sistema de coordenadas local pode ser definido pelos vetores $\mathbf{i_s}$, $\mathbf{j_s}$ e $\mathbf{k_s}$ como

$$\begin{aligned} \mathbf{i_s} &= \mathbf{u_s}/||\mathbf{u_s}||, \\ \mathbf{j_s} &= \mathbf{v_s}/||\mathbf{v_s}||, \text{ e} \\ \mathbf{k_s} &= (\mathbf{u_s} \times \mathbf{v_s})/||\mathbf{u_s} \times \mathbf{v_s}||. \end{aligned} \tag{3.4}$$

A curva de recorte será definida sobre o plano $\mathbf{i_s} \times \mathbf{j_s}$ desse sistema de coordenadas local do splat. As extremidades da curva serão fixadas sobre a borda do splat. Dessa forma, os pontos $\mathbf{P_1}$ e $\mathbf{P_3}$ podem ser representados pelos ângulos centrais α_1 e α_3, respectivamente, avaliados no sentido anti-horário a partir de $\mathbf{i_s}$ (Figura 24). Essa representação das extremidades é usada para reduzir a quantidade de valores de ponto flutuante armazenados e enviados aos shaders. Entretanto, as coordenadas de $\mathbf{P_1}$ e $\mathbf{P_3}$ serão calculadas no sistema de coordenadas local do splat ainda no *vertex shader* da forma

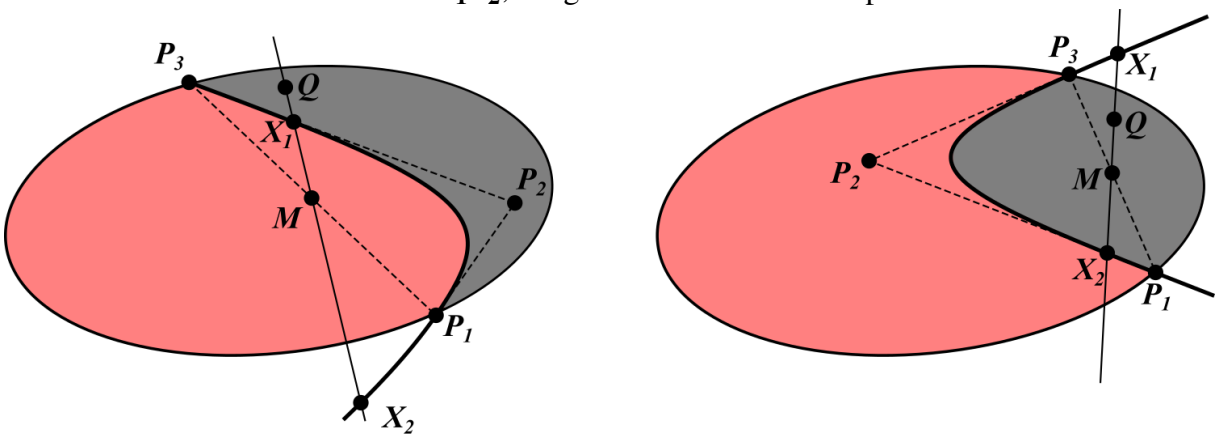
$$\mathbf{P_i} = \begin{pmatrix} r(\alpha_i)cos\alpha_i \\ r(\alpha_i)sin\alpha_i \end{pmatrix} \tag{3.5}$$

onde

$$r(\alpha) = \frac{||\mathbf{u_s}||||\mathbf{v_s}||}{\sqrt{(||\mathbf{v_s}||cos\alpha)^2 + (||\mathbf{u_s}||sin\alpha)^2}}. \tag{3.6}$$

é a distância do centro do splat elíptico a um ponto na sua borda calculado a partir do ângulo central.

Figura 25 – Definição da área de recorte, ilustrada em cinza. (a) Se $\mathbf{P_3}$ estiver a esquerda da semirreta $\overrightarrow{\mathbf{P_1P_2}}$, a região externa delimitada pela curva é recortada. (b) Se $\mathbf{P_3}$ estiver a direita da semirreta $\overrightarrow{\mathbf{P_1P_2}}$, a região interna delimitada pela curva é recortada.



Fonte: o autor.

3.2.2 Área de Recorte

Em Zwicker *et al.* (2004), a ordem dos vértices do segmento de reta utilizado para recortar o splat definia qual lado do segmento deveria ser recortado do splat. De forma análoga, a ordem dos pontos de controle definem a área do splat a ser removida a partir da curva definida na Equação 3.3. A área de recorte é definida como sendo a área do splat limitada entre o arco elíptico da borda do splat iniciando em $\mathbf{P_1}$ e terminando em $\mathbf{P_3}$ no sentido anti-horário e a própria curva de recorte (Figura 25).

A curva de recorte divide o plano do splat em dois semi-espaços: um semi-espaço convexo, denominado aqui por *região interna*, e um semi-espaço côncavo, denominado por *região externa*. Se ao longo da polilinha de controle $\mathbf{P_1} - \mathbf{P_2} - \mathbf{P_3}$, saindo de $\mathbf{P_1}$ para $\mathbf{P_3}$, uma curva à esquerda for feita em $\mathbf{P_2}$, a região a ser recortada é a região externa (Figura 25a). Se, por outro lado, a curva em $\mathbf{P_2}$ for à direita, então a região a ser recortada é a região interna (Figura 25b).

Com essa classificação definida, precisa-se determinar em qual região um determinado ponto do splat está. Seja $\mathbf{Q}$ um ponto qualquer sob o plano do splat. Como o peso w sempre será considerado positivo, pode-se afirmar que o ponto médio $\mathbf{M}$ do segmento $\overline{\mathbf{P_1P_3}}$ sempre estará na região interna da curva. Dessa forma, se o segmento $\overline{\mathbf{QM}}$ interceptar a curva de recorte, significa que $\mathbf{Q}$ está na região externa (Figura 25a). Caso contrário, o ponto $\mathbf{Q}$ está na região interna (Figura 25b). A reta paramétrica que passa através dos pontos $\mathbf{Q}$ e $\mathbf{M}$ pode ser definida

como

$$\mathbf{R}(t) = \mathbf{Q} + t(\mathbf{M} - \mathbf{Q}). \tag{3.7}$$

Combinando as equações 3.3 e 3.7, diferenciando os parâmetros da reta e da curva por t_r e t_b, respectivamente, obtém-se um sistema de duas equações e duas variáveis. Os valores de t_b nos pontos de interseção são dados por

$$t_b = \frac{a \pm \sqrt{a^2 - bc}}{c}, \tag{3.8}$$

onde:

$$\begin{aligned} a &= \mathbf{d_2} \cdot [\mathbf{P_1} - \mathbf{Q} + w(\mathbf{Q} - \mathbf{P_2})] \\ b &= \mathbf{d_2} \cdot (\mathbf{P_1} - \mathbf{Q}) \\ c &= \mathbf{d_2} \cdot [\mathbf{P_1} + \mathbf{P_3} + 2(w\mathbf{Q} - \mathbf{Q} - w\mathbf{P_2})] \\ \mathbf{d} &= \mathbf{M} - \mathbf{Q} \\ \mathbf{d_2} &= \begin{pmatrix} d_y \\ -d_x \end{pmatrix} \end{aligned} \tag{3.9}$$

Dessa forma, os pontos $\mathbf{X_1} = \mathbf{B}(t_{b_1})$ e $\mathbf{X_2} = \mathbf{B}(t_{b_2})$ são os pontos de interseção entre a reta e a curva. Ambos os pontos não precisam ser computados usando a Equação 3.3, porque a única que importa é aquela que é interna ao splat. A reta $\mathbf{R}(t)$ certamente cruza a curva de recorte na seção interna ao splat, que possui parâmetros t_b no intervalo $[0,1]$. Seja t_b^* o parâmetro da curva de recorte que obedece a essa propriedade e $\mathbf{X}^* = \mathbf{B}(t_b^*)$. Pode-se computar o parâmetro da reta t_r^* do ponto de interseção correspondente ao substituir $\mathbf{X}^*$ na Equação 3.7. Dessa forma, há duas possibilidades:

- se o recorte for da região interna (convexa), o segmento não deve interceptar a curva, então $t_r^* \notin [0,1]$; e
- se o recorte for da região externa (côncava), o segmento deve interceptar a curva, então $t_r^* \in [0,1]$.

Uma propriedade importante das curvas de Bézier racionais utilizadas para recorte é ilustrada na Figura 23: quanto maior for o peso w do ponto de controle central, mais a curva tende a se aproximar do polígono de controle; e quanto menor for o peso w, mais a curva tende a ser um segmento de reta entre os pontos de controle extremos. A Seção 3.3 mostra que a curva de recorte é computada por um método de aproximação, logo, por motivos de erros numéricos, w nunca será realmente zero. Algo similar acontece nos cantos do modelo, pois eles nunca serão

representados apropriadamente mesmo com valores muito altos em w. Dessa forma, o usuário pode apresentar limiares mínimo, w_{min}, e máximo, w_{max}, para o peso da curva de recorte w. Assim há três possibilidades:

1. se $w < w_{min}$, o recorte é feito pela reta definida pelos pontos $\mathbf{P_1}$ e $\mathbf{P_3}$;
2. se $w > w_{max}$, o recorte é feito pelo polígono de controle; e
3. caso contrário, o recorte é feito pela curva de recorte.

3.3 Computação das Curvas de Recorte

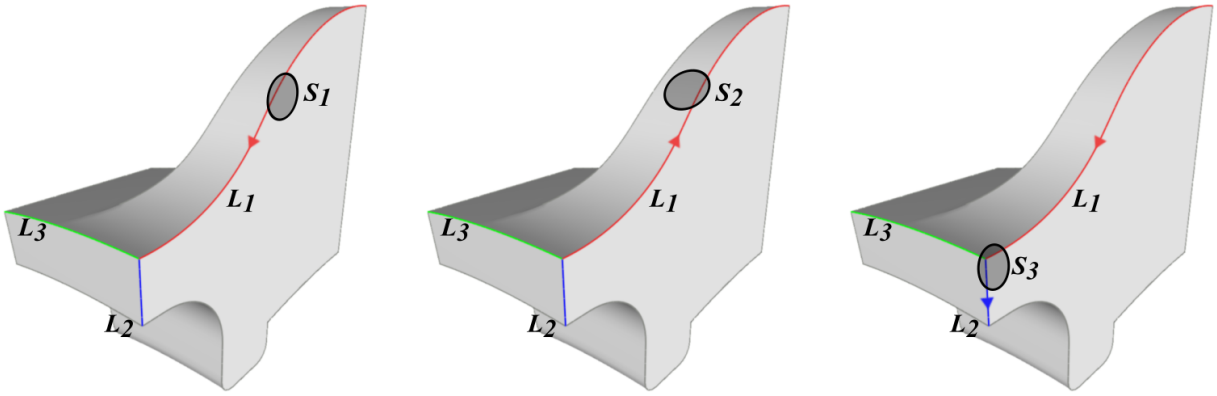
Quando um splat é interceptado por uma aresta do modelo, valores apropriados para os três pontos de controle, $\mathbf{P_1}$, $\mathbf{P_2}$ e $\mathbf{P_3}$, e para o parâmetro w devem ser definidos para obter a melhor aproximação possível do splat à área local do modelo. Deixar para o usuário especificar esses valores seria impraticável. Dessa forma, esta seção discute como determinar os atributos da curva automaticamente.

As curvas das arestas podem ser definidas desde polilinhas a *NURBS (Non Uniform Rational Basis Spline)* e podem ser calculadas usando algum algoritmo de detecção de arestas (DANIELS *et al.*, 2008) ou atribuídas explicitamente pelo usuário (ZHANG; KAUFMAN, 2007). A curva de recorte busca aproximar pedaços dessas arestas para recortar os splats apropriadamente. Por esse motivo, a Seção 3.3.1 mostra como computar apropriadamente a interseção das arestas do modelo com o splat que será levada em consideração para aproximá-lo. Os parâmetros da curva de recorte (Equação 3.3) são determinados através de um procedimento de otimização a partir de uma aproximação inicial (Seção 3.3.2) e minimiza sua distância às arestas usando uma função de erro (Seção 3.3.3).

3.3.1 Interseção splat-arestas

Antes de computar a curva de recorte, identifica-se as arestas que cruzam o splat e qual seu formato com relação ao splat. As curvas das arestas são definidas em espaço de objeto, um espaço $3D$. Entretanto, as curvas de recorte são definidas em um sistema de coordenadas local do splat, um espaço $2D$. Um bom exemplo para verificar as relações entre curva da aresta e as curvas de recorte dos splats pode ser visto na Figura 26. Os splats S_1 e S_2 são próximos e cruzam a mesma aresta, L_1, mas devem possuir curvas de recorte distintas. No splat S_2, o recorte deve ser reto, porque a superfície no outro lado da aresta L_1 é plana. Entretanto, no splat S_1, o

Figura 26 – Observações quanto à linha da aresta utilizada para computar a curva de recorte. (a-b) A partir de uma mesma aresta podem decorrer diferentes tipos de recorte. (c) Nem todas as arestas em contato com um splat participam do cálculo da curva de recorte.



(a) (b) (c)

Fonte: o autor.

recorte deve ser curvo. Por este motivo, a curva a ser aproximada pela curva de recorte é uma projeção da linha da aresta sobre o plano do splat.

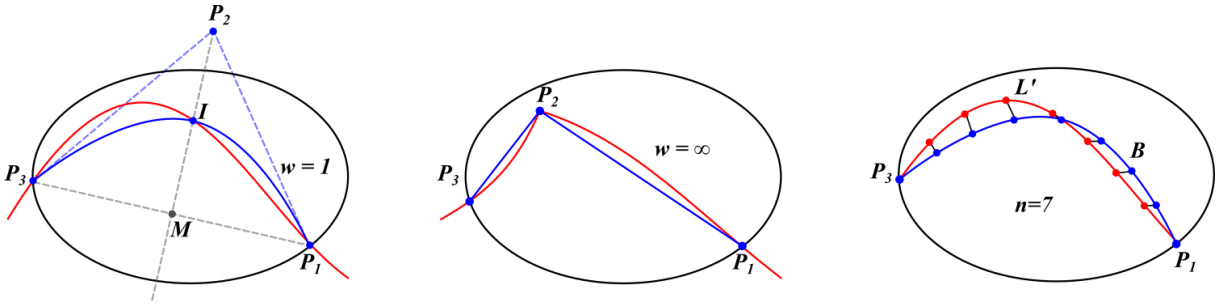
Outro detalhe que deve ser considerado durante a implementação do recorte de splats diz respeito à orientação da aresta. A área de recorte definida na Seção 3.2.2 depende da ordem dos pontos de controle. Por esse motivo, a orientação da linha da aresta L_1 deve ser distinta para os splats S_1 (Figura 26a) e S_2 (Figura 26b).

Essa etapa requer algumas precauções para evitar erros grosseiros nas etapas posteriores. Nos cantos, um splat entra em contato com diversas arestas. Na Figura 26, o splat S_3 tem contato com três arestas diferentes: L_1, L_2 e L_3. A aresta L_3 deve ser descartada da computação da curva de recorte, uma vez que as arestas presentes nas bordas locais da superfície à qual S_3 pertence são apenas L_1 e L_2. Esta distinção é feita usando o produto escalar entre o vetor que aponta na direção local da aresta e o vetor normal do splat. Se o produto local é próximo de zero, então a aresta é próxima do plano do splat e é mantida na computação da curva de recorte, caso contrário, a aresta está trespassando o splat, então ela é descartada do processo.

3.3.2 Curva de Recorte Inicial

Assim como qualquer outro método de otimização, uma boa estimativa inicial reduz a complexidade do procedimento. Seja L' a projeção da linha da aresta computada seguindo as orientações da seção anterior. A definição da curva inicial para o processo de adaptação primeiro computa as interseções de L' com as bordas do splat. Sob essas interseções, os pontos

Figura 27 – Adaptação da curva de recorte, em azul, à linha local da aresta, em vermelho. (a) Curva inicial definida como uma parábola simétrica na qual o ponto médio intercepta a linha da aresta. (b) Quando um canto está presente na linha da aresta, o recorte é feito pelo recorte reto do polígono de controle. (c) A função-erro é definida como a distância média entre pontos igualmente espaçados sobre a curva de recorte e a linha da aresta.



Fonte: o autor.

de controle $\mathbf{P_1}$ e $\mathbf{P_3}$ são fixados e mantidos durante todo o resto do processo. Seja os pontos $\mathbf{M}$ e $\mathbf{I}$ tais que $\mathbf{M}$ é o ponto médio de $\overline{\mathbf{P_1P_3}}$ e $\mathbf{I}$ é o ponto de interseção entre a mediatriz de $\overline{\mathbf{P_1P_3}}$ e a curva L'. A curva de recorte inicial proposta é a parábola que passa através dos pontos $\mathbf{P_1}$, $\mathbf{I}$ e $\mathbf{P_3}$ (Figura 27a).

A definição dos valores iniciais de $\mathbf{P_2}$ e w exploram as seguintes propriedades de curvas Bézier racionais:

- Quando todos os pesos de uma curva Bézier racional são iguais, a curva é igual a uma curva Bézier clássica. Como são três pontos de controle, a curva é uma parábola;
- Quando os pontos de controle formam um triângulo isósceles no qual a base é o segmento conectando as duas extremidades da curva, a bissetriz da base é também um eixo de simetria da parábola;
- Quando a curva é simétrica, então seu ponto médio pertence à mediatriz da base do triângulo isósceles usado como polígono de controle. Esse ponto também é o ponto médio da altura do triângulo isósceles.

Dessa forma, o peso inicial e o ponto de controle central da curva de recorte são definidos como:

$$w = 1.0 \;\; ; \;\; \mathbf{P_2} = 2\mathbf{I} + \mathbf{M}. \tag{3.10}$$

Caso L' possua um vértice, então o splat está próximo de um canto do modelo. Nesse caso, não há curva Bézier racional que apresente boa representação dessa região. Por esse motivo, utiliza-se uma aproximação linear ao recortar o splat utilizando seu polígono de controle. Para este fim, o ponto de controle $\mathbf{P_2}$ é posicionado sob o vértice de L' e o peso w é definido arbitrariamente com um valor superior ao valor w_{max}, como discutido na Seção 3.2.2 (Figura 27b).

3.3.3 Função-Erro de Aproximação

O processo de adaptação necessita de uma função de erro que defina quão próxima a curva de recorte B está da projeção da linha da aresta sobre o splat, L'. Se fosse possível encontrar um eixo de coordenadas tal que ambas as curvas L' e B pudessem ser convertidas em funções $l(x)$ e $b(x)$, respectivamente, uma boa função de erro poderia ser definida como

$$e = \frac{1}{x_2 - x_1} \int_{x_1}^{x_2} |l(x) - b(x)| dx. \tag{3.11}$$

Esse valor corresponde à soma das distâncias entre os pontos da função $b(x)$ e os pontos correspondentes em $l(x)$.

Entretanto, nem sempre é possível encontrar um sistema de coordenadas comum no qual ambas as curvas se comportem como funções. Por esse motivo, é utilizada uma abordagem discreta ao invés de contínua. Sejam dois conjuntos de pontos espaçados uniformemente em ambas as curvas. A função de erro é definida como

$$e = \frac{1}{n} \sum_{i=1}^{n} |\mathbf{L_i} - \mathbf{B_i}|, \tag{3.12}$$

onde $\mathbf{L_i}$ são os pontos distribuídos sobre L', $\mathbf{B_i}$ são os pontos distribuídos sobre B e n é a quantidade de pontos (Figura 27c).

Os pontos sobre a curva de recorte B não podem ser espaçados utilizando o parâmetro da curva, então uma parametrização em comprimento de arco é computada previamente. Essa curva não possui um ponto de inflexão, então, uma tabela simples que correlaciona valores dos parâmetros com comprimentos do arco pode ser utilizada para posicionar pontos igualmente espaçados sobre B. A amostragem uniforme de pontos sobre a curva L' depende da representação da linha da aresta.

Definida essa função de erro, um processo de minimização com relação aos parâmetros $\mathbf{P_2}$ e w é realizado. Esse procedimento é computacionalmente caro, entretanto, é uma etapa feita em pré-processamento, não afetando o tempo de renderização de modelos estáticos e, normalmente, poucos ciclos de otimização são necessários.

3.4 Renderização do Recorte Curvo

A implementação em GPU é feita através de *vertex shaders* e *fragment shaders*. Todas as operações comuns a todos os fragmentos devem ser computadas no vertex shader e enviadas ao fragment shader. Por exemplo, a conversão dos ângulos centrais em pares de coordenadas das extremidades da curva de recorte, mostrada nas equações 3.5 e 3.6, é feita no vertex shader porque é um atributo comum a todos os fragmentos gerados por um determinado splat. Isso evita cálculos repetidos para cada fragmento no fragment shader.

Todas as operações de recorte são feitas no fragment shader. A técnica proposta de recorte necessita da posição exata sobre o splat para cada pixel. Dessa forma, o splat é rasterizado usando o método de *ray casting* (BOTSCH *et al.*, 2005). Após o ray casting, o ponto 3D estará disponível, mas todas as informações sobre o recorte estão no sistema de coordenadas local do splat $\mathbf{i_s} \times \mathbf{j_s}$. Dessa forma, o ponto, $\mathbf{Q}$, determinado pelo ray casting é descrito no sistema de coordenadas local do splat.

A Figura 28 descreve os algoritmos da técnica de recorte proposta. O algoritmo `clipping` decide que tipo de recorte será realizado em um determinado splat. Entre as linhas 2-7, é determinado se o ponto $\mathbf{P_3}$ está à esquerda ou à direita da semirreta definida pelo vetor $\overrightarrow{\mathbf{P_1P_2}}$. Se estiver à direita, a área de recorte é convexa. Se estiver à esquerda, a área de recorte é côncava. Se os três pontos forem quase colineares ou o peso w for menor que o limiar mínimo w_{min}, o recorte pode ser feito pela reta que conecta $\mathbf{P_1}$ a $\mathbf{P_3}$ (linha 7). O método `clip_line` retorna `true` se o ponto $\mathbf{Q}$ estiver do lado direito do vetor $\overrightarrow{\mathbf{P_1P_3}}$ e retorna `false` caso contrário. Caso a curva de recorte tenha um peso w acima de w_{max}, o recorte é feito usando somente o polígono de controle, representando assim o canto (linha 9). Por fim, caso nenhum desses casos especiais ocorra, o recorte curvo é computado (linha 10).

O algoritmo `clipping_corner` descreve o recorte realizado em splats próximos a cantos. Nas linhas 2 e 3, é determinado se o ponto seria recortado pelos vetores $\overrightarrow{\mathbf{P_1P_2}}$ e $\overrightarrow{\mathbf{P_2P_3}}$ utilizando o método `clip_line`. Se a área de recorte é convexa, o fragmento é descartado somente quando ele está na área de recorte de ambos os vetores (linha 5). Se a área de recorte é

Figura 28 – Algoritmos da técnica de recorte curvo. O algoritmo `clipping` decide qual método de recorte é o melhor para cada situação. O algoritmo `clipping_corner` descreve o método de recorte realizado em splats localizados próximos a cantos, onde os fragmentos são comparados ao polígono de controle. O algoritmo `clipping_curve` descreve o método de recorte realizado pela curva de Bézier racional.

```
Algorithm clipping
Input: point2D: Q

 1. clipped ← false
 2. v21 ←  P2 - P1
 3. t21 ← (v21.y , -v21.x)
 4. prod ← dot(P3 - P1,t21)
 5. if prod > 0 then clipType ← CONVEX
 6. if prod < 0 then clipType ← CONCAVE
 7. if prod ≈ 0 or w < wmin then clipped ← clip_line(Q, P1, P3)
 8. else
 9.     if w > wmax then clipped ← clip_corner(Q, clipType)
10.     else             clipped ← clip_curve(Q, clipType)
11. if(clipped) discard

Algorithm clip_corner
Input: point2D Q, type clipType
Output: boolean clipped

 1. clipped ← false
 2. clipped1 ← clip_line(Q, P1, P2)
 3. clipped2 ← clip_line(Q, P2, P3)
 4. if clipType = CONVEX then
 5.     if clipped1 and clipped2 then clipped ← true
 6. if clipType = CONCAVE then
 7.     if clipped1 or clipped2 then clipped ← true

Algorithm clip_curve
Input: point2D Q, type clipType
Output: boolean clipped

 1. clipped ← false
 2. M ← 1/2(P1+P3)
 3. d ← Q - M
 4. lineQuery ← line(M, d)
 5. tb ← intersection(lineQuery, clipCurve)
 6. X  ← pointOnClipCurve(tb)
 7. tr ← (Xb.x - Q.x)/d.x;
 8. if clipType = CONVEX then
 9.     if tr < 0 or tr > 1 then clipped ← true
10. if clipType = CONCAVE then
11.     if tr > 0 and tr < 1 then clipped ← true
```

Fonte: o autor.

côncava, o fragmento é descartado quando ele estiver na área de recorte de pelo menos um dos vetores (linha 7).

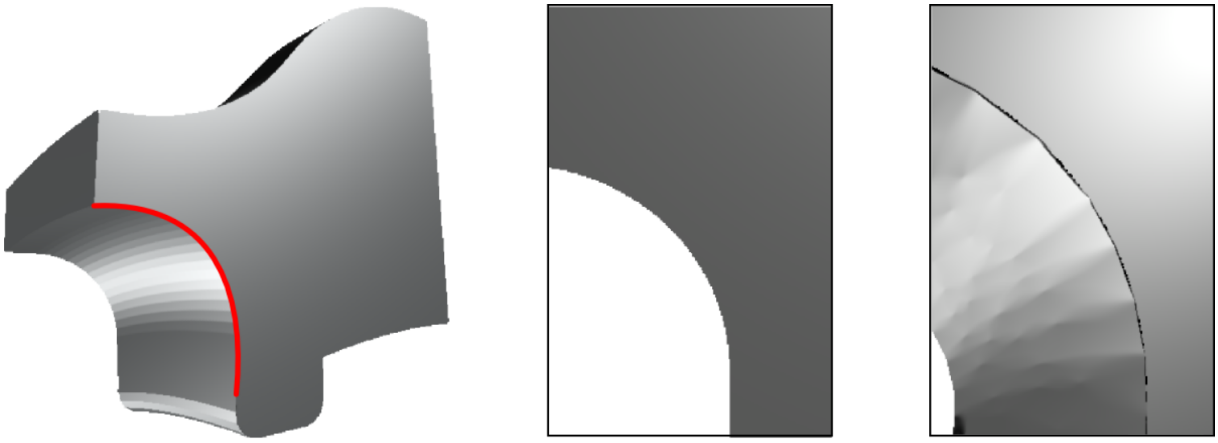
O algoritmo `clipping_curve` implementa a técnica descrita na Seção 3.2.2. clipCurve é a curva de recorte descrita na Equação 3.3 e lineQuery é a reta definida na Equação 3.7 utilizada para verificar se o ponto **Q** deve ser recortado ou não. Na linha 5, O método `intersection` retorna o parâmetro da curva $t_b^* \in [0,1]$ do ponto de interseção entre a curva de recorte B e reta de teste r através das Equações 3.8 e 3.9. A partir desse parâmetro, computa-se o parâmetro da reta t_r^* do ponto de interseção (linhas 6 e 7). Caso o recorte desejado seja convexo, o ponto **M** pertence à área de recorte, ou seja, descarta-se o ponto **Q** quando o segmento $\overline{\mathbf{MQ}}$ não toca a curva (linha 9). Caso o recorte desejado seja côncavo, o ponto **M** não pertence à área de recorte, ou seja, descarta-se o ponto **Q** quando o segmento $\overline{\mathbf{MQ}}$ toca a curva (linha 11).

3.5 Considerações Finais

Esse capítulo apresentou a técnica de recorte de splats através de curvas racionais Bézier. Inicialmente, foi definida a classe de curvas utilizadas e como representá-las no sistema de coordenadas local do splat utilizando a menor quantidade de informação possível. A área do splat a ser recortada pela curva foi definida dependendo de sua orientação. Após essas definições, é mostrado como as curvas de recorte são computadas automaticamente a partir de um conjunto pré-computado de linhas retornado por algum método de detecção de arestas. Finalmente, uma possível implementação em GPU é descrita para assegurar a aplicabilidade da técnica.

Apesar da técnica de recorte curvo apresentar uma representação de arestas curvas com maior fidelidade, ela ainda se trata de uma aproximação discreta, ou seja, depende da quantidade de elementos para ser fidedigna ao modelo. Entretanto a quantidade de splats pode ser consideravelmente menor. Uma dificuldade na representação de arestas que não pode ser coberta pelos recortes curvos são as junções entre os splats recortados. Isso se deve ao fato de os splats ainda serem planos. Ou seja, os splats presentes do outro lado de uma aresta curva estão representando uma superfície curva com um conjunto de amostras lineares. Observe o caso ilustrado na Figura 29). A aresta destacada afeta splats presentes em uma superfície plana e splats presentes em uma superfície curva de formas diferentes. Projetar a aresta nos splats presentes na área plana acarretará em uma curva e a técnica de recorte curvo adapta corretamente os splats a ela. Projetar a aresta nos splats presentes na área curva acarretará a obtenção de uma reta e a técnica de recorte plano adapta corretamente os splats a ela. Entretanto, as junções não

Figura 29 – Contraste entre recorte curvo e splat plano. (a) A aresta curva está destacada em vermelho. (b) Os splats presentes na superfície plana são fielmente adaptados à aresta através do recorte curvo proposto. (c) Entretanto, artefatos podem ser notados na conexão das superfícies porque os splats planos representam uma superfície curva, ou seja, aproximações lineares da aresta curva.



(a) (b) (c)

Fonte: o autor.

são perfeitas devido à natureza plana dos splats. Esse problema pode ser resolvido ao aplicar, nas regiões nos entornos de arestas, splats curvos, descritos no próximo capítulo.

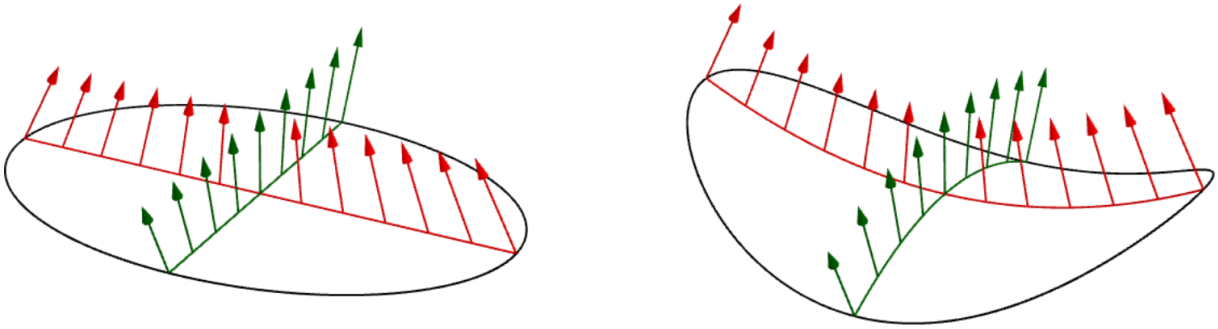
4 QUADRIC SPLATS

A técnica de surface splatting apresenta uma boa qualidade de renderização devido à utilização de filtros passa-baixa em espaço de objeto e em espaço de imagem que permitem uma transição suave entre duas amostras discretas do modelo. Quando os cálculos de sombreamento são realizados utilizando apenas o centro de cada splat e sua cor é combinada com as amostras vizinhas através da técnica de surface splatting, tem-se uma renderização similar àquela alcançada pelo sombreamento de Gouraud porque transições mais abruptas de iluminação, especialmente em efeitos especulares, são suavizadas pelos filtros passa-baixa da técnica de surface splatting (Figura 18). Tendo isso em vista, as técnicas que incorporam mapas de normais ao splat permitiram obter renderizações com o mesmo grau de realismo que as imagens obtidas com sombreamento Phong. Além do mais, com a capacidade de programação das placas gráficas atuais, obtém-se uma forma bastante eficiente de gerar renderizações de altíssima qualidade com alta taxa de frames por segundo e baixo consumo de memória, sem a necessidade de gerar uma malha poligonal.

Como ilustrado no início dessa tese, o sombreamento não consegue esconder a forma plana dos splats em regiões próximas das silhuetas (Figura 3). Outra região problemática para splats planos é o entorno de arestas curvas, pois mesmo os splats sendo recortados apropriadamente, sua natureza plana é perceptível nessas regiões onde splats não se combinam com seus vizinhos (Figura 29). Esta tese propõe o uso de splats curvos. Essas primitivas têm forma de uma superfície quadrática local e serão denominadas *quadric splats*. A representação de uma superfície por quadric splats permite uma melhor aproximação da superfície representada sem excessiva discretização: por exemplo, quadric splats podem ser maiores em regiões planas, mas também em regiões onde a curvatura é razoavelmente constante. Isso dá liberdade para escolher splats maiores, acelerando o processo de amostragem por agrupamento e reduzindo o consumo de memória gráfica para armazenar o modelo. Outra vantagem é a detecção de silhueta de forma mais precisa sem a necessidade de consulta a amostras vizinhas, permitindo detecções de silhueta eficientes em GPU ainda na etapa de processamento de vértices (*vertex shader*).

As próximas seções definem o *quadric splat* (Seção 4.1), como os quadric splats podem ser gerados sob uma nuvem de pontos (Seção 4.2), a rasterização desses quadric splats em GPU (Seção 4.3) e a detecção desses quadric splats na silhueta (Seção 4.4).

Figura 30 – O mapa de normais usado em *Phong Splatting* define todos os vetores normais de uma superfície quadrática definida sobre o sistema de coordenadas local do splat. (a) Flat splat. (b) Quadric splat.



(a) (b)

Fonte: o autor.

4.1 Definição do *Quadric Splat*

O método *Phong Splatting* (BOTSCH *et al.*, 2004) atribui explicitamente uma função vetorial linear de vetores normais e a utiliza no sombreamento de cada splat ao invés de utilizar um vetor normal constante (Figura 30a). Essa mesma função vetorial é utilizada para definir a superfície quadrática sobre a qual o quadric splat é rasterizado.

Seja S um splat elíptico com centro $\mathbf{C_s}$ e eixos ortogonais $\mathbf{u_s}$ e $\mathbf{v_s}$. Seu sistema de coordenadas local pode ser definido pelos vetores $\mathbf{i_s}$, $\mathbf{j_s}$ e $\mathbf{k_s}$ como

$$\begin{aligned} \mathbf{i_s} &= \mathbf{u_s}/||\mathbf{u_s}||, \\ \mathbf{j_s} &= \mathbf{v_s}/||\mathbf{v_s}||, \text{ e} \\ \mathbf{k_s} &= (\mathbf{u_s} \times \mathbf{v_s})/||\mathbf{u_s} \times \mathbf{v_s}||. \end{aligned} \tag{4.1}$$

O mapa de normais do splat, definido sobre o plano $\mathbf{i_s} \times \mathbf{j_s}$ do sistema de coordenadas local do splat, é especificado de tal forma que o vetor não normalizado, perpendicular ao splat em um determinado ponto $\mathbf{Q} \in S$ é descrito como

$$\mathbf{n_s}(u,v) = \begin{pmatrix} \bar{n}_{us} + \alpha_s u \\ \bar{n}_{vs} + \beta_s v \\ 1 \end{pmatrix}^T \begin{pmatrix} \mathbf{i_s} \\ \mathbf{j_s} \\ \mathbf{k_s} \end{pmatrix}, \tag{4.2}$$

onde u e v são quaisquer coordenadas locais referentes aos eixos $\mathbf{i_s}$ e $\mathbf{j_s}$, respectivamente. Dessa forma, os vetores $\mathbf{n_s}(u,v)$ são cópias inclinadas do vetor normal definido para o centro do splat. A inclinação é calculada como uma função definida sobre as coordenadas (u,v).

A Equação 4.2 é uma função bilinear. Os vetores $\mathbf{n_s}(u,v)$ são perpendiculares a uma superfície quadrática (Figura 30b). Como os vetores normais são deslocados ao longo dos eixos

do splat, a superfície quadrática definida sobre o sistema de coordenadas local do splat tem a forma:

$$w(u,v) = au^2 + bv^2 + cu + dv. \tag{4.3}$$

Os coeficientes da Equação 4.3 podem ser calculados a partir dos termos da Equação 4.2 como descrito a seguir:

$$\begin{aligned} a &= -\alpha_s/2, \\ b &= -\beta_s/2, \\ c &= -\bar{n}_{us}, \text{ e} \\ d &= -\bar{n}_{vs}. \end{aligned} \tag{4.4}$$

Dessa forma, todos os pontos $\mathbf{Q} \in S$ descritos em coordenadas locais do splat possuem a forma:

$$\mathbf{Q}(u,v) = \begin{pmatrix} u \\ v \\ w(u,v) \end{pmatrix}^T \begin{pmatrix} \mathbf{i_s} \\ \mathbf{j_s} \\ \mathbf{k_s} \end{pmatrix}. \tag{4.5}$$

Ou seja, a coordenada w é computada em função das outras duas. Como o formato do splat plano é uma elipse sobre o plano $\mathbf{i_s} \times \mathbf{j_s}$, então o quadric splat pode ser visto como a projeção dessa elipse sobre a superfície quadrática definida na Equação 4.3 na direção do vetor $\mathbf{k_s}$. Isso permite, por exemplo, manter procedimentos como a remoção de fragmentos externos ao splat pela condição

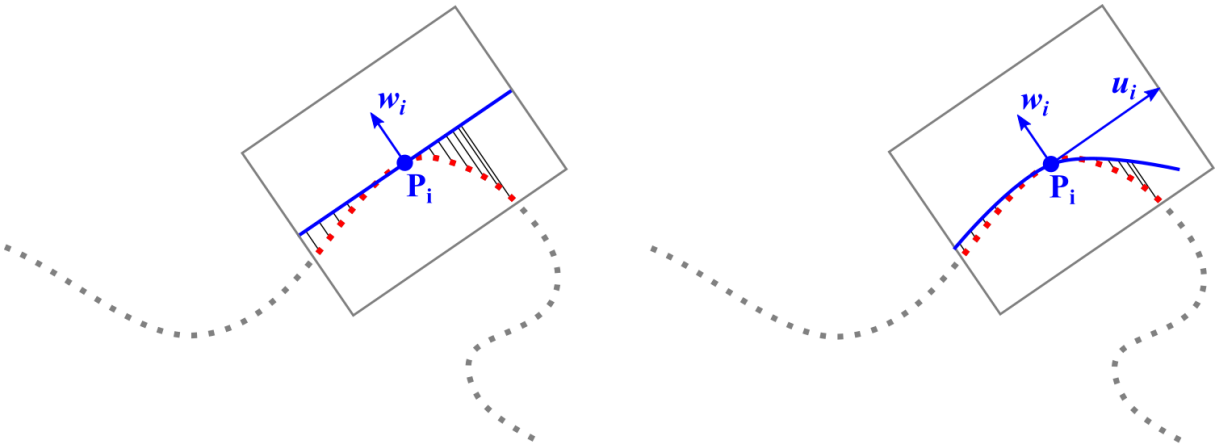
$$\left(\frac{u}{||\mathbf{u_s}||}\right)^2 + \left(\frac{v}{||\mathbf{v_s}||}\right)^2 \leq 1, \tag{4.6}$$

assim como o recorte de splats descrito no Capítulo 3 inalterados, uma vez que eles são descritos apenas sobre as coordenadas u e v.

4.2 Amostragem de *Quadric Splats*

Nuvens de splats podem ser geradas de diversas maneiras, porém a mais comum é subamostrando uma nuvem de pontos densa. Independentemente do método de subamostragem, cada splat S_i pode ser associado com um subconjunto de pontos Π_i cobertos por ele, ou seja, pontos que estão dentro do cilindro elíptico gerado pela translação do splat plano na direção de sua normal a uma distância ε. A maioria desses métodos leva em consideração alguma métrica

Figura 31 – Métrica de erro em splats planos e em quadric splats. (a) Métricas de erro para splats planos normalmente computam a distância entre os pontos vizinhos e o plano que os aproxima. A distância máxima permitida é usada normalmente para limitar o tamanho de um splat gerado dessa forma. (b) O erro de aproximação de um quadric splat centrado em um ponto $\mathbf{P_i}$ é a distância média entre os pontos presentes nessa vizinhança e a superfície quadrática local obtida na direção de $\mathbf{w_i}$.



(a) (b)

Fonte: o autor.

de erro para controlar o tamanho de cada splat, permitindo, assim, amostragens adaptativas à curvatura local da superfície. Por exemplo, em Wu e Kobbelt (2004), a manipulação de ε é usada para essa finalidade. Inicialmente, um conjunto de splats circulares é gerado, um para cada ponto da nuvem. Cada splat inicial cresce ao adicionar pontos vizinhos na ordem das distâncias de suas projeções ao centro do splat sendo gerado. Ao limitar uma distância máxima entre um ponto vizinho e sua projeção, evita-se que splats grandes sejam gerados em regiões de alta curvatura (Figura 31a).

Seja Π uma vizinhança local de um ponto $\mathbf{P}$ definida como o conjunto de pontos localizados dentro do cilindro elíptico formado por dois eixos ortogonais $\mathbf{u_s}$ e $\mathbf{v_s}$ de tamanhos fixos r e uma altura ε. Uma estimativa empírica simples do erro médio de aproximação do splat S, gerado a partir de $\mathbf{P}$ e dos eixos $\mathbf{u_s}$ e $\mathbf{v_s}$, com relação à superfície representada pela nuvem de pontos pode ser dada pela distância média entre os pontos de Π e suas projeções sobre o plano de S:

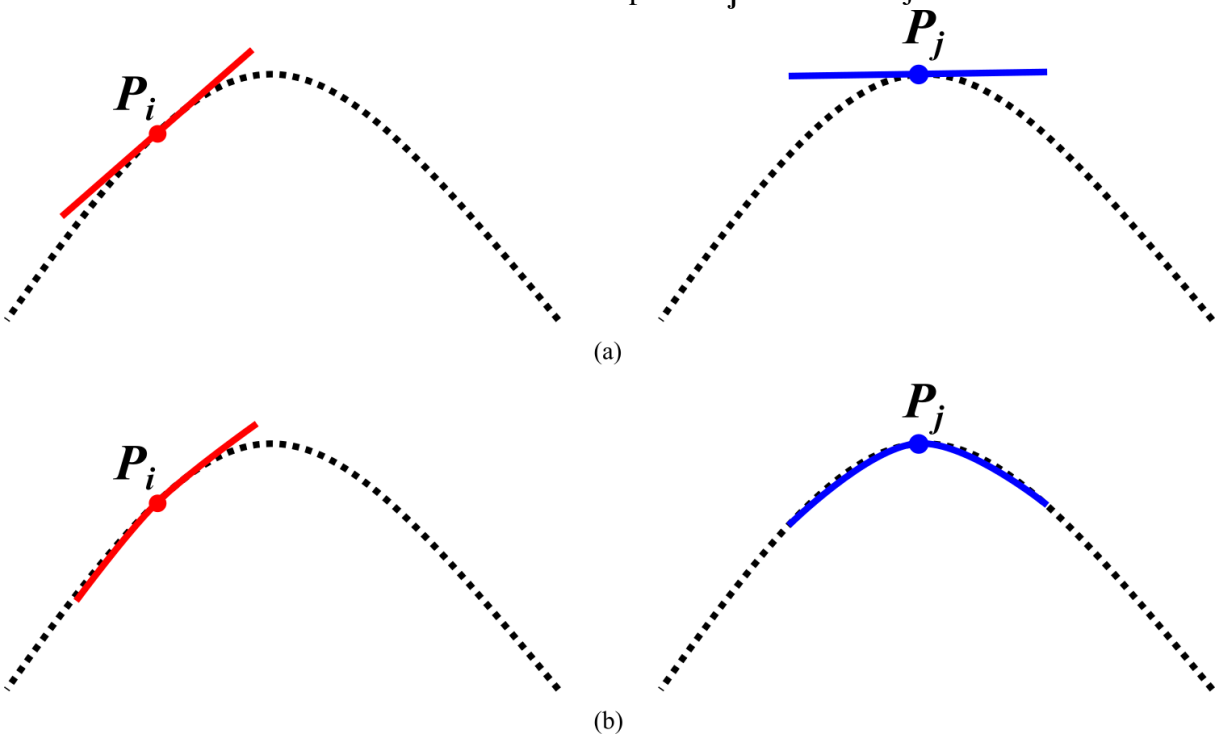
$$E_p(S) = \frac{1}{\|\Pi\|} \sum_{\mathbf{P_i} \in \Pi} \|w_i\|, \tag{4.7}$$

onde

$$w_i = (\mathbf{P_i} - \mathbf{P}) \cdot \mathbf{k_s} \tag{4.8}$$

e $\mathbf{k_s}$ é definido como na Equação 4.1.

Figura 32 – Escolha do tipo de splat em regiões de baixa (ponto $\mathbf{P_i}$) e alta (ponto $\mathbf{P_j}$) curvaturas. (a) Ao utilizar splats planos, o erro é menor em um splat centrado no ponto $\mathbf{P_i}$ do que no ponto $\mathbf{P_j}$. (b) Utilizar quadric splats reduz o erro em ambos os pontos, mas nesse caso uma amostra centrada no ponto $\mathbf{P_j}$ é mais vantajosa.



Fonte: o autor.

Apesar dessa abordagem ser simples, ela apresenta boas qualidades de amostragem adaptativas. Entretanto, ela não explora a natureza curva dos quadric splats. Por exemplo, na Figura 32 pode-se observar que locais ruins para posicionar splats planos, podem ser bons para posicionar quadric splats e vice-versa. Dessa forma, a função de erro de aproximação de um quadric splat é a distância média entre os pontos Π cobertos pelo splat e suas respectivas projeções sobre a superfície quadrática que o aproxima (Figura 31b):

$$E_q(S) = \frac{1}{\|\Pi\|} \sum_{\mathbf{P_i} \in \Pi} \|w(u_i, v_i) - w_i\|, \tag{4.9}$$

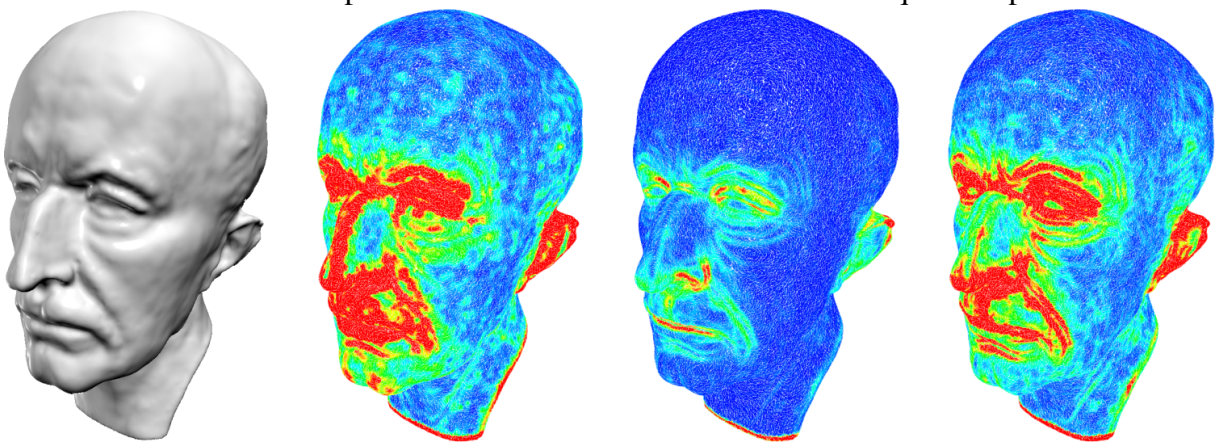
onde

$$\begin{aligned} u_i &= (\mathbf{P_i} - \mathbf{P}) \cdot \mathbf{i_s} \\ v_i &= (\mathbf{P_i} - \mathbf{P}) \cdot \mathbf{j_s} \end{aligned} \tag{4.10}$$

e $\mathbf{i_s}$ e $\mathbf{j_s}$ são definidos como na Equação 4.1.

A Figura 33 justifica a utilização dessa métrica. Na Figura 33b, cada ponto da nuvem recebeu uma cor de acordo com seu erro utilizando um splat plano centrado nele. Na Figura 33c, o mesmo foi realizado utilizando a métrica proposta. O erro médio foi reduzido

Figura 33 – Comparação dos erros computados caso um splat fosse localizado em uma determinada amostra de uma nuvem de pontos. Os erros maiores estão coloridos com cores mais quentes e os erros menores com cores mais frias. (a) Modelo Max Plank. (b) Erro computado ao simular a utilização de splats planos. (c) Erro computado ao simular a utilização de quadric splats. (d) As cores da figura anterior foram reescalonadas para ilustrar as áreas de maior erro ao usar quadric splats.



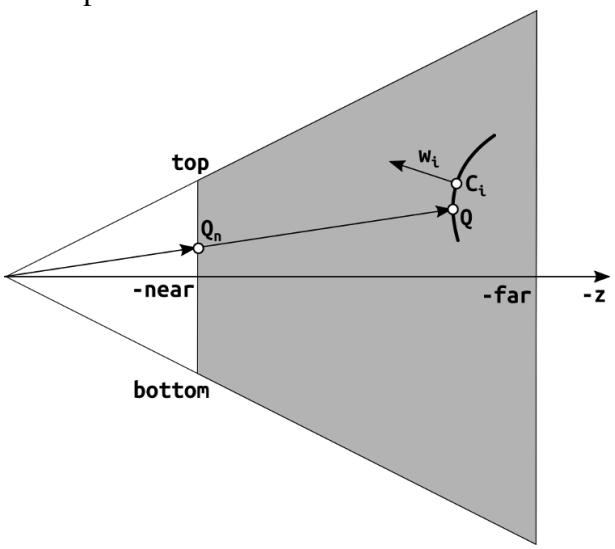
(a) (b) (c) (d)

Fonte: o autor.

em aproximadamente 72%. Ao reescalonar as cores do erro obtido com a utilização de quadric splats (Figura 33d), observa-se que as áreas com maior erro encontram-se em regiões levemente diferentes. Por exemplo, regiões como a parte frontal do nariz e a área acima da boca precisariam de uma maior quantidade de splats planos porque o erro de aproximação nessas regiões por esse tipo de primitiva é alto, entretanto essas regiões não são as mais problemáticas ao utilizar quadric splats. Por outro lado, a região abaixo dos olhos possui uma das regiões com maior erro considerando a métrica proposta, logo ela seria uma das áreas que receberia os menores quadric splats, entretanto, essas regiões não são as áreas com maior erro utilizando a métrica para splats planos.

Após essa computação de erro para cada ponto da nuvem, define-se um raio máximo para cada splat gerado a partir de um ponto – amostragem adaptativa. O processo de amostragem pode ser: por clusterização, por hierarquia ou por simulação de partículas. Entretanto, os coeficientes da superfície quadrática para cada splat devem ser recalculados, porque, como os raios das amostras sendo geradas provavelmente serão diferentes do raio fixo r escolhido para ponderar a nuvem de pontos, o conjunto de pontos cobertos por um determinado splat será diferente.

Figura 34 – Computando o ponto **Q** ao lançar um raio através do respectivo ponto $\mathbf{Q_n}$ presente no plano *near* do frustum e calculando sua interseção com a superfície quádrica definida pelo campo de normais.



Fonte: o autor.

4.3 Renderização de Quadric Splats

Esta seção descreve como renderizar *quadric splats* utilizando as funcionalidades das GPUs modernas. A abordagem de renderização proposta segue as abordagens descritas em Botsch *et al.* (2004) e Botsch *et al.* (2005). O resumo da técnica pode ser visto no Capítulo 1 e na Figura 1.

As contribuições propostas nesta tese estão presentes nas etapas de visibilidade e de atributos na pipeline da geração da imagem. No *vertex shader* de ambas as etapas, dois procedimentos são realizados por cada splat: o *back-face culling*; e a projeção de um quadrado através da computação do tamanho do *OpenGL point*. Em splats planos, o teste de *back-face* pode ser feito facilmente, assim como é feito em polígonos. Se o centro do splat é visível, ele é renderizado. Entretanto, em *quadric splats*, partes do splat podem ser visíveis enquanto outras partes podem não ser. Por esse motivo, se o centro do quadric splat não é visível, isso não significa que o quadric splat não deve ser rasterizado. Utilizar apenas o centro para tomar essa decisão pode tornar perceptível o aparecimento de buracos sobre a silhueta do modelo. Por esse motivo, além de outros, como suavização de silhueta, otimização de processamento etc, um teste mais cuidadoso é realizado para determinar quando splats devem ou não ser descartados (Seção 4.4).

No *fragment shader*, o processo de rasterização é aplicado sobre cada pixel dentro do sprite quadrado gerado pelo *OpenGL point*. A tarefa do fragment shader é descartar fragmentos

moldando o quadrado na forma do quadric splat e computar a profundidade da superfície com relação ao observador. Essas atividades são realizadas ao computar um *ray casting* da superfície.

Seja $\mathbf{Q_n}$ o ponto sobre o plano *near* do frustum que corresponde ao pixel (x,y) da janela. Lançar um raio a partir da origem (observador) através deste ponto e intersectá-lo com a superfície quadrática do splat leva ao ponto correspondente $\mathbf{Q}$ sobre o splat curvo (Figura 34). Se todos os atributos do splat são descritos no sistema de coordenadas de câmera, a equação do raio pode ser descrita no sistema de coordenadas do splat

$$\mathbf{P} = \mathbf{O} + t\ \mathbf{r} \tag{4.11}$$

onde

$$\begin{aligned} \mathbf{O} &= \begin{pmatrix} O_u \\ O_v \\ O_w \end{pmatrix} = \begin{pmatrix} -\mathbf{C_s}\cdot\mathbf{i_s} \\ -\mathbf{C_s}\cdot\mathbf{j_s} \\ -\mathbf{C_s}\cdot\mathbf{k_s} \end{pmatrix}, \text{ e} \\ \mathbf{r} &= \begin{pmatrix} r_u \\ r_v \\ r_w \end{pmatrix} = \begin{pmatrix} \mathbf{Q_n}\cdot\mathbf{i_s} \\ \mathbf{Q_n}\cdot\mathbf{j_s} \\ \mathbf{Q_n}\cdot\mathbf{k_s} \end{pmatrix}. \end{aligned} \tag{4.12}$$

Inserir a equação do raio na definição da quadrática (Equação 4.3) revela a fórmula para determinação dos pontos de interseção

$$At^2 + Bt + C = 0, \tag{4.13}$$

onde

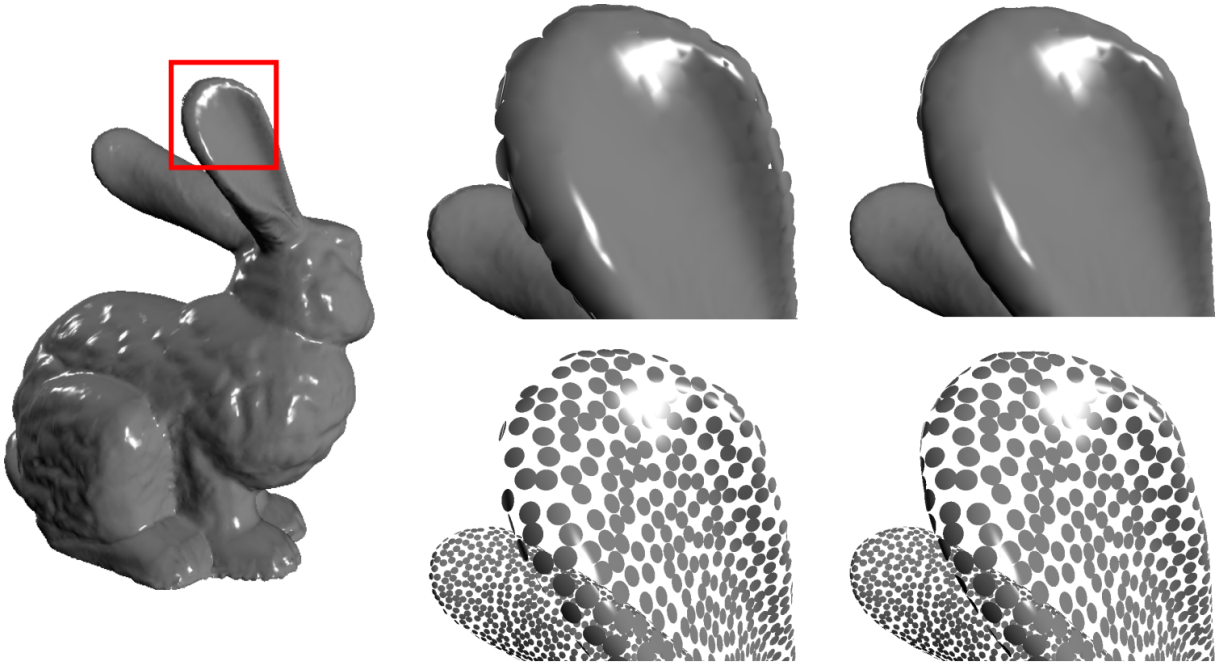
$$\begin{aligned} A &= ar_u^2 + br_v^2, \\ B &= 2ar_uO_u + 2br_vO_v + cr_u + dr_v - r_w, \\ C &= aO_u^2 + bO_v^2 + cO_u + dO_v - O_w. \end{aligned} \tag{4.14}$$

Se a Equação 4.13 não possui raízes reais, então o fragmento correspondente é descartado. Caso contrário, ainda é necessário saber se o ponto de interseção é interno ao quadric splat. O splat curvo não é uma elipse, mas é uma projeção de uma elipse sobre uma superfície quadrática na direção do vetor $\mathbf{k_s}$. Dessa forma, o teste é realizado da mesma forma como é feito para splats planos. Os parâmetros locais (u,v) do ponto $\mathbf{Q}$ podem ser definidos por

$$\begin{aligned} u &= (\mathbf{Q} - \mathbf{C_s})\cdot\mathbf{i_s}, \\ v &= (\mathbf{Q} - \mathbf{C_s})\cdot\mathbf{j_s}, \end{aligned} \tag{4.15}$$

e o teste definido na Equação 4.6 pode ser realizado. Esses parâmetros locais também são reutilizados para a função de recorte como descrito no Capítulo 3, caso seja necessário.

Figura 35 – Quadric splats podem ser parcialmente visíveis. (a) Em regiões de alta curvatura e baixas amostragens, buracos podem ser perceptíveis. (b) Renderização com splats planos. (c) Renderização com quadric splats. Note a presença de alguns splats que não foram rasterizados na imagem anterior quando apenas os seus centros foram considerados para remoção no processo de rasterização.



(a) (b) (c)

Fonte: o autor.

4.4 Detecção de *Quadric Splats* na Silhueta

Diferentemente de splats planos, quadric splats podem ser parcialmente visíveis. Em regiões de alta curvatura e baixa amostragem ou em visualizações muito próximas, não é incomum observar quadric splats cujos centros estão ocultos do observador, mas parte de sua superfície encurvada está de frente para a câmera, ou seja, seus centros estão além da linha da silhueta do objeto a partir da perspectiva do observador. Em tais situações, quando métodos usuais de remoção de superfícies ocultas são aplicados, estes splats são removidos e buracos aparecerão no modelo renderizado (Figura 35). Em regiões afastadas da silhueta, o Phong splatting de amostras planas provê um melhor custo-benefício na renderização porque o ray casting de superfícies planas é mais eficiente computacionalmente do que o ray casting de superfícies quadráticas. Dessa forma, uma computação mais acurada de splats próximos da silhueta melhora a eficiência da renderização do modelo, sem comprometer a qualidade da imagem gerada. Para usar os quadric splats somente onde eles são necessários, esta seção descreve um método para detectá-los na silhueta do modelo ainda no vertex shader, antes da

rasterização no fragment shader.

O método de detecção de silhueta considera que um splat esteja na silhueta se sua extensão ou uma escala de sua extensão cruza a linha da silhueta da superfície quadrática associada ao splat. Seja $\mathbf{E} = (e_u, e_v, e_w)^T$ o ponto onde localiza-se a câmera no sistema de coordenadas local do splat. Um ponto $\mathbf{P} = (u, v, w(u,v))^T$ sobre a superfície quadrática está na curva da silhueta se

$$(\mathbf{E} - \mathbf{P}) \perp \mathbf{n_p} \tag{4.16}$$

onde $\mathbf{n_p}$ é o vetor unitário normal a $w(u,v)$ em $\mathbf{P}$. Essa propriedade pode ser reescrita como

$$(\mathbf{E} - \mathbf{P}) \cdot \mathbf{n_p} = 0. \tag{4.17}$$

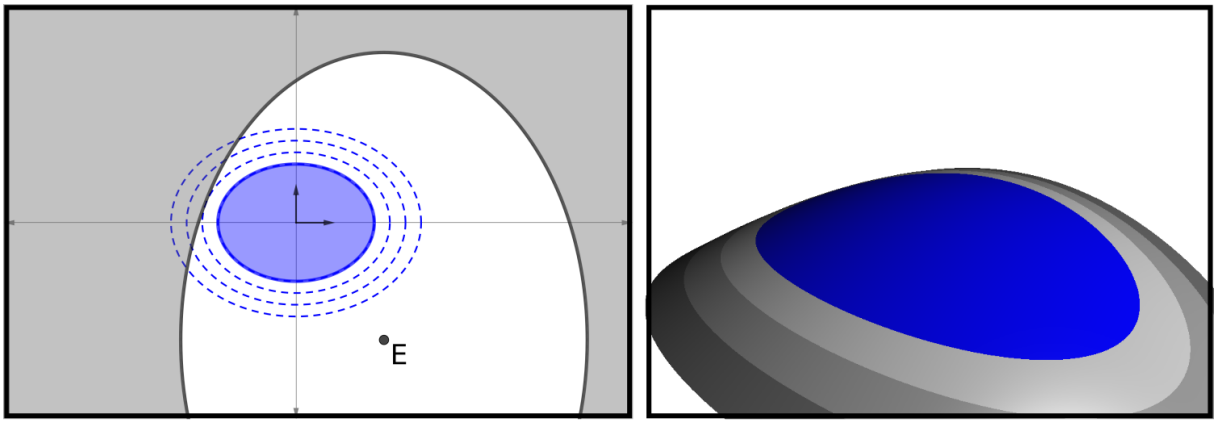
Substituindo $\mathbf{E}$, $\mathbf{P}$ e $\mathbf{n_p}$ por suas definições na Equação 4.17, o resultado, em termos de u e v, é

$$au^2 + bv^2 - 2ae_u u - 2be_v v - ce_u - de_v + e_w = 0. \tag{4.18}$$

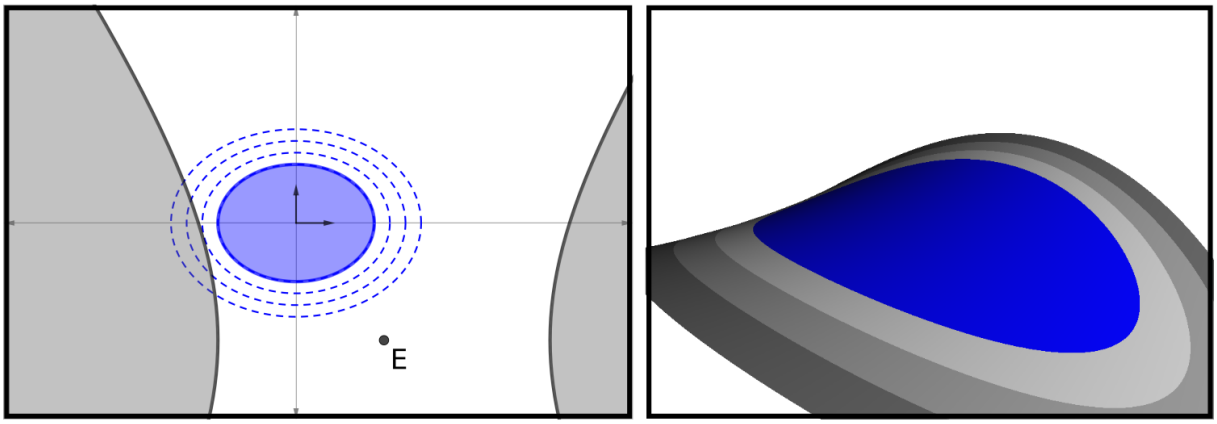
A Equação 4.18 define uma superfície quádrica sobre o eixo de coordenadas local do splat, entretanto, ela não possui nenhum coeficiente diferente de zero de algum termo do eixo w, ou seja, se trata de uma superfície cilíndrica paralela ao vetor $\mathbf{k_s}$. Se parte da extensão do quadric splat é interceptada por essa superfície, a primitiva é determinada como pertencente à silhueta. O quadric splat é uma projeção da elipse definida na Equação 4.6 sobre a superfície quadrática definida na Equação 4.3 na direção do vetor $\mathbf{k_s}$. Dessa forma, o problema pode ser visualizado de forma simplificada removendo o eixo w, ou seja, como um problema de interseção entre a curva que a Equação 4.18 define sobre o plano $\mathbf{i_s} \times \mathbf{j_s}$ e a borda do splat (Equação 4.6). A Figura 36 ilustra à esquerda visualizações destas curvas vistas em uma projeção ortográfica a partir do eixo $\mathbf{k_s}$ e à direita visualizações em perspectiva do quadric splat utilizando o ponto $\mathbf{E}$ como centro de projeção. Nota-se que quando os valores α_s e β_s do mapa de normais possuem sinais iguais, a curva definida pela Equação 4.18 é uma elipse (Figura 36a), mas quando possuem sinais diferentes, a curva é uma hipérbole (Figura 36b).

Para resolver o sistema com as Equações 4.18 e 4.6, pode-se eliminar a variável v para obter uma equação polinomial de quarto grau $H(u) = 0$. Não é necessário determinar a solução exata do sistema para saber onde exatamente ocorre a interseção, apenas se ela existe. Então, o problema pode ser simplificado para computar as raízes de $\partial H(u)/\partial u$ e então determinar se $H(u)$ possui raízes reais: se o valor mínimo de $H(u)$ é negativo ou o valor máximo de $H(u)$ é

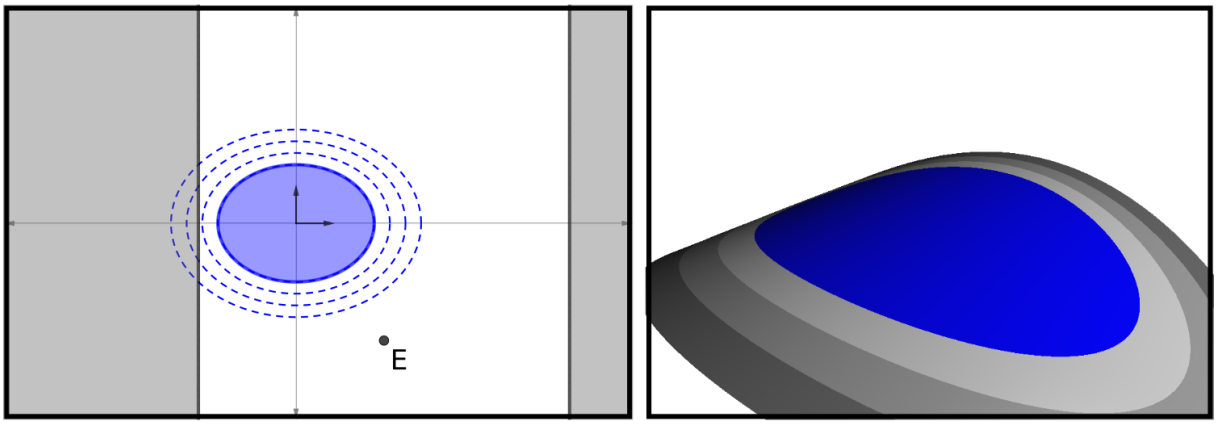
Figura 36 – O método de detecção de silhueta computa se há uma interseção entre a curva da silhueta da superfície quádrica e alguma escala da borda do splat. (a) A curva da silhueta é uma elipse quando as curvaturas nas direções dos eixos do splat possuem o mesmo sinal. (b) A curva da silhueta é uma hipérbole quando essas curvaturas possuem sinais diferentes. (c) A curva da silhueta é um segmento de reta quando uma das curvaturas for zero.



(a)



(b)



(c)

Fonte: o autor.

positivo, então o sistema de equações possui solução e, logo, há interseção entre as Equações 4.18 e 4.6 acarretando na detecção de um quadric splat próximo da silhueta.

Quando o splat não possui curvatura na direção de um dos eixos do splat, ou seja, quando α_s ou β_s são iguais a zero, trata-se de um caso degenerado que pode ser resolvido de forma mais simples. Sem perda de generalidade, seja α_s igual a zero. A Equação 4.18 é simplificada para

$$bv^2 - 2be_v v - ce_u - de_v + e_w = 0. \tag{4.19}$$

A curva da silhueta são retas paralelas ao eixo u que cruzam o eixo v nas raízes da Equação 4.19 (Figura 36c). Dessa forma, o quadric splat tem parte de sua extensão em uma região oculta se

$$\min\{|v_1|, |v_2|\} < |\mathbf{v_s}| \tag{4.20}$$

onde v_1 e v_2 são as raízes da Equação 4.19.

Essa abordagem é suficiente se for usada somente para remover os splats voltados para trás. Entretanto, se todos os splats não detectados pela silhueta forem renderizados como splats planos, alguns artefatos persistirão. A Figura 36 mostra três casos onde o quadric splat azul não é detectado como pertencendo a uma silhueta se apenas sua extensão for usada como parâmetro. Mas renderizá-lo como splat plano acarretaria em artefatos visíveis. Para evitar esse tipo de problema, o usuário pode manipular um parâmetro global σ que aplica uma escala uniforme sobre a elipse do splat descrita na Equação 4.6 antes da execução do algoritmo de detecção. Apesar de ser uma abordagem empírica, a manipulação do parâmetro σ apresenta uma menor quantidade de falsos positivos do que a abordagem utilizada em Zakaria e Seidel (2004).

Este método requer mais operações computacionalmente complexas, mas elas são realizadas em vertex shader, ou seja, o número de operações é significantemente menor do que o número de operações realizadas no fragment shader. Isto é especialmente verdade quando o tamanho do splat é grande em espaço de imagem, uma situação comum em modelos com baixa amostragem ou em exibições muito próximas.

4.5 Considerações Finais

Este capítulo apresentou uma maneira alternativa de renderizar splats, os *quadric splats*, para evitar a presença de artefatos na silhueta do objeto. Inicialmente, a forma dos splats curvos propostos foi definida como a função quadrática descrita a partir da função vetorial

bilinear de vetores normais computada para cada splat na técnica de Phong Splatting. Em seguida, discutiu-se uma métrica de erro aplicada à geração de splats curvos a partir de uma nuvem de pontos que melhor exploram a característica das novas primitivas. A rasterização de quadric splats no fragment shader é apenas a realização de um ray casting de uma superfície quadrática. Como essa rasterização é mais custosa, é interessante que ela seja utilizada apenas onde a sua ausência seja sentida, como em regiões próximas de silhuetas. Logo, a detecção de quadric splats próximos da silhueta se fez necessária. Esses splats são aqueles cuja extensão ou uma escala de sua extensão é interceptada pela curva da silhueta da superfície quadrática associada ao splat com relação à posição da câmera. Essa detecção de splats presentes na silhueta é vantajosa, pois é realizada por primitiva e não por fragmento, além de poder ser utilizada para outras finalidades, como renderizações não realistas de nuvens de splats.

5 RESULTADOS E ANÁLISE

Nesse capítulo cada uma das contribuições propostas nesta tese são analisadas e comparadas. Na Seção 5.1 verifica-se a correlação entre a amostragem de modelos baseados em splats e a eficiência na geração e renderização desse modelo. Na Seção 5.2, são apresentados os resultados obtidos pela adaptação da amostragem próxima de arestas, proposta na Seção 3.1. Na Seção 5.3, as melhorias obtidas com recorte curvo de splats são apresentadas e discutidas. A Seção 5.4 apresenta a melhoria obtida na renderização de silhuetas e arestas com a utilização de splats curvos. A Seção 5.5 compara e analisa a detecção de silhueta através do uso de quadric splats. E por fim, a Seção 5.6 mostra os tempos de renderização obtidos com o uso de quadric splats.

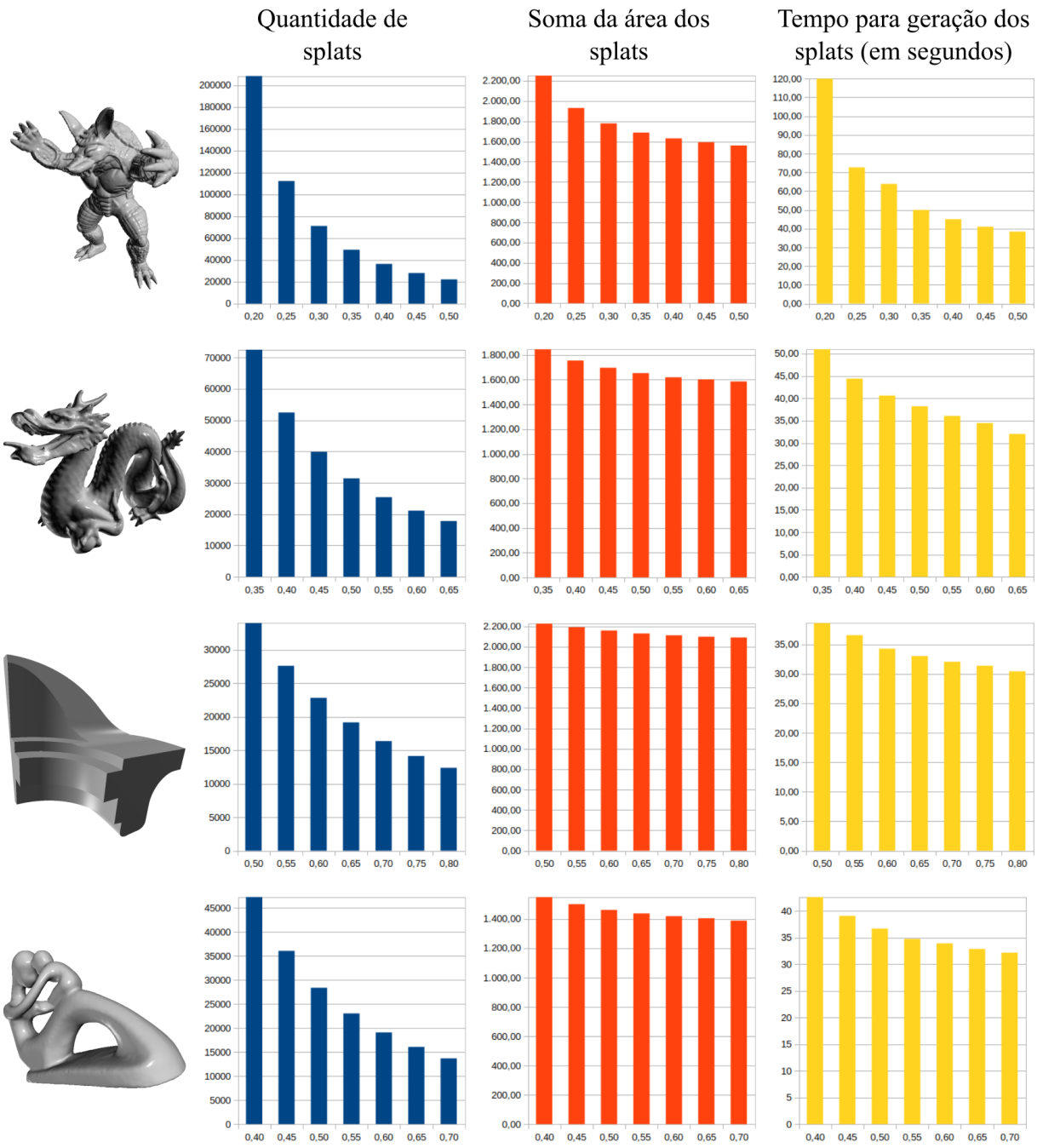
5.1 Amostragem

Uma das principais motivações da utilização de nuvens de splats é a possibilidade de reconstruir a superfície representada por uma nuvem de pontos sem a necessidade de manter uma topologia entre os elementos e obter, como resultado, uma renderização comparável a malhas poligonais. Nesse contexto, busca-se determinar qual é a quantidade de primitivas mais adequada tentando equilibrar o tempo de geração das amostras, o tempo de renderização da nuvem de splats e a qualidade das imagens geradas.

Neste trabalho, sugere-se a utilização de amostras maiores e em menor quantidade, mantendo a qualidade da renderização com a utilização de primitivas não lineares. Por esse motivo, investiga-se a vantagem dessa situação. O número de chamadas ao *vertex shader* e ao *fragment shader* são bons indicadores da eficiência da renderização de qualquer modelo. O número de chamadas ao vertex shader é igual ao número de primitivas e o número de chamadas ao fragment shader depende de diversos outros fatores como a distância dos splats à câmera, a resolução da imagem, a sobreposição das amostras etc.

Na Figura 37, compara-se a quantidade de amostras de uma nuvem de splats (ilustrados nos eixos verticais da esquerda) com a área total das amostras e o tempo para gerar esse modelo (ilustrados nos eixos verticais da direita). Para tornar mais fácil a correlação entre essas grandezas, as amostragens dos modelos é uniforme, ou seja, com raios dos splats aproximadamente constantes (ilustrados nos eixos horizontais). Nas diferentes amostragens, os raios são escolhidos proporcionalmente à diagonal principal da *bounding box* dos modelos. Em todas

Figura 37 – Analise de diferentes parâmetros em amostragens uniformes de modelos por splats. Cada gráfico é em função do raio escolhido para a amostragem uniforme. Os eixos horizontais ilustram as medidas do raios como uma porcentagem da diagonal principal de cada modelo.



Fonte: o autor.

essas amostragens, a nuvem de pontos inicial consistia em uma nuvem de pontos com exatamente um milhão de amostras dispostas de forma pseudo-aleatória sobre as superfícies dos modelos. O processo de subamostragem escolhido foi o método de clusterização por crescimento de região (WU; KOBBELT, 2004) com o auxílio de uma octree. Nos gráficos, a soma das áreas dos splats em espaço de objeto pode ser utilizada como uma grandeza correlata ao número de chamadas ao fragment shader. Apesar de não haver uma correlação exata, ela é independente da distância do modelo com relação a câmera e da resolução da imagem. Mesmo que, em uma amostragem uniforme, metade das amostras seja descartada durante a etapa de remoção de superfícies ocultas, essa estimativa é plausível, uma vez que o raio do splat determina a quantidade de fragmentos a serem rasterizados.

A partir da Figura 37, podem-se tirar algumas conclusões. A primeira e mais óbvia é a de que, quanto maior for o raio dos splats, menor é a quantidade de splats na nuvem resultante. Essa redução não é linear, mostrando que há um limiar de melhor custo-benefício em cada modelo. Sabe-se que a menor quantidade de amostras reduz o número de chamadas ao vertex shader, mas, tão importante quanto essa redução, é o menor uso da memória gráfica. As placas de vídeo atuais possuem memórias internas consideravelmente grandes, entretanto, em caso de modelos não estáticos, os dados das amostras enviados da CPU para a GPU podem tornar-se facilmente o gargalo da renderização.

O segundo parâmetro analisado na Figura 37 é a soma das áreas dos splats. Essa soma decai à medida que a quantidade de splats diminui. A redução da área total dos splats sugere uma redução do número de chamadas ao fragment shader. Observa-se que essa redução não é na mesma proporção que a redução na quantidade de primitivas, mas deve-se ter em mente que essa redução torna-se cada vez mais relevante quanto maior for a resolução da imagem, pois o tempo de rasterização passa a dominar o tempo de renderização do modelo. Observa-se também que essa redução apresenta um limiar, tornando-se quase irrelevante em alguns modelos com amostragem já bastante baixas. Essa análise leva em consideração apenas a quantidade de splats e a soma das áreas das amostras para estimar a eficiência da renderização do modelo, mas deve-se considerar que em um modelo com amostras maiores, artefatos tornam-se mais perceptíveis, exigindo uma renderização mais cuidadosa e, consequentemente, mais complexa.

O último parâmetro presente na Figura 37 a ser analisado é o tempo de geração das amostras. Percebe-se que, quanto maior for o raio dos splats, menor é esse tempo. Splats são gerados a partir da nuvem de pontos. Com splats maiores, cada amostra gerada cobre

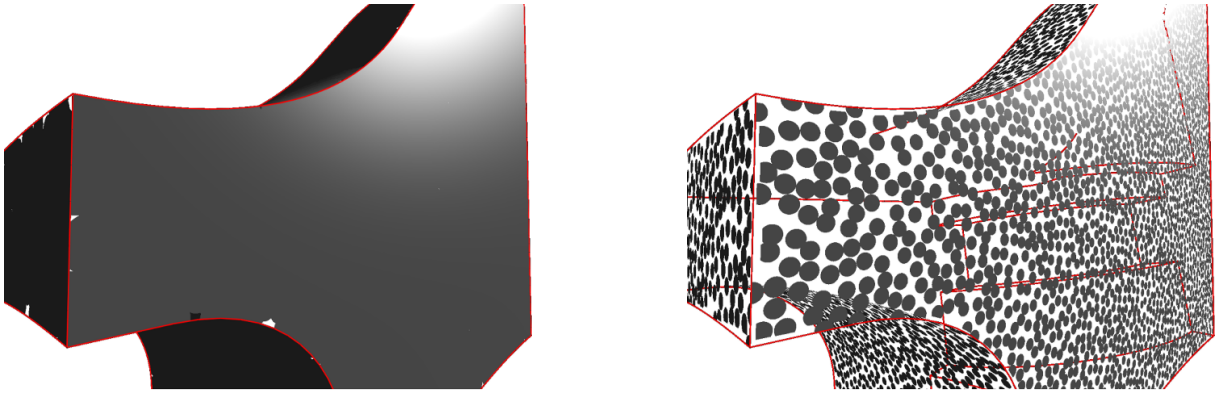
uma maior quantidade de pontos, acarretando uma aceleração do processo de amostragem, independentemente de qual método seja utilizado. Com as devidas otimizações, processos de subamostragem eficientes podem permitir renderizações de nuvens de pontos não estáticas bem mais eficientes.

Um fator muito importante que não pode ser visualizado nessa análise é a qualidade da renderização. Sabe-se que, em qualquer modelo representado de forma discreta, uma quantidade maior de amostras permite renderização com qualidade superior. Porém, sabe-se também que há uma amostragem que apresenta o melhor custo-benefício com a qualidade da renderização. Métodos adaptativos de amostragens normalmente reduzem a quantidade de amostras necessárias para atingir esse objetivo posicionando maiores quantidades de primitivas nas regiões do modelo onde elas são mais necessárias. Nesta tese, propõe-se métodos não lineares para recortar splats e para rasterizá-los nas silhuetas do modelo, permitindo assim alcançar objetivo similar. Para comprovar a qualidade dos métodos propostos, as análises a seguir normalmente irão mostrar a melhora na qualidade de renderização para modelos com mesma amostragem e a melhora na eficiência que se pode alcançar ao obter renderizações de qualidade similar mas com nuvens de splats com quantidades inferiores de amostras.

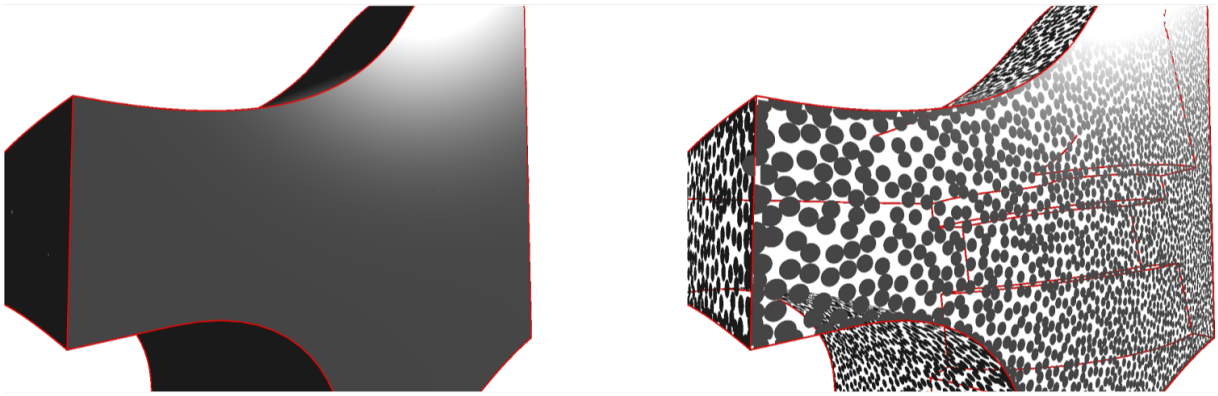
5.2 Amostragem na proximidade de arestas

A Seção 3.1 analisa o processo de amostragem de splats em regiões próximas de arestas e cantos e propõe uma adaptação do processo de amostragem proposto em (WU; KOBBELT, 2004). Apesar desse trabalho tratar-se de um processo de amostragem por clusterização, a ideia proposta pode ser aplicável a qualquer processo de subamostragem. Em Wu e Kobbelt (2004), pontos presentes no fecho convexo do conjunto de pontos cobertos por um splat são considerados não cobertos para que sejam cobertos por outros splats próximos garantindo a ausência de buracos. Entretanto, esse trabalho e a grande maioria dos métodos de subamostragem pressupõem que a superfície de um modelo é suave, ou seja, a vizinhança de um determinado ponto representa uma mesma superfície suave. Essa observação não é verdade nos entornos de uma aresta de um modelo. Se os cálculos de vizinhança utilizados nesses métodos não levarem em consideração a orientação das superfícies às quais os pontos pertencem, alguns pontos podem ser considerados cobertos erroneamente, acarretando buracos nos entornos das arestas (Figura 38a). Se os cálculos de vizinhança descartarem pontos presentes no lado oposto de uma aresta, os pontos mais próximos da aresta estarão nos fechos convexos de todos os

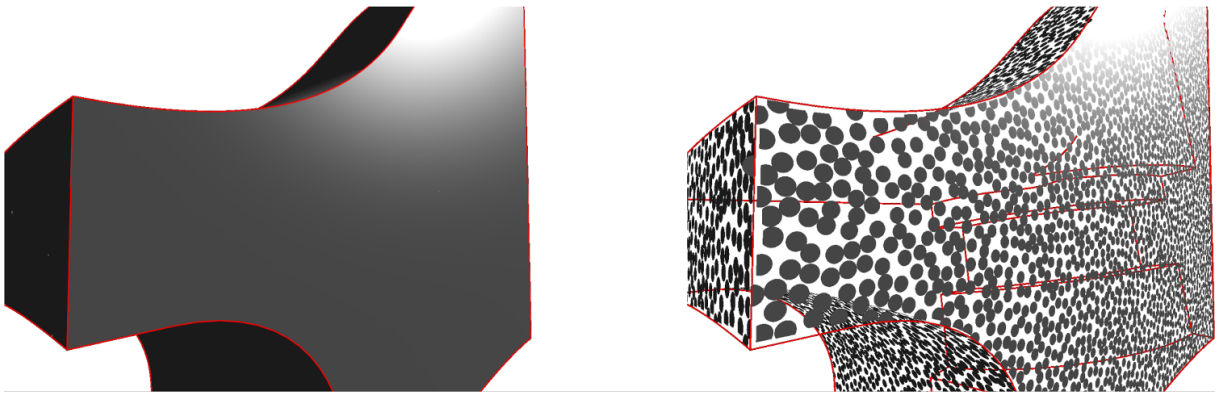
Figura 38 – Amostragem de splats na proximidade de arestas. (a) Considerando apenas a posição dos pontos para determinar se são cobertos por um splat. (b) Considerando as posições e normais dos pontos. (c) Adicionando as curvas das arestas na determinação de pontos cobertos por um splat.



(a)



(b)



(c)

Fonte: o autor.

splats próximos gerados ao seu redor, ou seja, esses pontos serão considerados como pontos não cobertos e tornar-se-ão splats no processo de subamostragem. Isso gera uma sobreamostragem desnecessária nos entornos das arestas do modelo (Figura 38b).

Na seção 3.1 propõe-se adicionar pedaços das linhas das arestas no cálculo do fecho convexo utilizado para determinar a cobertura de um splat. Apesar de não serem informações originais da nuvem de pontos a ser amostrada, as curvas das arestas são informações importantes a serem exploradas no processo de amostragem. Os pontos próximos de arestas não estarão mais em todos os fechos convexos dos splats gerados a seu redor porque os pontos mais externos cobertos por um splat nessas regiões são os pontos da linha da aresta, evitando o problema anterior da sobreamostragem nessas áreas (Figura 38c).

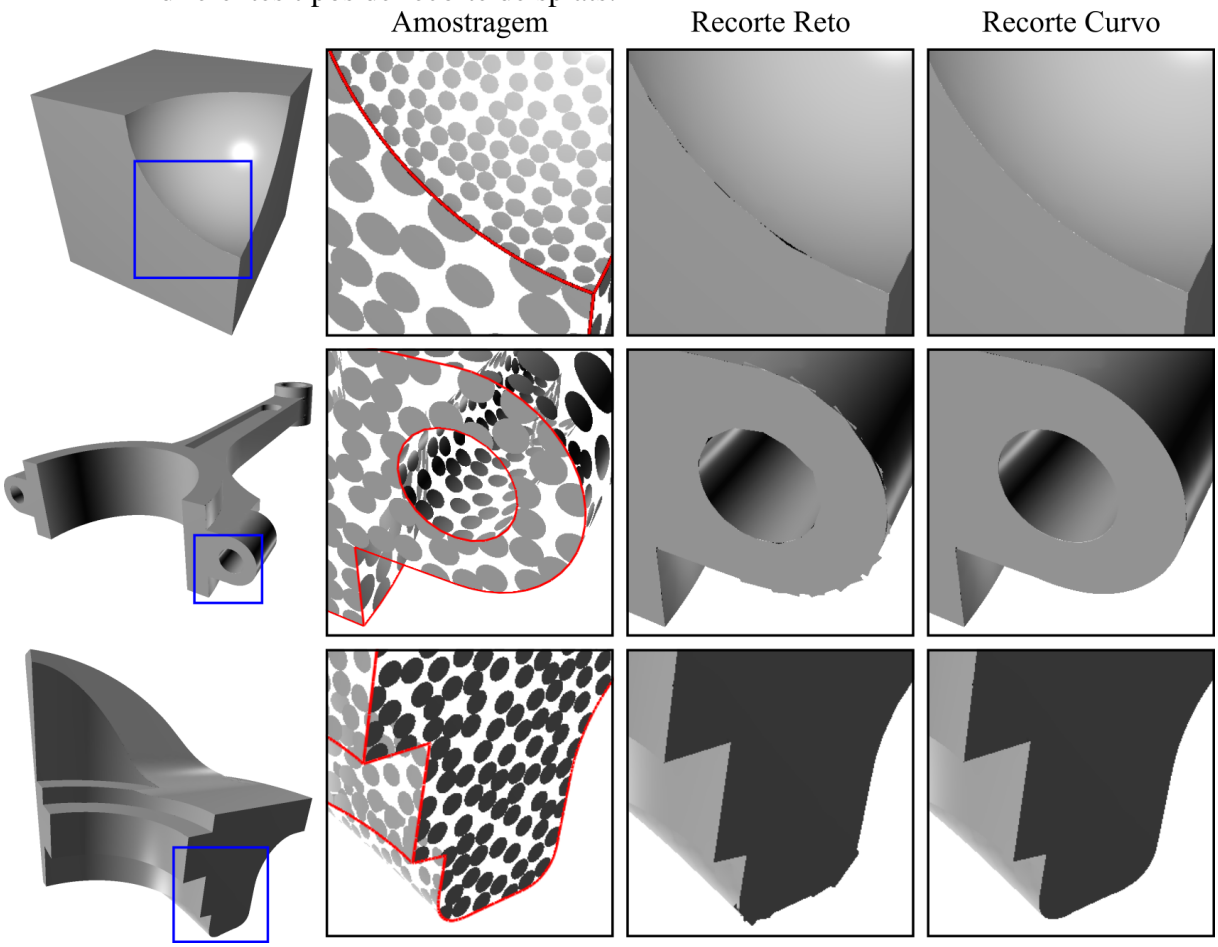
5.3 Recorte curvo de splats

Em modelos baseados em splats, os raios dos discos devem ser suficientemente grandes para evitar buracos, mas sem gerar artefatos. Esses artefatos são mais prováveis de acontecer em regiões de alta curvatura e podem ser minimizados aumentando a densidade de amostras. Para uma aresta, uma densidade muito grande seria necessária para evitar que artefatos fossem perceptíveis e essa solução ainda dependeria da distância do observador para o modelo. A Figura 2 ilustra o problema de se representar arestas e cantos em um modelo baseado em splats. Por esse motivo, splats próximos de arestas precisam ser recortados. Estes recortes podem ser interpretados como uma forma de representação discreta da aresta do modelo.

Normalmente, splats são recortados por linhas retas (PAULY *et al.*, 2003b; ZWICKER *et al.*, 2004). Isso significa que as arestas do modelo são aproximadas por segmentos de reta desconexos, uma vez que cada splat é independente. A ideia proposta nesta tese é utilizar uma classe específica de curvas racionais de Bézier para recortar splats que cruzam as arestas do modelo. A Figura 39 compara o recorte curvo proposto nesse trabalho com o recorte linear para modelos com a mesma quantidade e disposição de splats. Note que artefatos como buracos ou saliências, mais presentes no recorte por retas, são mais perceptíveis quando os splats são maiores e mais esparsos. O recorte curvo apresenta resultados superiores com arestas mais suaves.

Um outra vantagem de se obter boas renderizações de modelos com baixa amostragem é a redução no consumo de memória gráfica. Assumindo que todos os números de ponto flutuante ocupam 4 bytes de memória, os splats presentes nos modelos da Figura 39 possuem 64

Figura 39 – Renderizações dos modelos que apresentam arestas com mesma amostragem, mas diferentes tipos de recorte de splats.



Fonte: o autor.

Tabela 1 – Comparativo do uso de memória pelas técnicas de recorte reto e recorte curvo em diversos modelos. (*: N = Número de splats. NR = Número de splats recortados)

Modelos	N	NR	Memória (KB)		
			Reto	Curvo	Razão
Cube-S	619	233	40,51	43,24	6,74%
Piston Rod	4937	3237	333,85	371,79	11,36%
Fandisk	3163	1060	205,97	218,39	6,03%

Fonte: dados da pesquisa.

Bytes de informação cada (3 valores para posição do centro, 6 valores para eixos, 3 valores para cor, 4 valores para mapa de normais usado no Phong shading). Numa implementação simples, onde todos os splats possuem as mesmas informações, ou seja, os splats que possuem recorte tem o mesmo tamanho e são armazenados no mesmo buffer que os demais splats, o recorte curvo adiciona $16,67\%$ no uso da memória em comparação com o recorte reto, uma vez que o recorte reto necessita de 2 valores de ponto flutuante, enquanto o recorte curvo precisa de

5. Numa implementação mais complexa, com diferentes pares de shader e diferentes VBOs (*Vertex Buffer Object*), um para splats que possuem recorte e outro para aqueles que não possuem recorte, a Tabela 1 mostra que o acréscimo de memória necessário para armazenar modelos possuindo recorte curvo com relação ao recorte reto pode ser ainda menor. O modelo *Piston-Rod* possui uma diferença maior no uso de memória, porque possui, proporcionalmente, uma maior quantidade de splats recortados com relação a quantidade total de amostras.

Tabela 2 – Comparativo do uso de memória pelas técnicas de recorte reto e recorte curvo para uma mesma superfície. (*: N = Número de splats. NR = Número de splats recortados)

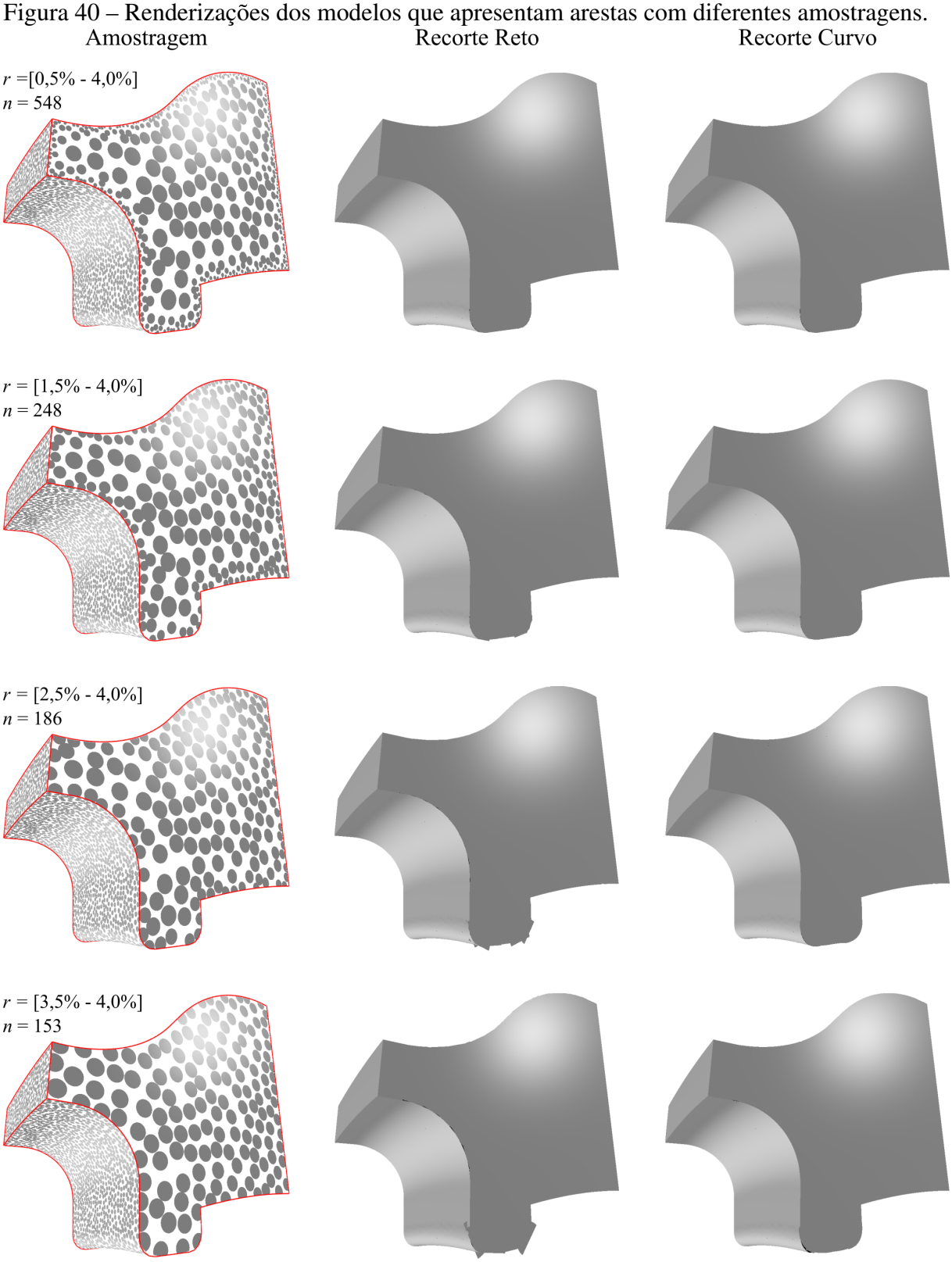
Dimensões	N	NR	Memória (KB)		
			Reto	Curvo	Razão
0,5% - 4,0%	548	336	36,88	40,81	10,68%
1,5% - 4,0%	248	120	16,44	17,84	8,56%
2,5% - 4,0%	186	81	12,26	13,21	7,74%
3,5% - 4,0%	153	60	10,03	10,73	7,01%

Fonte: dados da pesquisa.

A Figura 40 mostra renderizações utilizando recorte reto e recorte curvo de uma face do modelo Fandisk em diferentes resoluções. Os raios dos splats são descritos em comparação com a diagonal principal da bounding box do modelo e são adaptados de acordo com a proximidade com as bordas da face. A Tabela 2 resume as quantidades de memória correspondentes em uma implementação com VBOs diferentes para splats recortados e para os demais. Nota-se que, com recorte curvo e usando modelos de menor densidade de splats é possível obter renderização com qualidade comparável à renderização obtida com recorte reto com modelos muito mais densos. Por exemplo, a renderização obtida com recorte curvo de splats variando de 2,5% a 4,0% de raio é comparável em qualidade a renderização obtida com recorte reto de splats variando de 0,5% a 4,0% de raio, porém o modelo com recorte curvo utiliza 35,8% da memória utilizada pelo modelo com recorte reto.

Como cada splat possui sua própria curva de recorte individual, a aresta continua não sendo representada de forma contínua, o que pode levar a artefatos como os mencionados anteriormente. Isso pode ser observado na Figura 40 na renderização do modelo com recorte curvo de splats variando de 3,5% a 4,0% de raio. Entretanto, o método permite uma redução de amostras considerável até que os artefatos voltem a aparecer no modelo. O recorte linear realizado por uma polilinha proposto por Zhang e Kaufman (2007) pode não possuir a mesma suavidade do que o recorte curvo, mas provavelmente possui resultados sem buracos e saliências

Figura 40 – Renderizações dos modelos que apresentam arestas com diferentes amostragens.



Fonte: o autor.

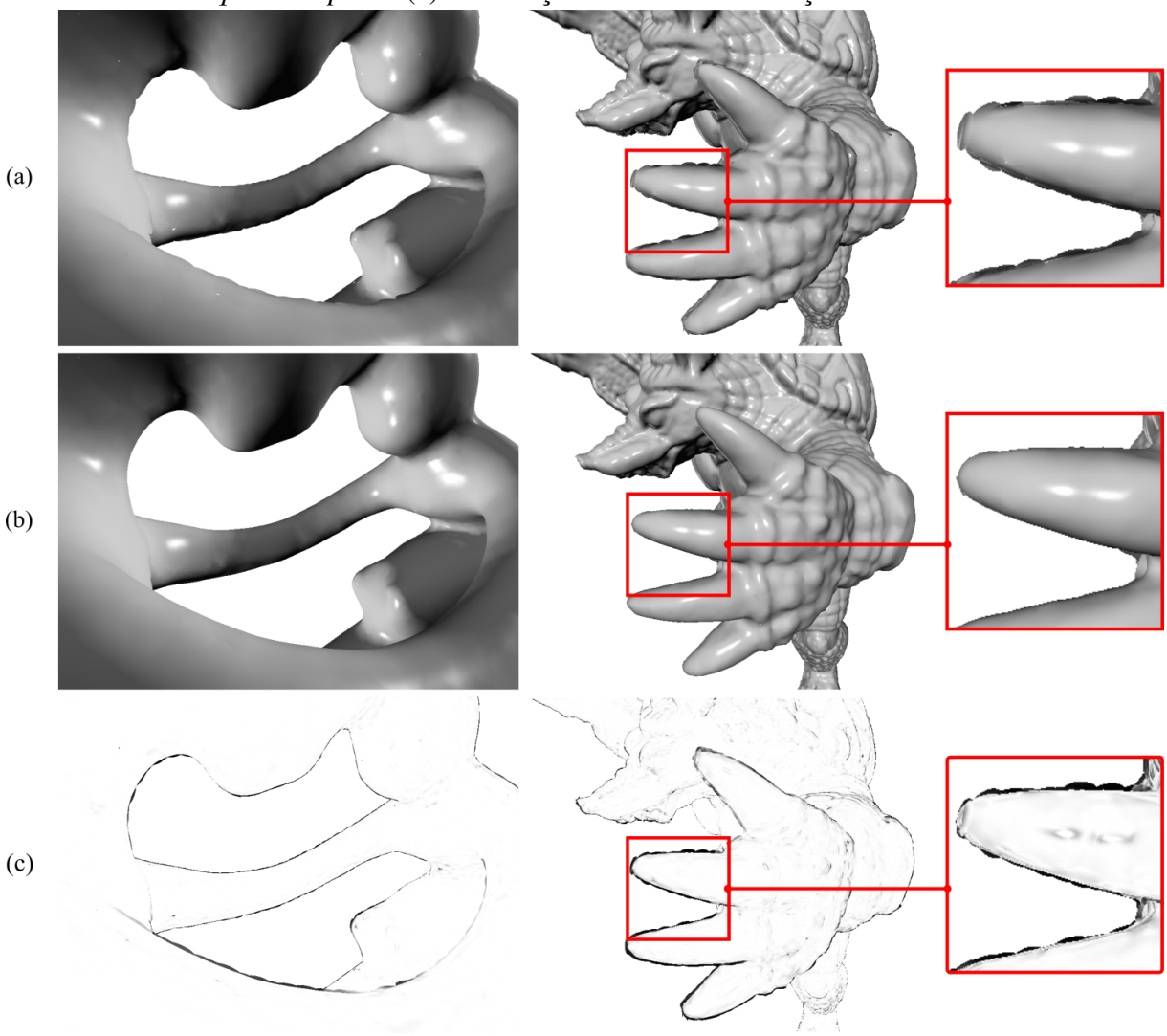
porque a polilinha que representa a aresta é uma estrutura global aplicada a todos os splats no seu entorno. No entanto, como a interseção dessa polilinha com o splat pode ter uma quantidade variável de segmentos, um par de shaders precisa ser implementado para cada caso. O recorte curvo mostra, então, um bom equilíbrio entre qualidade de rasterização, eficiência e facilidade de implementação.

5.4 Representação de silhuetas

Em termos de qualidade da renderização, os *quadric splats* propostos nesta tese são comparados com os splats planos usando sombreamento de Phong (BOTSCH *et al.*, 2004). Como todas as nuvens de splats foram geradas através de um método de subamostragem a partir de uma nuvem de pontos densa, os campos de normais foram computados por um método de mínimos quadrados usando a nuvem de pontos original.

A Figura 41 mostra uma comparação entre modelos com mesma quantidade e posicionamento das amostras usando splats planos e quadric splats, ambos circulares. As renderizações dos modelos inteiros usando ambos os tipos de primitivas são similares, mas as diferenças estão localizadas primariamente nas silhuetas dos modelos (Figura 41c). A motivação de usar Phong Splatting é obter boas renderizações de modelos com baixas densidades de amostragem a fim de acelerar o processo de renderização. Quanto menor for a densidade das amostras, maior é o erro permitido de aproximação entre os splats e a nuvem de pontos original. Como o mapa de normais permite um sombreamento muito superior aos splats, esse erro é facilmente tolerado. Entretanto, como mostrado no modelo *Fertility* (Figuras 3 e 41a), os artefatos presentes nas silhuetas dos modelos não podem ser escondidos pelo sombreamento, mesmo em regiões onde a curvatura é pequena. As silhuetas de um modelo são características dependentes da orientação da câmera, logo haveria apenas duas maneiras de lidar com esse problema usando splats planos: diminuindo o tamanho dos splats em todo o modelo, consequentemente aumentando a densidade da nuvem e indo na contramão da motivação da técnica proposta; ou utilizando o geometry shader para inserir novas amostras apenas na silhueta, diminuindo consideravelmente a eficiência da renderização (lembre que são duas etapas de rasterização para gerar uma imagem) e com uma implementação não tão simples, uma vez que informações de topologia em uma nuvem de splats não estão presentes. Por outro lado, a utilização de quadric splats, reduz consideravelmente esses artefatos na silhueta do modelo, mantendo a filosofia original do trabalho de Botsch *et al.* (2004), sem aumentar muito o tempo de renderização. Outra vantagem está na maior liberdade

Figura 41 – Renderizações dos modelos *Fertility* e *Armadillo* com mesma amostragem, mas diferentes tipos de splats. (a) Renderizações com splats planos. (b) Renderizações com *quadric splats*. (c) Diferenças entre as renderizações.

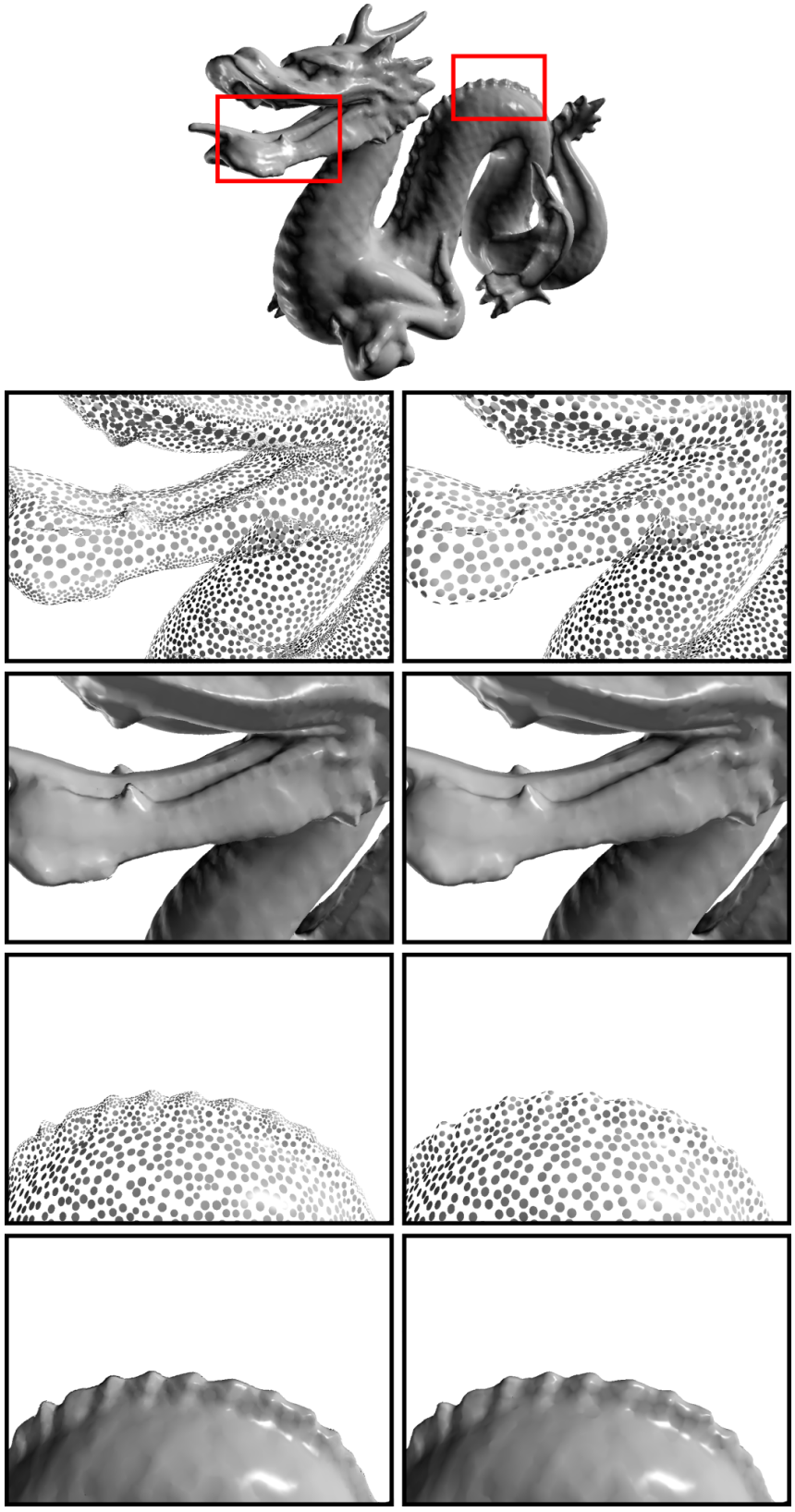


Fonte: o autor.

na geração da nuvem de splats. Note que em regiões onde a curvatura é alta, porém constante, como nos dedos do modelo *Armadillo*, a utilização de splats maiores é permitida desde que usando quadric splats para obter uma renderização com menos artefatos e melhor qualidade visual.

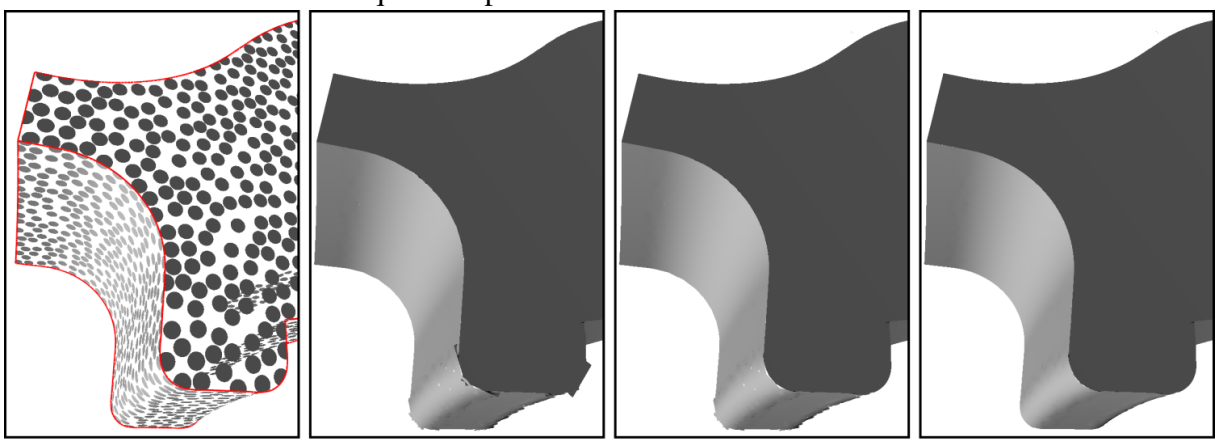
A Figura 42 compara a utilização dos dois tipos de splats em um mesmo modelo, com qualidades visuais similares, mas amostragens adaptativas diferentes. Na versão do modelo *Dragon* amostrado com splats planos, foi utilizada uma amostragem adaptativa com raios de splat variando entre $0,2$ a $0,7\%$ da diagonal principal da bounding box do modelo, resultando numa quantidade total de 57 mil splats. Na versão do modelo amostrado por quadric splats, os raios das amostras variam entre $0,5$ e $0,7\%$ da diagonal principal da bounding box do modelo,

Figura 42 – Representação de silhuetas para diferentes densidades de amostragem, mas aproximadamente a mesma qualidade visual do modelo *Dragon*. As imagens detalhadas são das regiões destacadas na imagem do topo. A coluna da esquerda ilustra as amostragens e a renderização final do modelo através de splats planos e a coluna da direita ilustra as amostragens e a renderização final através de quadric splats.



Fonte: o autor.

Figura 43 – Unindo quadric splats à rasterização de arestas curvas. (a) Amostragem e arestas do modelo. (b) Recorte linear e splats planos. (c) Recorte curvo e splats planos. (d) Recorte curvo e quadric splats.



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

Fonte: o autor.

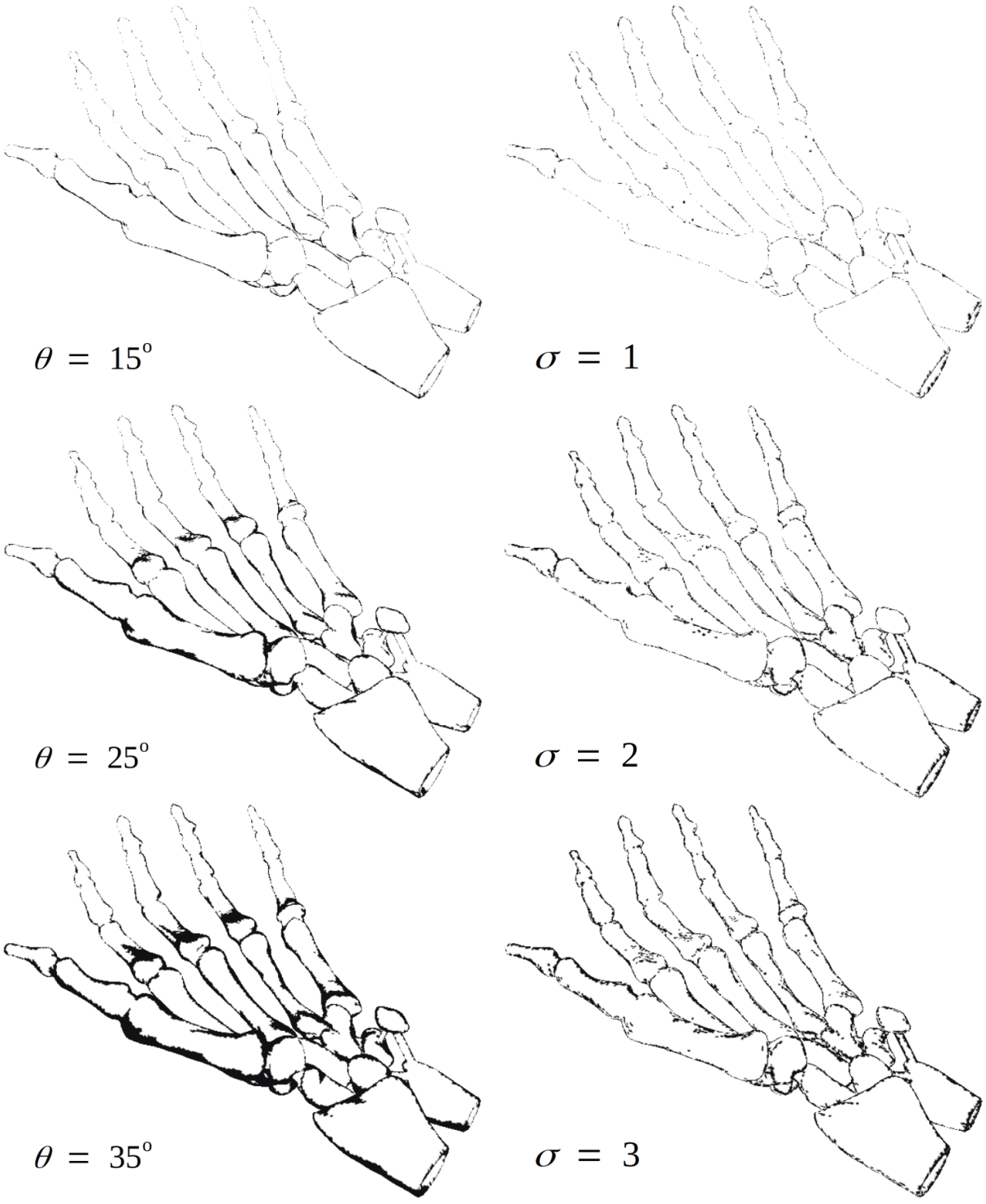
resultando em uma nuvem com aproximadamente 20 mil splats. Note que, mesmo com alta densidade de amostras, a nuvem de splats quase não consegue remover totalmente os artefatos em regiões com alta curvatura como no dente e abaixo da mandíbula do modelo.

A Figura 43 mostra a correlação entre o quadric splat e o recorte curvo. Na parte inferior do modelo *Fandisk*, uma aresta separa uma superfície plana de uma superfície curva. Os splats presentes na superfície plana são recortados apropriadamente pelas curvas de recorte como visto na seção anterior. Entretanto, os splats na superfície curva são recortados por retas, uma vez que a superfície que os intercepta é plana. As junções entre esses splats apresentam alguns artefatos devido a essa diferença (Figura 43c). Para evitar esse tipo de problema, o quadric splat pode ser utilizado nas amostras em que os splats são recortados e também na região da silhueta (Figura 43d). As junções entre os splats em lados opostos da aresta continuarão a não ser justapostas, mas sua proximidade será consideravelmente maior diminuindo assim esses artefatos.

5.5 Detecção de Silhueta

A maioria dos métodos de detecção de pontos localizados próximo da silhueta de um modelo são baseados em espaço de imagem. Entretanto, para a finalidade desta tese, é necessário que se determine se um splat está na silhueta antes de rasterizá-lo. Ou seja, é necessário verificar se uma amostra está na silhueta em espaço de objeto. A maioria das técnicas desse grupo possuem alguma semelhança com a técnica proposta por Xu *et al.* (2004). Por esse motivo, nas

Figura 44 – Comparação entre métodos de detecção de silhueta com diferentes parâmetros aplicados sobre o modelo Hand com amostragem uniforme (47 mil splats). (a) Método de limiarização por ângulo do vetor normal com vetor câmera. (b) Método proposto utilizando a curva de silhueta sobre a superfície quadrática associada ao quadric splat.



Fonte: o autor.

figuras 4 e 44, o método de detecção de silhueta proposto é comparado com o de limiarização de vetores normais. Como dito anteriormente, limiarização de vetores normais através de seu ângulo com o vetor-direção da câmera pode ao mesmo tempo detectar splats demais em regiões de baixa curvatura e não detectar splats em regiões de alta curvatura. Ou seja, não é possível determinar um simples limiar adequado para corrigir ambos os tipos de erros. Pode-se visualizar esse problema na Figura 44a. Ao escolher um ângulo pequeno, splats localizados em regiões de alta curvatura, como nas pontas dos dedos, não são detectados. Ao aumentar o ângulo limite entre o vetor normal e o vetor-direção da câmera, mais splats são detectados, entretanto, isso causa uma detecção exagerada em outras regiões.

Splats carregam consigo mais informações sobre a superfície que representam do que pontos puros. Por exemplo, em amostragens adaptativas, a extensão de um splat com relação ao tamanho médio das amostras pode indicar um conhecimento da curvatura local da superfície. Os mapas de normais associados a cada splat, necessários para a renderização Phong Splatting, trazem bastante informação sobre a geometria local da superfície porque ela foi construída a partir de diversos pontos durante o processo de amostragem. A vantagem de utilizar mapas de normais na detecção de splats localizados próximos da silhueta está na sua descrição concisa. O tamanho fixo dos mapas de normais associados aos quadric splats permite uma implementação fácil em vertex shader, diferente de métodos que necessitam de vizinhança ou outras informações topológicas. A Figura 44b mostra a detecção de splats presentes na silhueta através do método proposto e descrito na Seção 4.4. A abordagem proposta detecta quadric splats como pertencentes à silhueta se parte de sua extensão é visível e parte não é. Splats com maior curvatura e próximos da silhueta são mais suscetíveis a serem detectados através dessa abordagem. Isso é vantajoso, uma vez que renderizar esses splats por amostras planas acarretaria artefatos mais perceptíveis do que os gerados por splats localizados em regiões de baixa curvatura. Entretanto, alguns quadric splats próximos da silhueta podem ter toda sua extensão visível, fazendo com que sejam renderizados de forma plana e assim acrescentando artefatos na silhueta do modelo. Estes quadric splats serão denominados aqui por falso-negativos e são mais comuns quando se utiliza o fator de escala $\sigma = 1$. Aumentar o valor de σ aumenta a quantidade de splats detectados como próximos da silhueta. Essa ação reduz a quantidade de falsos positivos, alcançando alguns splats próximos da silhueta que se renderizados usando splat plano acarretarão na aparência de artefatos. Provavelmente, alguns desses splats detectados não causariam artefatos se renderizados pelo método original. Esses casos serão denominados aqui por falsos-positivos. A ocorrência de

Figura 45 – Comparação entre métodos de detecção de silhueta com mesmos parâmetros, mas aplicados sobre modelos com amostragens diferentes. (a) Modelo *Max Plank* com aproximadamente 14 mil splats. (b) Modelo *Max Plank* com aproximadamente 100 mil splats.

amostragem | limiarização de vetores normais | deteção com quadric splats

(a)

amostragem | limiarização de vetores normais | deteção com quadric splats

(b)

Fonte: o autor.

falsos positivos não prejudica a qualidade da imagem, mas reduz a eficiência da renderização, porque são quadric splats rasterizados que apresentam mínima melhoria gráfica. Entretanto, a ocorrência de falsos-positivos no método proposto é mais aceitável do que a limiarização de vetores normais (Figura 44b), tornando este parâmetro mais fácil de manipular pelo usuário.

Outra característica do método de detecção de silhueta proposto é sua relação com o tamanho das amostras. Métodos que utilizam apenas o vetor normal para classificar splats

pertencentes à silhueta estão fadados a determinar regiões próximas de silhueta, independentemente dos tamanhos das amostras e de sua densidade local. Nota-se isto na Figura 45. O método de limiarização de vetores normais classifica os splats presentes na silhueta do Modelo *Max Plank* com duas densidades diferentes e visualizado a partir do mesmo ponto de câmera. Percebe-se que as regiões detectadas são muito similares. Dessa forma, no modelo mais denso, muitos falso-positivos foram detectados. Já no método proposto, a extensão dos splats é levada em consideração durante a etapa de classificação. Visualmente, isto torna as linhas das silhuetas mais estreitas, por causa dos splats menores, mas evita a adição de falsos-positivos, detectando apenas os splats presentes na silhueta.

5.6 Tempos de renderização

Tabela 3 – Tempos de renderização. (NS: Número de splats; NF: Número de fragmentos; FPS: Frames por segundo; SS: Splats na Silhueta)

		Flat		Quadric ($\sigma = 1$)		
Model	NS	NF	FPS	SS	NF	FPS
Armadillo	36,6k	776k	177,12	7,7k	974k	174,16
Dragon	19,7k	736k	188,84	3,9k	900k	186,51
Dragon	57,0k	739k	176,84	9,6k	857k	174,45
Fertility	10,3k	934k	183,13	1,4k	1061k	180,78

		Quadric ($\sigma = 2$)			Quadric ($\sigma = 3$)		
Model	NS	SS	NF	FPS	SS	NF	FPS
Armadillo	36,6k	13,6k	1102k	171,34	17,7k	1186k	169,89
Dragon	19,7k	6,7k	1008k	184,73	8,7k	1084k	183,44
Dragon	57,0k	17,3k	953k	172,43	23,5k	1027k	170,60
Fertility	10,3k	2,5k	1157k	180,74	3,4k	1235k	179,39

Fonte: dados da pesquisa.

A Tabela 3 compara os tempos de renderização usando splats planos e quadric splats com diferentes parâmetros para detecção da silhueta. Todos os valores na tabela são médias de medições computadas após várias rotações do modelo de forma a pegar todas as nuances das suas formas. Os experimentos utilizados para comparar eficiência da renderização foram realizados em um computador equipado com um processador quadcore IntelCore i7 2.7GHz, 16GB de RAM e uma placa gráfica NVIDIA GeForce 940MX usando OpenGL Shading Language.

Na técnica de surface splatting, as duas primeiras etapas (visibilidade e atributos) compreendem as principais influências no tempo de renderização de um modelo baseado em splats e, como dito na Seção 5.1, o tempo de execução dessas etapas dependem primariamente do tempo gasto no vertex shader e no fragment shader. O tempo gasto no vertex shader é diretamente proporcional ao número de splats, e o tempo gasto no fragment shader é diretamente proporcional ao número de fragmentos gerados pelas primitivas. Normalmente, quando o número de splats diminui, a velocidade de renderização aumenta. Entretanto, quando a densidade de pontos diminui a partir de um certo valor, o tempo de renderização permanece quase o mesmo ou mesmo aumenta um pouco porque os tamanhos das projeções dos splats em espaço de imagem faz com que a quantidade de fragmentos gerados torne o fragment shader o gargalo da pipeline de renderização. Esse efeito pode ser observado no modelo *Fertility*, onde, apesar da quantidade baixa de splats, o tempo de renderização não é reduzido proporcionalmente. Uma quantidade maior de fragmentos é gerada durante a rasterização, acarretando uma maior quantidade de operações de ray casting. Várias dessas operações serão desperdiçadas uma vez que fragmentos são descartados quando o ray casting resultar em pontos fora das dimensões do splat.

Encontrar a amostragem que produza o melhor desempenho não é uma tarefa fácil. Por exemplo, com splats planos, a velocidade de renderização do modelo *Fertility* – modelo com maior número de splats do que o modelo *Armadillo* – é superior àquela do modelo *Armadillo* – modelo com maior quantidade de fragmentos gerados. Entretanto, o modelo *Fertility* possui um tempo de renderização inferior ao do modelo *Dragon* com aproximadamente 20 mil splats, porque a combinação número de primitivas e número de fragmentos do modelo *Dragon* é mais favorável.

De maneira geral, a renderização usando quadric splats é mais lenta do que a renderização usando splats planos. Primeiro, porque o cálculo de ray casting de um plano exige menos operações matemáticas do que o ray casting de uma superfície quadrática. Segundo, porque a detecção de silhueta adiciona alguns splats na rasterização de quadric splats que seriam descartados pela remoção de superfícies ocultas quando usando apenas o centro do splat como parâmetro (Figura 35). Um fator correlacionado a esse segundo motivo é o parâmetro σ. Quanto maior for o valor de σ, mais splats são detectados como pertencentes à silhueta e mais splats são rasterizados no fragment shader, aumentando o trabalho de renderização e diminuindo a taxa de frames.

Apesar dessa perda de eficiência, a taxa de decréscimo é baixa. Assim, considerando a melhor qualidade visual obtida, o custo-benefício final é favorável. Além disso, renderizar com menos amostras mantendo a qualidade de renderização permite, de modo geral, alcançar uma eficiência ligeiramente melhor. Considere, por exemplo, o modelo *Dragon* com diferentes amostragens ilustrado na Figura 42. A versão desse modelo com aproximadamente 20 mil quadric splats apresenta qualidade de renderização similar à versão com aproximadamente 57 mil splats planos. Entretanto, a velocidade de renderização do modelo menos amostrado utilizando quadric splats é maior do que a velocidade de renderização do modelo com 57 mil splats planos, mesmo quando os splats planos são renderizados por uma técnica mais simples. Além disso, a memória usada para armazenar o modelo com 20 mil quadric splats é 34,6% daquela usada para armazenar o modelo com 57 mil splats planos.

5.7 Considerações finais

Essa seção apresentou os resultados obtidos com as técnicas propostas e apresentadas nos capítulos anteriores. As melhorias apresentam-se primariamente na melhoria de qualidade visual nas renderizações utilizando recorte curvo e quadric splats. A melhoria na eficiência da renderização é derivada da redução da quantidade de amostras que a técnica permite utilizar sem haver aparecimento de artefatos visuais muito perceptíveis. Como o recorte de splats e os quadric splats são rasterizados a nível de fragmento, elas mantém suas características até mesmo em visualizações bastante próximas da câmera.

Apesar das melhorias, as técnicas são propostas especialmente para modelos estáticos. Isso se deve ao alto tempo de computação durante a fase de preprocessamento do modelo, onde o erro de aproximação, as curvas de recorte e os mapas de normais dos quadric splats são calculados. Computar esses valores é um procedimento ainda caro computacionalmente, tornando inviável a renderização em tempo real de modelos que modificam sua geometria e topologia no decorrer do tempo segundo as técnicas propostas nesse trabalho.

6 CONCLUSÃO

Normalmente, as placas gráficas modernas podem renderizar rapidamente modelos com quantidades enormes de elementos. Entretanto, o custo de processamento de um modelo ainda é proporcional ao número de primitivas usadas para representá-lo. Dessa forma, reduzir a quantidade das amostras usadas para representar uma geometria amostradas por splats é tão importante quanto reduzir o número de elementos triangulares da malha usada para representar essa geometria. Por esse motivo, o foco deste trabalho foi a renderização de alta qualidade de modelos, com ou sem arestas, mas com baixa densidade de amostras. Nesta tese, formas não lineares foram utilizadas para recortar splats localizados próximo de arestas – os *recorte curvo* –, e para rasterizar o próprio splat – os *quadric splat* – em regiões próximas da silhueta do modelo.

Os modelos baseados em splat que utilizam baixa amostragem necessitam de amostras maiores para sobrepor os espaços entre as primitivas que representam uma superfície. Entretanto, nessas situações artefatos em torno de arestas tornam-se muito visíveis devido à natureza circular do splat. A abordagem proposta nesta tese de utilizar uma curva racional de Bézier para recortar splats localizados próximos de arestas mantém uma estrutura de dados consistente, permitindo uma fácil implementação em GPU. Além disso, essa abordagem alcança qualidade de renderização em regiões problemáticas bastante superior àquelas alcançadas por abordagens com recortes lineares. A curva de recorte de cada splat não é inteiramente consistente com as curvas de recorte de splats vizinhos. Técnicas que representam arestas de maneira global (ZHANG; KAUFMAN, 2007) são mais complexas de implementar em hardware gráfico, e a qualidade não é tão superior para justificar sua utilização.

Usuários notam com mais facilidade a baixa resolução de um modelo amostrado por primitivas lineares nos entornos da silhueta do objeto, uma vez que, nessas regiões, texturas e sombreamento não conseguem disfarçar a natureza do modelo. Os quadric splats propostos neste trabalho têm por objetivo reduzir essa percepção ao serem rasterizados como uma primitiva não linear. Assim como no recorte curvo, sua estrutura é consistente e compacta, visando a implementação em GPU. Visto que os splats lineares renderizados com a técnica de Phong Splatting apresentam resultados de qualidade satisfatória nas regiões do modelo afastadas da silhueta, os quadric splats são empregados preferencialmente na vizinhança da silhueta. Para isso, foi desenvolvido um método para detectar, em espaço de objeto, se um quadric splat pertence à silhueta. Essa detecção pode então ser realizada em vertex shader, e como em modelos e baixa resolução, a quantidade de primitivas é bastante inferior à quantidade de pixels da imagem,

essa etapa a mais na pipeline de rasterização não causa grande influência na velocidade de renderização. Como a técnica de surface splatting precisa de duas etapas de rasterização (etapa de visibilidade e etapa de atributos), essa detecção de silhueta é realizada duas vezes. Em um possível trabalho futuro, essa detecção pode ser armazenada na etapa de visibilidade para evitar ser recalculada na etapa de atributos.

A qualidade visual da renderização de um modelo baseado em splats depende muito da amostragem do modelo, especialmente da distribuição das amostras sobre a superfície. Como discutido na Seção 4.2, utilizar métodos de amostragem tradicionais, com splats planos, pode fazer com que quadric splats não sejam posicionados onde eles melhor se adéquem. A abordagem proposta naquela seção melhora um pouco essa situação, mas utiliza muito processamento de CPU, uma vez que para cada ponto da nuvem, uma quadrática é adaptada somente para calcular seu erro local. Um possível trabalho futuro é tornar essa computação de erro mais simples para permitir o posicionamento mais eficiente dos quadric splats. O mesmo pode ser dito levando em consideração o recorte curvo do splat, e então posicionar splats em torno de arestas em pontos que apresentem o menor erro de recorte possível.

Problemas ainda ocorrem próximo de regiões onde os recortes continuam sendo realizados por retas. Os cantos são representados com recorte dois recortes lineares, usando o polígono de controle, então os splats nos entornos dessa região devem ser pequenos caso os entornos do canto sejam arestas curvas. Outra característica que uma aresta pode possuir e que não pode ser representada pelo recorte proposto é possuir concavidade e convexidosto é possuir concavidade e convexidade ao mesmo tempo. A curva de recorte proposta não possui pontos de inflexão, então a única solução atual para esses problemas é o aumento local da densidade de splats. Situação similar ocorre na construção de quadric splats. Uma superfície quádrica não pode representar uma variação na concavidade da superfície.

A detecção de silhueta proposta apresenta uma heurística que captura splats localizados na silhueta sob diversas condições de curvatura e possui um controle melhor do que as abordagens por limiarização de vetores normais. Entretanto, diversos falsos positivos ainda ocorrem, especialmente em regiões de alta curvatura e côncavas. Como trabalho futuro, propõe-se um estudo mais aprofundado para saber quando ocorre uma diferença grande da rasterização de um splat plano e um quadric splat dependendo de sua curvatura, extensão e, principalmente, orientação. Unindo isso a técnicas em espaço de imagem, é possível obter renderizações estilizadas de silhuetas superiores e independentes da densidade de amostras do modelo.

REFERÊNCIAS

ADAMS, B.; DUTRÉ, P. Interactive boolean operations on surfel-bounded solids. **ACM Transactions on Graphics**, Association for Computing Machinery, New York, v. 22, n. 3, p. 651–656, 2003.

AKENINE-MöLLER, T.; HAINES, E.; HOFFMAN, N.; PESCE, A.; IWANICKI, M.; HILLAIRE, S. **Real-time rendering**. Boca Raton: Taylor & Francis, CRC Press,, 2018.

ALEXA, M.; BEHR, J.; COHEN-OR, D.; FLEISHMAN, S.; LEVIN, D.; SILVA, C. T. Point set surfaces. In: **Proceedings of the Conference on Visualization '01**. San Diego: IEEE Computer Society, 2001. p. 21–28.

BOLLA, N. K.; NARAYANAN, P. J. Algebraic splats representation for point based models. In: **2008 Sixth Indian Conference on Computer Vision, Graphics & Image Processing**. Bhubaneswar: IEEE Computer Society, 2008. p. 71–78.

BOTSCH, M.; HORNUNG, A.; ZWICKER, M.; KOBBELT, L. High-quality surface splatting on today's gpus. In: **Proceedings of the Second Eurographics / IEEE VGTC Conference on Point-Based Graphics**. Stony Brook: Eurographics Association, 2005. p. 17–24.

BOTSCH, M.; KOBBELT, L. High-quality point-based rendering on modern gpus. In: **Proceedings of the 11th Pacific Conference on Computer Graphics and Applications**. Canmore: IEEE Computer Society, 2003. p. 335–343.

BOTSCH, M.; SPERNAT, M.; KOBBELT, L. Phong splatting. In: **Proceedings of the First Eurographics Conference on Point-Based Graphics**. Goslar: Eurographics Association, 2004. p. 25–32.

DACHSBACHER, C.; VOGELGSANG, C.; STAMMINGER, M. Sequential point trees. **ACM Transactions on Graphics**, Association for Computing Machinery, New York, v. 22, n. 3, p. 657–662, 2003.

DANIELS, J. I.; OCHOTTA, T.; HA, L. K.; SILVA, C. T. Spline-based feature curves from point-sampled geometry. **The Visual Computer**, Springer-Verlag, Heidelberg, v. 24, p. 449–462, 2008.

FLEISHMAN, S.; COHEN-OR, D.; ALEXA, M.; SILVA, C. T. Progressive point set surfaces. **ACM Transactions on Graphics**, Association for Computing Machinery, New York, v. 22, n. 4, p. 997–1011, 2003.

GROSS, M.; PFISTER, H. **Point-Based Graphics**. San Francisco: Morgan Kaufmann Publishers Inc., 2007.

GROSSMAN, J. P.; DALLY, W. J. Point sample rendering. In: **Proceedings of Eurographics Workshop on Rendering 98**. Vienna: Springer Vienna, 1998. p. 181–192.

GUENNEBAUD, G.; BARTHE, L.; PAULIN, M. Splat/mesh blending, perspective rasterization and transparency for point-based rendering. In: **Proceedings of the 3rd Eurographics / IEEE VGTC Conference on Point-Based Graphics**. Boston: Eurographics Association, 2006. p. 49–57.

GUENNEBAUD, G.; GERMANN, M.; GROSS, M. H. Dynamic sampling and rendering of algebraic point set surfaces. **Computer Graphics Forum**, Wiley-Blackwell, Oxford, v. 27, n. 2, p. 653–662, 2008.

GUENNEBAUD, G.; PAULIN, M. Efficient screen space approach for hardware accelerated surfel rendering. In: **Proceedings of Vision, Modeling and Visualization**. Munich: IEEE Computer Society, 2003. p. 485–495.

GUMHOLD, S. Splatting illuminated ellipsoids with depth correction. In: **Proceedings of the Vision, Modeling, and Visualization Conference 2003**. Munich: Aka GmbH, 2003. p. 245–252.

GUMHOLD, S.; WANG, X.; MACLEOD, R. Feature extraction from point clouds. In: **Proceedings of the 10th International Meshing Roundtable**. Newport Beach: Sandia National Laboratories, 2001. p. 293–305.

HERTZMANN, A. Introduction to 3d non-photorealistic rendering: Silhouettes and outlines. In: **Non-Photorealistic Rendering, SIGGRAPH Course Notes**. Los Angeles: Association for Computing Machinery, 1999.

INSAFUTDINOV, E.; DOSOVITSKIY, A. Unsupervised learning of shape and pose with differentiable point clouds. In: **Proceedings of the 32nd International Conference on Neural Information Processing Systems**. Montreal: Curran Associates Inc., 2018. p. 2807–2817.

ISENBERG, T.; GUERICKE, O. von; FREUDENBERG, B.; HALPER, N.; SCHLECHTWEG, S.; STROTHOTTE, T. A developer's guide to silhouette algorithms for polygonal models. **IEEE Computer Graphics and Applications**, IEEE Computer Society Press, Washington, v. 23, p. 28–37, 2003.

IVO, R. F.; VIDAL, C. A.; NETO, J. B. C. A method for clipping splats on sharp edges and corners. **The Visual Computer**, Springer-Verlag, Inc., Heidelberg, v. 28, p. 995–1004, 2012.

KALAIAH, A.; VARSHNEY, A. Modeling and rendering of points with local geometry. **IEEE Transactions on Visualization and Computer Graphics**, IEEE Educational Activities Department, Piscataway, v. 9, n. 1, p. 30–42, 2003.

KLEIN, T.; ERTL, T. Illustrating magnetic field lines using a discrete particle model. In: **Proceedings of the Vision, Modeling, and Visualization Conference 2004**. Stanford: Aka GmbH, 2004. p. 387–394.

KOBBELT, L. P.; BOTSCH, M.; SCHWANECKE, U.; SEIDEL, H.-P. Feature sensitive surface extraction from volume data. In: **Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques**. New York: Association for Computing Machinery, 2001. p. 57–66.

KOENDERINK, J. J. What does the occluding contour tell us about solid shape? **Perception**, SAGE Publications, Thousand Oaks, v. 13, n. 3, p. 321–330, 1984.

LEVIN, D. Mesh-independent surface interpolation. In: **Geometric modeling for scientific visualization**. Heidelberg: Springer Berlin Heidelberg, 2003. p. 37–49.

LEVOY, M.; PULLI, K.; CURLESS, B.; RUSINKIEWICZ, S.; KOLLER, D.; PEREIRA, L.; GINZTON, M.; ANDERSON, S.; DAVIS, J.; GINSBERG, J.; SHADE, J. The digital michelangelo project: 3d scanning of large statues. In: **Proceedings of the 27th annual conference on Computer Graphics and Interactive Techniques**. New Orleans: Association for Computing Machinery, 2000. p. 131–144.

LEVOY, M.; WHITTED, T. **The Use of Points As Display Primitive**. Computer Science Department, University of North Carolina at Chapel Hill, 1985.

LIN, C.-H.; KONG, C.; LUCEY, S. Learning efficient point cloud generation for dense 3d object reconstruction. In: **Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence**. New Orleans: AAAI Press, 2018. p. 7114–7121.

LIPMAN, Y.; COHEN-OR, D.; LEVIN, D.; TAL-EZER, H. Parameterization-free projection for geometry reconstruction. **ACM Transactions on Graphics**, Association for Computing Machinery, New York, v. 26, n. 3, p. 22–27, 2007.

LOPER, M. M.; BLACK, M. J. Opendr: An approximate differentiable renderer. In: **Proceedings of Euro Conference on Computer Vision**. Cham: Springer International Publishing, 2014. p. 154–169.

LUEBKE, D.; REDDY, M.; COHEN, J. D.; VARSHNEY, A.; WATSON, B.; HUEBNER, R. **Level of Detail for 3D Graphics**. San Francisco: Morgan Kaufmann Publishers Inc., 2003.

OHTAKE, Y.; BELYAEV, A.; ALEXA, M. Sparse low-degree implicit surfaces with applications to high quality rendering, feature extraction, and smoothing. In: **Proceedings of the Third Eurographics Symposium on Geometry Processing**. Aire-la-Ville: Eurographics Association, 2005.

OHTAKE, Y.; BELYAEV, A.; ALEXA, M.; TURK, G.; SEIDEL, H.-P. Multi-level partition of unity implicits. **ACM Transactions on Graphics**, Association for Computing Machinery, New York, v. 22, n. 3, p. 463–470, 2003.

OLSON, M.; DYER, R.; ZHANG, H.; SHEFFER, A. Point set silhouettes via local reconstruction. **Computers & Graphics**, Elsevier, Amsterdam, v. 35, p. 500–509, 2011.

PAJAROLA, R. Efficient level-of-details for point based rendering. In: **Computer Graphics and Imaging**. Honolulu: IASTED/ACTA Press, 2003. p. 141–146.

PAULY, M.; GROSS, M.; KOBBELT, L. P. Efficient simplification of point-sampled surfaces. In: **Proceedings of the Conference on Visualization '02**. Boston: IEEE Computer Society, 2002. p. 163–170.

PAULY, M.; KEISER, R.; ADAMS, B.; DUTRé, P.; GROSS, M.; GUIBAS, L. J. Meshless animation of fracturing solids. In: **ACM SIGGRAPH 2005 Papers**. New York: Association for Computing Machinery, 2005. p. 957–964.

PAULY, M.; KEISER, R.; GROSS, M. Multi-scale feature extraction on point-sampled surfaces. **Computer Graphics Forum**, Wiley-Blackwell, Oxford, v. 22, p. 281–289, 2003.

PAULY, M.; KEISER, R.; KOBBELT, L. P.; GROSS, M. Shape modeling with point-sampled geometry. **ACM Transactions on Graphics**, Association for Computing Machinery, New York, v. 22, n. 3, p. 641–650, 2003.

PFISTER, H.; ZWICKER, M.; BAAR, J. van; GROSS, M. Surfels: Surface elements as rendering primitives. In: **Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques**. New Orleans: ACM Press/Addison-Wesley Publishing Co., 2000. p. 335–342.

REINA, G.; ERTL, T. Hardware-accelerated glyphs for mono- and dipoles in molecular dynamics visualization. In: **Proceedings of the Seventh Joint Eurographics / IEEE VGTC Conference on Visualization**. Aire-la-Ville: Eurographics Association, 2005. p. 177–182.

ROVERI, R.; ÖZTIRELI, A. C.; PANDELE, I.; GROSS, M. Pointpronets: Consolidation of point clouds with convolutional neural networks. **Computer Graphics Forum**, Wiley-Blackwell, Oxford, v. 37, n. 2, p. 87–99, 2018.

RUSINKIEWICZ, S.; LEVOY, M. Qsplat: A multiresolution point rendering system for large meshes. In: **Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques**. New Orleans: ACM Press/Addison-Wesley Publishing Co., 2000. p. 343–352.

SIGG, C.; WEYRICH, T.; BOTSCH, M.; GROSS, M. Gpu-based ray-casting of quadratic surfaces. In: **Proceedings of the 3rd Eurographics / IEEE VGTC Conference on Point-Based Graphics**. Goslar: Eurographics Association, 2006. p. 59–65.

STOLL, C.; GUMHOLD, S.; SEIDEL, H. Visualization with stylized line primitives. In: **16th IEEE Visualization Conference**. Minneapolis: IEEE Computer Society, 2005. p. 695–702.

STOLL, C.; GUMHOLD, S.; SEIDEL, H.-P. Incremental raycasting of piecewise quadratic surfaces on the gpu. In: **Proceedings of 2006 IEEE Symposium on Interactive Ray Tracing**. Salt Lake City: IEEE Computer Society, 2006. p. 141–150.

WICKE, M.; TESCHNER, M.; GROSS, M. Csg tree rendering for point-sampled objects. In: **Proceedings of the Computer Graphics and Applications, 12th Pacific Conference**. Seoul: IEEE Computer Society, 2004. p. 160–168.

WU, J.; KOBBELT, L. Optimized sub-sampling of point sets for surface splatting. **Computer Graphics Forum**, Blackwell Publishing, Inc, Oxford, v. 23, n. 3, p. 643–652, 2004.

WU, J.; ZHANG, Z.; KOBBELT, L. Progressive splatting. In: **Proceedings of the Second Eurographics / IEEE VGTC Conference on Point-Based Graphics**. Stony Brook: Eurographics Association, 2005. p. 25–32.

XU, H.; NGUYEN, M. X.; YUAN, X.; CHEN, B. Interactive silhouette rendering for point-based models. In: **Proceedings of the First Eurographics Conference on Point-Based Graphics**. ETH Zurich: Eurographics Association, 2004. p. 13–18.

YIFAN, W.; SERENA, F.; WU, S.; ÖZTIRELI, C.; SORKINE-HORNUNG, O. Differentiable surface splatting for point-based geometry processing. **ACM Transactions on Graphics**, Association for Computing Machinery, New York, v. 38, n. 6, 2019.

ZAKARIA, N.; SEIDEL, H.-P. Interactive stylized silhouette for point-sampled geometry. In: **Proceedings of the 2nd International Conference on Computer Graphics and Interactive Techniques in Australasia and South East Asia**. Singapore: Association for Computing Machinery, 2004. p. 242–249.

ZHANG, H.; KAUFMAN, A. E. Point-and-edge model for edge-preserving splatting. **The Visual Computer**, Springer-Verlag, Heidelberg, v. 23, n. 6, p. 397–408, 2007.

ZHANG, L.; SUN, Q.; HE, Y. Splatting lines: An efficient method for illustrating 3d surfaces and volumes. In: **Proceedings of the 18th Meeting of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games**. San Francisco: Association for Computing Machinery, 2014. p. 135–142.

ZWICKER, M.; PFISTER, H.; BAAR, J. van; GROSS, M. Surface splatting. In: **Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques**. New York: Association for Computing Machinery, 2001. p. 371–378.

ZWICKER, M.; RäSäNEN, J.; BOTSCH, M.; DACHSBACHER, C.; PAULY, M. Perspective accurate splatting. In: **Proceedings of Graphics Interface 2004**. Waterloo: Canadian Human-Computer Communications Society, 2004. p. 247–254.