Rafael Fernandes Ivo

Uma Nova Estratégia para Renderizar Descontinuidades e Superfícies Intersectantes em Modelos Baseados em Splats

Fortaleza

2011

Rafael Fernandes Ivo

Uma Nova Estratégia para Renderizar Descontinuidades e Superfícies Intersectantes em Modelos Baseados em Splats

Dissertação apresentada ao Departamento de Computação da Universidade Federal do Ceará como requisito parcial para obtenção do título de Mestre em Ciência da Computação.

Orientador:
Creto Augusto Vidal

Co-orientador:
Joaquim Bento Cavalcante Neto

UNIVERSIDADE FEDERAL DO CEARÁ
DEPARTAMENTO DE COMPUTAÇÃO
MESTRADO E DOUTORADO EM CIÊNCIA DA COMPUTAÇÃO
CRAB - COMPUTAÇÃO GRÁFICA, REALIDADE VIRTUAL E ANIMAÇÃO

Fortaleza

2011

Agradecimentos

Aos meus orientadores Creto Vidal e Joaquim Bento pelo apoio, empenho e ideias que tornaram este trabalho possível.

Ao CNPq pelo apoio financeiro.

Aos colegas do CRAb que deram boas dicas e apoio.

Aos meus pais, que me deram total suporte para conseguir chegar aqui, e aos meus irmãos, que me alegram o dia.

À minha namorada, que me deu força para continuar mesmo sabendo que tínhamos que nos afastar por um tempo para terminar a dissertação.

Resumo

Modelos baseados em splats têm ganhado crescente atenção devido a seu potencial para renderizações de modelos geométricos complexos de forma eficiente e com alta qualidade. A ausência de informações de conectividade desses modelos permite operações de modelagem complexas, como operações booleanas, e fraturas em simulações físicas. Entretanto, essas operações geralmente geram modelos com arestas e cantos que não podem ser representados corretamente com um número finito de splats sem que um tratamento seja feito. Neste trabalho, um grafo de vizinhança utiliza uma estimativa que garante a conexão de todos os splats presentes em lados opostos de uma descontinuidade e que precisam ser recortados uns contra os outros. Após utilizar um método de detecção de descontinuidades no grafo gerado, os vizinhos que participam do recorte de um splat, os *clip partners*, são determinados e classificados para que recortem o splat de forma a adaptá-lo à curva da descontinuidade. Outro problema encontrado na renderização de modelos baseados em splats é reconstrução de superfícies intersectantes. Nas proximidades de interseções de superfícies, as superfícies são misturadas, resultando em artefatos. Para tratar esses casos, um algoritmo de segmentação realiza a separação das diversas superfícies presentes no modelo, identificando os splats que as formam e impedindo que eles sejam combinados em áreas próximas de interseções de superfícies na etapa de reconstrução da superfície em espaço de imagem.

Sumário

Lista de Figuras

Lista de Tabelas

1 Introdução

Nos últimos anos, as pesquisas voltadas para representação, modelagem, processamento e renderização eficiente de geometrias amostradas por pontos têm estado em foco. Há dois principais motivos para esse interesse: o crescente aumento da complexidade poligonal dos modelos gráficos e a popularização de scanners 3D. O alto custo envolvido na manipulação e na armazenagem de malhas excessivamente grandes, às vezes, não é justificável quando se percebe que regiões inteiras dessas malhas contribuem apenas para um pixel da imagem renderizada. Scanners 3D não só se tornaram ferramentas mais populares e poderosas nos últimos anos, como também mais acessíveis. Essa disponibilidade fez com que modelos 3D estejam sendo, cada vez mais, adquiridos ao invés de modelados.

1.1 Contextualização do Trabalho

A Figura 1.1 resume a pipeline de criação de conteúdo 3D. A etapa de aquisição envolve um processo físico para registrar objetos tridimensionais. Para capturar integralmente o objeto, o processo de aquisição de modelos 3D geralmente se realiza através de múltiplos registros pelo equipamento. O resultado é uma amostragem discreta do objeto físico. Às vezes, no caso de superfícies convolutas ou com muitas concavidades, também se faz necessário o preenchimento

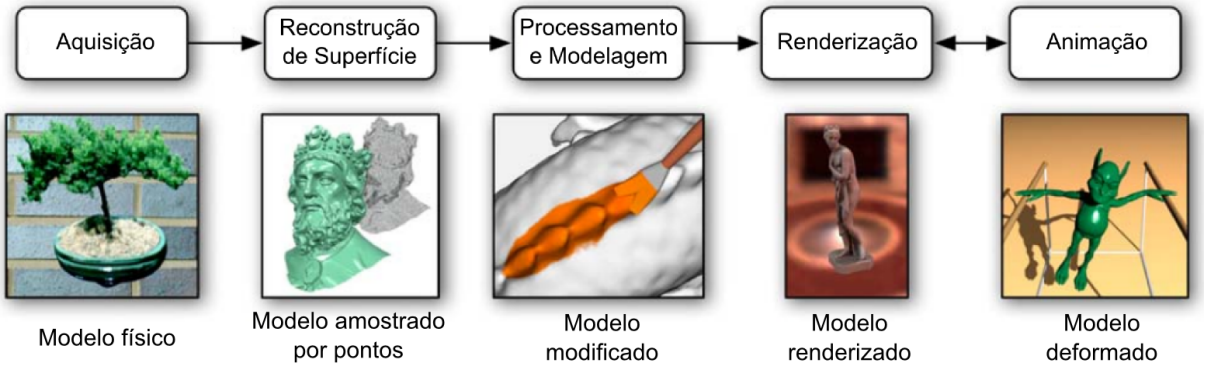

Figura 1.1: Pipeline de criação de conteúdo 3D.

de buracos não registrados pelo equipamento.

A conversão de uma nuvem de pontos amostrais de uma superfície para uma representação matemática contínua é chamada de reconstrução de superfície. Essa representação pode ser uma malha triangular, uma coleção de *patches* paramétricos ou o conjunto de zeros de uma função no espaço. O principal desafio dessa transformação é, a partir de um conjunto de amostras espalhadas, identificar características tais como curvatura, arestas, cantos e interseções de superfícies, sabendo que essas amostras podem conter *outliers* (pontos muito distantes da superfície, decorrentes de erros grosseiros no processo de aquisição) ou ruídos. Assim, para realizar uma boa representação da superfície e detectar todas essas características na superfície desconhecida, é necessário um método robusto para estimar os vetores normais das amostras. As normais podem ser calculadas de maneira procedimental para objetos simples ou estimadas através de métodos estatísticos robustos que preservam inclusive cantos e arestas [Li et al. 2010]. Nesta dissertação, é proposta uma técnica para detecção automática de arestas e cantos, utilizando amostras munidas de normais. Por esse motivo, uma boa estimativa desses vetores normais é essencial para o bom funcionamento da técnica.

O processamento e a modelagem abrangem uma série de transformações para analisar ou modificar o modelo. O processamento inclui operações tais como filtragem, compressão, segmentação, extração de características e reamostragem (refinamento ou redução), entre outras. A modelagem visa editar a aparência e a forma do modelo como, por exemplo, deformações a mão livre, pinturas ou texturizações de superfícies e operações de extrusão ou booleanas (CSG). Nessa etapa, as amostras são adaptadas às arestas e cantos detectados na etapa anterior sem reamostragem do modelo e sem qualquer informação de modelagem. Entretanto, o modelo é segmentado em diversas superfícies suaves e contínuas.

O estágio final da pipeline refere-se à visualização do modelo. A renderização de pontos tem recebido muita atenção nos últimos anos e os algoritmos propostos podem ser classificados de acordo com a reconstrução da superfície utilizada. Pelo menos cinco diferentes abordagens têm sido propostas:

- **Detecção de buracos e preenchimento em espaço de imagem.** Amostras individuais são projetadas na tela e os pixels que não receberam amostras são detectados. A superfície é então interpolada a partir das amostras vizinhas.
- **Geração de mais amostras.** Uma superfície é adaptativamente interpolada no espaço de objeto para garantir que cada pixel receba pelo menos uma amostra.
- **Splatting.** Uma amostra da superfície é projetada na tela e sua contribuição é espalhada

sobre os pixels vizinhos para garantir a cobertura. Métodos de alta qualidade ponderam as contribuições de todos os splats para um determinado pixel.

- **Tecelagem.** Uma malha poligonal é usada para interpolar as amostras da superfície. Isso é bastante caro, pois os buracos, geralmente, abrangem apenas poucos pixels, e um algoritmo completo de renderização de polígonos é necessário.

- **Ray Tracing.** Raios são lançados a partir do observador através da tela, e verificam-se as interseções com alguma representação do modelo, como splats ou superfícies implícitas. Apesar de enquadrar-se entre as técnicas mais caras, apresentam efeitos interessantes tais como reflexão, refração, sombras projetadas, etc.

Este trabalho enquadra-se no grupo que utiliza a técnica de *Surface Splatting* proposta por Zwicker *et al*. [Zwicker et al. 2001]. Essa técnica é uma das mais utilizadas para renderização de nuvens de pontos, e devido ao seu tratamento de pontos como discos em espaço de objeto denominados splats, esses discos passaram a ser utilizados como uma nova forma de representar superfícies. Essa nova representação é utilizada neste trabalho por permitir uma série de simplificações com relação aos métodos tradicionais de reconstrução de geometrias baseadas em pontos e por reunir vantagens de geometrias baseadas em pontos e malhas poligonais.

Além da renderização, modelos podem precisar ser animados, ou seja, sua forma e seus atributos são controlados e alterados através do tempo. As animações fisicamente realistas, utilizando representações amostradas por pontos, emergiram recentemente como uma alternativa promissora às simulações convencionais de elementos finitos. Essas técnicas são inspiradas pelos métodos *meshless*, onde o *continuum* é discretizado usando amostras pontuais desestruturadas. Como a maioria dos trabalhos dessa área diferenciam os pontos do modelo em *phyxels* (*physics elements*) e *surfels* (*surface elements*), a renderização pode ser separada da simulação, permitindo que as técnicas propostas neste trabalho sejam utilizadas para renderizar quebras e fraturas decorrentes dessas animações.

1.2 Motivação

Representações de superfícies baseadas puramente em pontos correspondem a funções de interpolação constantes seccionadas. Segundo a teoria da aproximação, o erro de aproximação nesse caso depende de derivadas de primeira ordem da superfície representada. Geometricamente falando, o erro de aproximação entre a superfície contínua e o conjunto discreto de pontos é limitado pela distância geodésica entre os pontos. Para reduzir o erro por um fator de dois, é

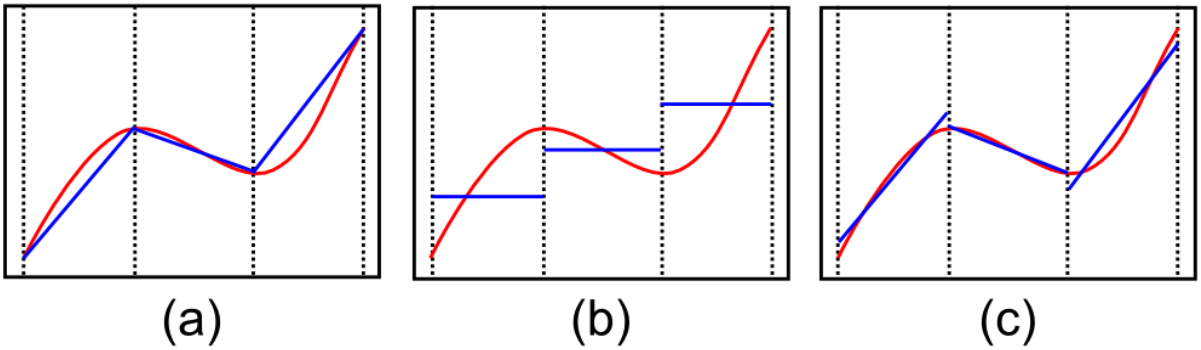

Figura 1.2: Comparação de diferentes aproximações de forma: (a) poligonos - lineares - C^0; (b) pontos - constantes - C^{-1}; (c) splats - lineares - C^{-1}.

necessário aumentar o número de amostras por um fator de quatro.

Para preencher os espaços entre as amostras mais eficientemente, representações baseadas em pontos são normalmente estendidas a representações baseadas em *splats* [Zwicker et al. 2001], onde a superfície é localmente aproximada por um pequeno disco ou elipse. Isso é de especial interesse no contexto de renderização, foco deste trabalho. Como esses discos proveem uma aproximação de primeira ordem à superfície (posição e normal), o erro de aproximação agora depende das derivadas de segunda ordem. Exatamente como outras representações de superfícies seccionadas linearmente, podem-se utilizar splats grandes em áreas planas da superfície e uma alta densidade de amostragem com splats menores em regiões de alta curvatura. Como consequência, o erro de aproximação pode ser reduzido por um fator de dois apenas dobrando o número de splats, que tem a mesma convergência de malhas triangulares. A principal diferença deve-se ao fato de que representações baseadas em splats não definem uma superfície contínua e, dessa forma, cada splat permanece um objeto geométrico independente, e a simplicidade conceitual das representações baseadas em pontos é preservada (Figura 1.2) [Gross e Pfister 2007].

Para muitas aplicações, a habilidade de renderizar regiões de superfícies onde a continuidade é C^0, tais como arestas e cantos, é um requerimento. Essas descontinuidades aparecem com frequência em modelos gerados através de operações booleanas (CSG) [Adams e Dutré 2003, Pauly et al. 2003] e em simulações físicas de rachaduras e quebra de materiais [Pauly et al. 2005]. Devido à natureza dos splats, seria necessário uma quantidade infinita de splats para representar arestas perfeitamente. Mas, assim como malhas poligonais (Figura 1.3), conhecendo-se a localização dessas descontinuidades é possível adaptar os splats, alterando a amostragem, rearranjando as amostras ou adicionando recortes. Neste trabalho, as amostras já existentes são adaptadas através de recortes para que representem de forma correta arestas e cantos. Para realizar essa tarefa, heurísticas são utilizadas a fim de detectar quando dois splats

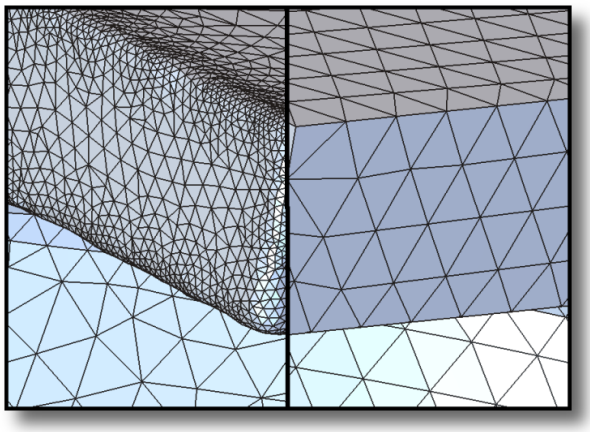

Figura 1.3: Adaptação dos elementos da malha a uma aresta previamente detectada. A mesma ideia pode ser aplicada a representações baseadas em splats.

pertencem a lados opostos de uma aresta. Após essas detecções serem feitas, a melhor forma de recortar os splats é escolhida de modo que os splats recortados adaptem-se corretamente à curva de descontinuidade.

Em modelos baseados em splats, as amostras se sobrepõem para garantir a ausência de buracos na imagem final. Então, cada pixel é associado a uma coleção de fragmentos de splats. Para reconstruir a superfície no espaço de imagem, é necessário encontrar todos os splats que contribuem para uma determinada superfície e então somar e normalizar suas contribuições. Como não há informação explícita de quais fragmentos pertencem a qual superfície, ou mesmo quantas superfícies existem no total, são utilizadas diversas heurísticas. Conceitualmente seria possível identificar as superfícies por algum tipo de identificador, entretanto, realizar a identificação das superfícies através do usuário durante a pipeline do *splatting* limitaria sua aplicabilidade. A ideia proposto é identificar essas superfícies de forma automática por meio de um algoritmo de segmentação através do grafo de relações de vizinhança. Dessa forma, a reconstrução de superfície no espaço de imagem é realizada de forma correta nos locais onde ocorrem interseções de superfícies.

1.3 Contribuições

As contribuições propostas neste trabalho visam preencher os espaços deixados pela técnica de *Surface Splatting* para a renderização correta de descontinuidades como arestas e cantos, sejam estas formadas pela quebra da suavidade da superfície ou por interseção desta com outra superfície. Em síntese, as contribuições deste trabalho são:

- Definição de um novo tipo de relação de vizinhança para modelos baseados em splats de

modo a acelerar o processo de detecção de descontinuidades minimizando a quantidade de cálculos de interseção e recorte de splats.

- Adaptação do método de recorte de splats em modelos CSG proposto por Wicke *et al*. [Wicke et al. 2004], para tratar qualquer modelo sem informação de modelagem.

- Detecção das diversas superfícies do modelo, e identificação, de maneira única, de todos os splats pertencentes a essas superfícies, por meio de buscas no grafo de vizinhança. No momento da reconstrução da modelo em espaço de imagem, os métodos de reconstrução existentes são adaptados de modo a não combinar fragmentos provenientes de superfícies diferentes.

1.4 Organização da Dissertação

Esta dissertação está organizada da seguinte forma:

- *Capítulo 2*:

  Descrevem-se os trabalhos que foram adaptados ou relacionados aos problemas tratados nesta dissertação, tais como o tratamento de arestas e cantos, sua detecção e a etapa de reconstrução de superfície de modelos baseados em splats.

- *Capítulo 3*:

  Discutem-se as diversas estimativas de vizinhança existentes e as comparam com a estimativa proposta. Essas estimativas são importantes para a geração do grafo de vizinhança e a detecção de arestas e superfícies contínuas, objetivo deste trabalho.

- *Capítulo 4*:

  Apresentam-se a detecção de arestas e cantos, os problemas com o recorte de splats e as soluções e adaptações encontradas para evitar o aparecimento de buracos no modelo e o borramento dessas descontinuidades.

- *Capítulo 5*:

  Mostram-se os métodos clássicos de reconstruir a superfície em espaço de imagem a partir do modelo e seus principais problemas e, em seguida, a adaptação proposta a esses métodos utilizando um método de detecção de superfícies contínuas.

- *Capítulo 6*:

O resultado da adição das técnicas propostas na geração de imagens de modelos baseados em splats são apresentados e discutidos, bem como o tratamento das descontinuidades para evitar o borramento e os buracos e a distinção de splats para não misturar superfícies intersectantes.

- *Capítulo 7*:

  Finalmente, o trabalho é concluído e linhas de trabalhos e alguns problemas a serem tratados futuramente são relatados.

2 *Trabalhos Relacionados*

O presente trabalho apresenta contribuições nas diversas etapas da criação de conteúdo gráfico 3D através de modelos baseados em splats. Por este motivo, para fins didáticos, apresenta-se, inicialmente, um resumo do método a ser descrito no restante do trabalho. Esse resumo permite classificar os trabalhos relacionados com cada área da qual se utiliza o trabalho proposto.

Como as contribuições são propostas para modelos baseados em splats, os vetores normais e dimensões das amostras são informações necessárias a priori. Métodos eficientes para o cálculo de vetores normais em conjuntos de pontos que apresentam ruídos, e para o cálculo de arestas foram propostos nos últimos anos [Mitra e Nguyen 2003, Lalonde et al. 2005, Jones et al. 2004, Oztireli et al. 2009, Li et al. 2010]. Vetores normais podem ser calculados, também, de maneira procedimental a partir da modelagem, dependendo da aplicação. Para o cálculo das dimensões dos splats, podem ser utilizadas heurísticas simples tais como a média das distâncias da amostra a seus vizinhos, podendo estas distâncias serem euclidianas, ortogonais ou geodésicas. Métodos mais elaborados como o utilizado no trabalho de Wu e Kobbelt [Wu e Kobbelt 2004], também podem ser empregados. Em todos os modelos apresentados neste trabalho as normais são calculadas através da modelagem e os raios dos splats são iguais à média das distâncias euclidianas da amostra a seus cinco vizinhos mais próximos. Apesar da forma simplista desses cálculos de normais e raios, a única restrição do método proposto é que o modelo apresente normais com baixa presença de ruído e que os splats se sobreponham de tal forma a evitar buracos na imagem gerada.

2.1 Resumo do Método

O método completo divide-se em duas etapas principais: o pré-processamento realizado antes da renderização e o processo durante a renderização do modelo.

1. Antes da renderização:

   (a) Geração do grafo de vizinhança e ponderação de suas arestas.

(b) Limiarização das arestas do grafo em *sharp-edges* e *continuous-edges*.

(c) Adição dos parceiros de recorte (*clip partners*) e de suas classificações.

(d) Agrupamento dos splats em conjuntos de superfícies contínuas.

2. Durante a renderização:

(a) Projeção dos splats em espaço de tela.

(b) Remoção dos fragmentos recortados.

(c) Reconstrução da superfície.

Antes da renderização, cria-se um grafo para armazenar as relações de vizinhança das amostras do modelo. A relação de vizinhança proposta visa identificar splats que devem ser recortados uns contra os outros e splats que se interceptam. Além disso, ela acelera o processo de classificação por buscar apenas as relações de vizinhança necessárias à aplicação estudada neste trabalho. Ao ponderar as arestas do grafo e classificá-las segundo seus pesos, as descontinuidades do modelos são detectadas e tratadas pelo restante do método. A adição e a classificação dos parceiros de recorte de cada amostra buscam adaptar o splat de forma a não cruzar a linha da aresta nem ser recortado desnecessariamente, deixando buracos no modelo renderizado. Por fim, identifica-se cada amostra com um *tag* associado à superfície contínua à qual a amostra pertence. Essa informação adicional é utilizada para melhor reconstruir a da superfície no espaço de tela.

Durante a renderização do modelo, após a projeção de cada amostra sobre a tela e a geração dos fragmentos na grade de pixels da imagem, efetua-se uma varredura nesses fragmentos, removendo-os de acordo com a classificação dos *clip partners* feita na etapa anterior. Por fim, reconstrói-se a superfície ao realizar a média ponderada das contribuições dos fragmentos de determinado pixel que foram gerados a partir de amostras de uma mesma superfície. A identificação de quais fragmentos combinar é dada pelas informações calculadas no passo de classificação de amostras da etapa anterior.

As contribuições apresentadas neste trabalho para a solução do problema proposto, contidas no método descrito sucintamente nesta seção, são: 1) a detecção de arestas e descontinuidades; 2) a representação destas arestas em modelos baseados em splats; 3) a utilização da técnica de segmentação do modelo em diversas superfícies contínuas para realizar uma melhor reconstrução de superfície em espaço de imagem. A seguir são apresentados os trabalhos relacionados a cada uma dessas contribuições.

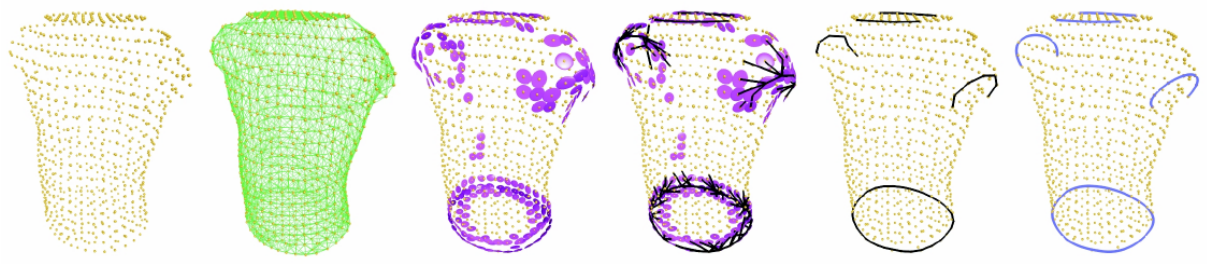

Figura 2.1: (a) Nuvem de pontos entrada. (b) Grafo de Vizinhança. (c) Classificação dos pontos. (d) Limiarização das arestas do grafo. (e) Formação dos padrões de descontinuidade. (f) Representação por spline dos padrões de descontinuidade.

2.2 Detecção de Arestas

O problema de detecção de arestas é bastante relacionado com a reconstrução da superfície. Extrair esse tipo de informação de uma geometria amostrada por pontos pode servir como entrada para muitas aplicações de processamento, tais como a tecelagem de malhas, a segmentação de modelos e a melhor representação de splats localizados próximos às descontinuidades.

Vários trabalhos utilizam a ideia de ponderar as amostras do modelo com alguma função que diga quão próximo um ponto está de uma descontinuidade e então separam um conjunto de amostras com pesos que se destacam das demais. O primeiro trabalho com contribuição significante utilizando essa ideia foi o trabalho de Gumhold *et al.* em [Gumhold et al. 2001]. A ideia básica de seu método é aplicar pesos nos pontos amostrais, onde quanto maior o peso, mais plana é a superfície onde o ponto se encontra. Então constrói-se um grafo de vizinhança e ponderam-se as arestas deste grafo com base nos pesos dos pontos e nas suas características geométricas. Ao extrair um subgrafo que minimize os pesos das arestas, produz-se um conjunto de padrões de descontinuidades. A Figura 2.1 mostra as etapas do trabalho de Gumhold para detecção de quebras de continuidade da superfície. As principais contribuições presentes no artigo que se relacionam com este trabalho são a geração do grafo de vizinhança, a formação do conjunto de vizinhos por amostra e a ponderação de pontos e arestas do grafo.

O grafo de vizinhança é construído de duas formas dependendo da quantidade de ruído presente no modelo amostrado. No caso de dados com pouco ruído, um grafo de vizinhança com uma quantidade relativamente pequena de arestas serve ao propósito de detectar descontinuidades. Para gerar o grafo de vizinhança, é calculada a tetraedralização de Delaunay dos pontos fornecidos e, em seguida, uma série de filtros remove as faces dos tetraedros que são internas ao objeto para que sobrem apenas triângulos sobre a superfície do objeto. Os pontos que compartilham uma aresta nessa triangulação apresentam uma aresta conectando-os no grafo de vizinhança. No caso de dados com muito ruído, a baixa quantidade de arestas causa problemas

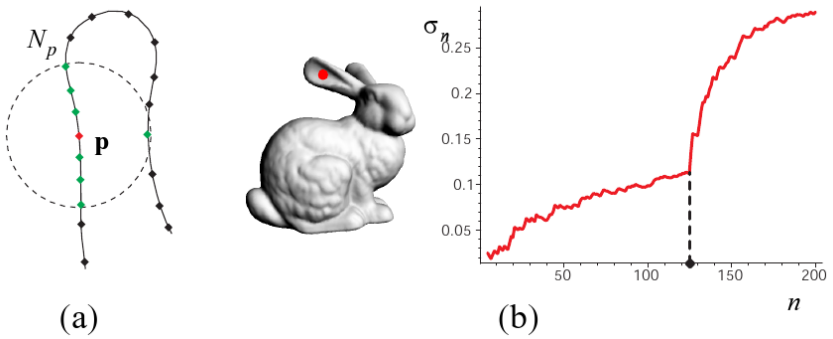

Figura 2.2: (a) Ilustração de uma vizinhança inválida. (b) O tamanho de vizinhança crítico para um ponto na orelha do coelho ocorre no repentino crescimento da curva da variância.

no estágio de detecção de descontinuidades no qual a saída é um subconjunto do grafo de vizinhança. Por esse motivo, nessa ocasião, é utilizado o grafo de Riemann que conecta cada ponto a seus k vizinhos mais próximos. Gumhold utiliza k entre 10 e 16 para atingir bons resultados.

O número de vizinhos reunidos depende do nível de ruído dos dados. Para um conjunto com pouco ou nenhum ruído, apenas os vizinhos diretos são necessários, mas, para dados com muito ruído, devem-se considerar todos os vizinhos com distância no grafo de três, quatro ou cinco arestas. A partir desse conjunto de vizinhos, é formada uma matriz de correlação, onde seus autovalores e autovetores ajudam a classificar o tipo de superfície onde o ponto está localizado: um plano, uma aresta, uma borda ou um canto. A partir dos autovalores, funções-pesos para cada um dos tipos enumerados são geradas e as funções-pesos das arestas do grafo utilizam os pesos e a distância relativa entre os pontos ligados pela aresta em questão. As arestas são então ordenadas pelo seu peso e a cada inserção de uma aresta no subgrafo é verificado o fechamento de um laço. Essa verificação é necessária para evitar pequenos ciclos ao redor das descontinuidades. O ciclo só é permitido se for mais longo do que um comprimento definido pelo usuário.

Pauly *et al*. [Pauly et al. 2003] utilizaram um método similar ao de Gumhold [Gumhold et al. 2001]. Eles também utilizam análise de covariância para classificação dos vértices, mas estenderam a técnica ao utilizar uma classificação de múltipla-escala, permitindo um melhor controle sobre superfícies com ruído. A função-peso para cada ponto funciona da seguinte forma: calcula-se a variância dos k vizinhos mais próximos da amostra, incrementa-se o tamanho da vizinhança local e repete-se o processo n vezes. A função-peso do ponto guarda a quantidade de vezes que a variância da vizinhança ultrapassa um determinado limiar σ_{max}. Entretanto, aumentar o tamanho da vizinhança local enquanto calcula-se a estimativa de variância eventualmente viola o pré-requisito de que todos os pontos da vizinhança pertençam a uma mesma região conectada da superfície. A solução encontrada foi buscar um repentino crescimento da

variância, o que indica que houve fortes desvios de direção de normal (Figura 2.2).

Pauly também utiliza grafo de vizinhança, mas apenas a única estimativa de vizinhança utilizada na construção do grafo foi a utilização dos k vizinhos mais próximos (Grafo de Riemann). Entretanto a limiarização dos pontos é feita antes do cálculo do peso das arestas. Essa limiarização ternária descarta os pontos cujos pesos são menores do que σ_{min}; insere os pontos cujos pesos são maiores do que σ_{max} em um conjunto Q separado; e guarda os pontos cujos pesos estão situados entre os dois limiares, apenas para construir uma ligação entre os espaços dos nós do conjunto Q durante a construção de uma árvore geradora mínima que armazenará todas as arestas que representarão a descontinuidade.

Em 2006, Demarsin *et al*. [Demarsin et al. 2006] apresentaram uma ideia simples para a detecção de arestas fechadas em nuvens de pontos. A principal condição do método proposto é a de que a nuvem de pontos fornecida apresente apenas arestas que formam ciclos, ou seja, nenhuma descontinuidade pode se desfazer numa superfície suave. Essa condição permite a utilização de um método de crescimento de regiões para dividir a nuvem de pontos em diversos *segmentos*. Essa segmentação é realizada em duas etapas: 1) estimativa de normais e seleção de vizinhança; 2) crescimento de regiões.

Na primeira etapa, calcula-se a vizinhança e, em seguida, estima-se o vetor normal. Ao determinar os k vizinhos mais próximos de um ponto $\mathbf{P}$, com k suficientemente grande, constrói-se o plano de mínimos quadrados (*least squares plane*) e projetam-se os pontos sobre o plano. Após executar uma triangulação de Delaunay sobre essas projeções, somente os pontos que compartilham com a projeção do ponto $\mathbf{P}$ uma aresta nessa triangulação constituem a vizinhança de Delaunay de $\mathbf{P}$ [Floater e Reimers 2001]. Os vetores normais são estimados pela *Principle Component Analysis* (PCA) desse conjunto de vizinhos. Na segunda etapa, os autores utilizam um ângulo limiar que especifica o ângulo máximo aceitável entre duas normais adjacentes em um segmento. A estimativa da normal depende fortemente da vizinhança calculada e, devido à natureza muito local da vizinhança, a variação das normais ao longo de uma aresta é bastante alta, o que resulta em diversos pequenos segmentos sobre as arestas e grandes segmentos em áreas planas com baixa variação de normais.

Após a segmentação do modelo (Figura 2.3a) constrói-se um grafo de vizinhança. Mas, diferente de Gumhold *et al*. [Gumhold et al. 2001] e Pauly *et al*. [Pauly et al. 2003], o grafo é construído em nível de segmentos, o que leva a uma considerável redução de custo. Cada vértice do grafo representa um segmento e cada aresta conecta dois segmentos que contém pelo menos um ponto sobreposto a suas vizinhanças. Para cada vértice do grafo, mantém-se uma representação de pontos do segmento, ou seja, a média de todos os pontos do segmento e

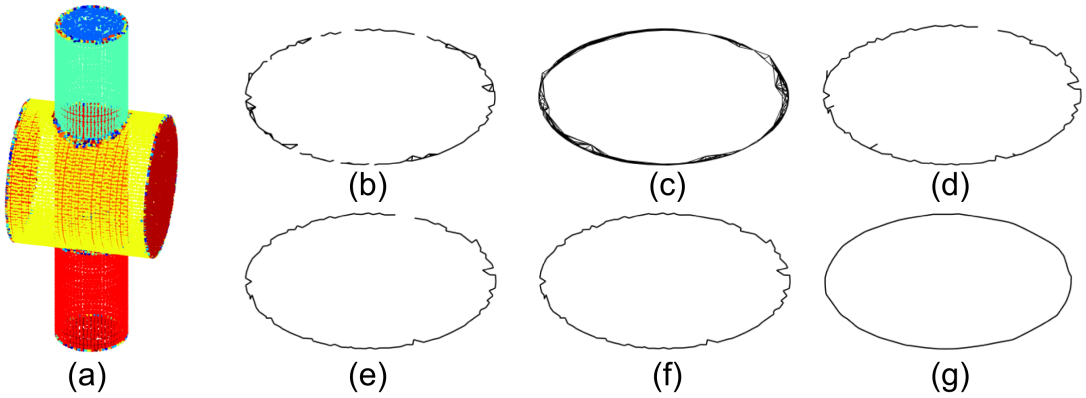

Figura 2.3: Diversas etapas da pipeline de Demarsin [Demarsin et al. 2006] para detecção de arestas fechadas.

informação sobre o tamanho do segmento. Para distinguir entre segmentos grandes e pequenos, pode-se utilizar um limiar definido pelo usuário. Entretanto, na prática, há muitos segmentos pequenos e poucos segmentos grandes. Os segmentos grandes têm tamanhos muito maiores do que os tamanhos dos segmentos menores, e, desta forma, utilizar a média de todos os tamanhos dos segmentos é uma boa heurística para separar segmentos pequenos e grandes.

Note na Figura 2.3a que, próximo às arestas, existem diversos segmentos devido à alta variação de normais estimadas nesta área. Considere um subgrafo G de pequenos segmentos vizinhos a dois segmentos grandes e as arestas que os conectam (Figura 2.3b). Esse subgrafo possui muitos pequenos espaços indesejáveis entre segmentos pequenos. Por esse motivo, arestas são adicionadas entre segmentos pequenos que compartilham os mesmos dois segmentos grandes como vizinhos (Figura 2.3c) formando o subgrafo $G_{extended}$. As arestas de $G_{extended}$ dão uma ideia da localização da linha da descontinuidade. Entretanto, muitos ciclos estão presentes, por isso, uma árvore geradora mínima G_{mst} é construída de $G_{extended}$ (Figura 2.3d), onde o peso das arestas é a distância entre os pontos representantes dos segmentos. A árvore G_{mst} possui uma reconstrução inicial da linha da descontinuidade, mas possui muitas ramificações pequenas. Essas ramificações são removidas através de um algoritmo de poda (Figura 2.3e) resultando no grafo $G_{mst\_prunned}$. Uma árvore não possui ciclos, porém objetiva-se encontrar linhas de descontinuidade fechadas. Então, para cada ponto terminal da árvore, busca-se o outro ponto terminal mais próximo para conectá-los e, assim, fechar o ciclo (Figura 2.3f) concluindo no grafo G_{closed}. Para uma melhor exibição da linha da descontinuidade, a polilinha formada pelas arestas do grafo G_{closed} pode adaptada através de algum método de suavização (Figura 2.3g).

A metodologia descrita em [Demarsin et al. 2006] apresenta várias similaridades com o método descrito nesta dissertação. A ideia de agrupar amostras, utilizando suas normais, segmentando o modelo foi utilizada aqui para melhor reconstrução de uma superfície em espaço

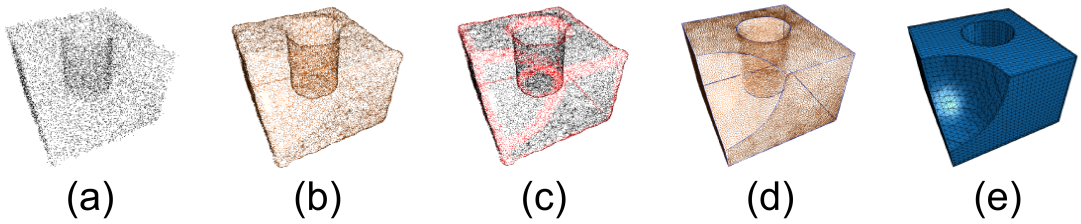

Figura 2.4: Reconstrução de superfície Bayesiana em [Jenke et al. 2006]. (a)Nuvem de pontos enruidada. (b)Suavização inicial. (c)Detecção de pontos com alta probabilidade de pertencer a uma descontinuidade. (d)Suavização dos pontos detectados. (e)Reconstrução das superfície através de triangulação.

de imagem no final da pipeline de *splatting*. Entretanto, o cálculo impreciso de vetores normais nas proximidades das arestas não pode ser utilizado aqui, por este motivo são utilizados métodos mais robustos para o cálculo de normais. Outra vantagem do método proposto neste trabalho é que as arestas não precisam ser fechadas para serem detectadas, apesar disso levar a falhas na segmentação da superfície. Outra vantagem do método proposto deve-se ao cálculo de vizinhança que apresenta a mesma característica local do método de Demarsin, mas, ao explorar a estrutura de splats, não precisa calcular os k vizinhos mais próximos, um sério gargalo principalmente para valores grandes de k, como reconhece Demarsin. Além disso, não há necessidade de fazer uma triangulação de Delaunay para estimar a vizinhança em cada ponto.

Jenke *et al*. [Jenke et al. 2006] utilizaram uma estratégia parecida com aquela descrita em [Demarsin et al. 2006] para reconstrução de superfícies através de estatísticas bayesianas. Como o método de reconstrução é estocástico, a medida utilizada para detectar pontos próximos a uma descontinuidade era uma função de probabilidade calculada com base na curvatura local. O método inicialmente suaviza os dados, ou seja, altera a geometria antes do início da detecção de arestas. Em seguida, estima as probabilidades de um ponto pertencer a uma descontinuidade e inicia um algoritmo de crescimento de região sobre o grafo de vizinhança, como é feito em [Demarsin et al. 2006]. Cada região recebe um identificador diferente e os pontos de bordas recebem um identificador diferenciado. Isso é feito para estimar as arestas com precisão, já que elas continuam com muito ruído. Dessa forma, suavizam-se novamente os dados dos pontos sobre arestas. Só então, realiza-se a reconstrução da superfície independentemente para cada pedaço da superfície previamente separado. A Figura 2.4 ilustra todo esse procedimento. Apesar da estratégia diferente de reconstruir a superfície, o método ainda necessita que as arestas sejam fechadas. Outro revés deve-se à alteração da geometria para o cálculo das probabilidades. Esse tipo de operação pode causar perda de características da superfície além de gastar tempo em pré-processamento.

Daniels *et al*. [Daniels et al. 2007] adotaram uma adaptação da reconstrução de superfícies

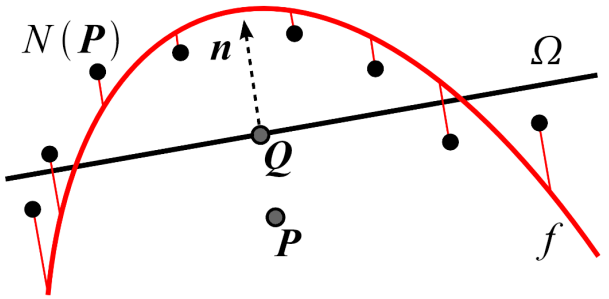

Figura 2.5: Quanto mais próximo de uma aresta, mais distante a superfície MLS fica dos pontos amostrais. Esta característica foi explorada em [Fleishman et al. 2005] e [Daniels et al. 2007] para ponderar as amostras com erro residual.

MLS (*Moving Least Squares*) chamada RMLS (*Robust Moving Least Squares*) de Fleishman *et al*. [Fleishman et al. 2005]. Nessa técnica, utiliza-se uma medição de erro residual para detecção de *outliers*. Se um ponto apresenta um erro residual alto, pode ser que ele seja um *outlier* ou que esteja próximo a uma descontinuidade, pois devido à característica da reconstrução MLS de suavizar as descontinuidades, os pontos próximos às arestas ficam cada vez mais distantes da função polinomial adaptada.

Considere uma função polinomial f sobre um plano de referência Ω que é definido por um ponto $\mathbf{Q}$ e uma normal $\mathbf{n}$. O erro residual ε na vizinhança de um determinado ponto $\mathbf{P}$ é

$$\varepsilon(\mathbf{P}) = max_{\mathbf{X}\in N(\mathbf{P})} \|\mathbf{X} - \mathbf{X}_f\|$$

onde $\mathbf{X}_f$ corresponde à projeção de $\mathbf{X}$ sobre f na direção da normal, $\mathbf{n}$, ao plano. A Figura 2.5 mostra o cálculo do erro residual utilizado em [Fleishman et al. 2005] e [Daniels et al. 2007].

Calculados os erros residuais de cada ponto, é definido que um ponto $\mathbf{P}$ do conjunto é um ponto de descontinuidade em potencial se $\varepsilon(\mathbf{P})$ for maior do que um limiar definido pelo usuário. Na tentativa de automatizar o processo, Daniels sugere aplicar ao limiar o valor do erro residual médio de todos os pontos amostrais.

Essa técnica apresenta as vantagens das técnicas presentes em [Gumhold et al. 2001] e [Pauly et al. 2003] de detectar arestas fechadas e abertas, entretanto as técnicas anteriores extraem as descontinuidades através da conexão de pontos existentes, ou seja, dependem fortemente da qualidade da amostragem do modelo na proximidade de arestas. O método de Daniels projeta os pontos sobre as arestas acarretando numa melhor aproximação das descontinuidades originais da superfície. A Figura 2.6 mostra o método de extração de arestas e cantos presente em [Daniels et al. 2007]. Os autores ainda estenderam sua técnica ao aplicar curvas b-spline sobre as curvas de arestas detectadas em [Daniels et al. 2008]. Apesar da boa qualidade do método, apresenta uma alta complexidade computacional devido à sua estratégia de otimização

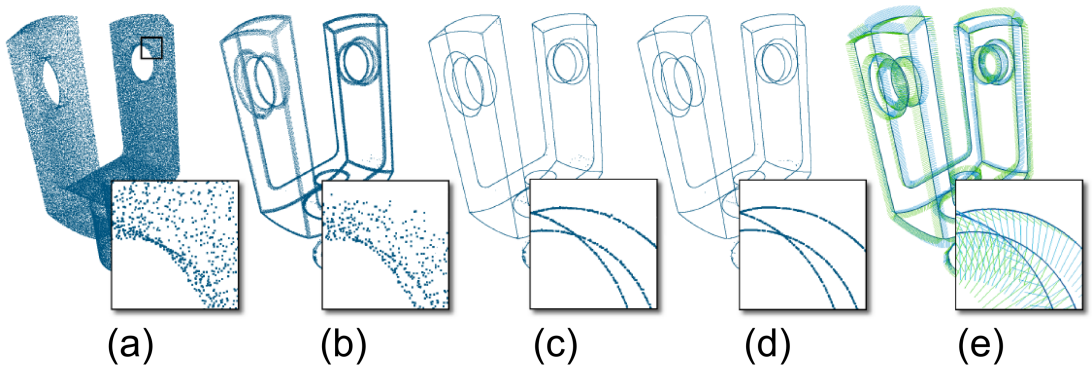

Figura 2.6: Método de detecção e reconstrução de descontinuidades proposto por Daniels *et al*. [Daniels et al. 2007]. (a) Nuvem de pontos original. (b) Identificação de pontos candidatos por estarem próximos a possíveis descontinuidades. (c) Projeção dos pontos candidatos sobre a interseção das superfícies RMLS. (d) Suavização dos pontos projetados. (e) Reconstrução das linhas que identificam as descontinuidades.

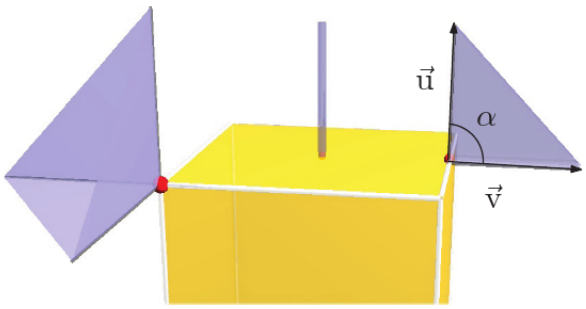

Figura 2.7: Detecção de arestas e cantos por [Mérigot et al. 2009]

iterativa que faz diversas projeções MLS por ponto, para calcular seu erro residual. Assim, para os objetivos de apenas encontrar pontos da entrada que tenham grande probabilidade de pertencerem a uma aresta, esse método não é o mais adequado.

Merigot *et al*. [Mérigot et al. 2009] estimam as curvaturas principais e direções de normais da superfície representada pela nuvem de pontos através da medida de covariância de Voronoi e provê valiosas garantias teóricas. Uma matriz de covariância convoluída é computada da união das células de Voronoi e retorna as curvaturas principais e as direções principais. A ideia básica pode ser vista na Figura 2.7. A célula de Voronoi infinitesimal de um ponto $\mathbf{X}$ sobre um cubo é pontuda nas áreas planas, triangular sobre as arestas e em forma de cone sobre os cantos. Apesar da estratégia diferenciada, esse método apresenta similaridades com os algoritmos baseados em PCA, pois as descontinuidades são estimadas para uma grande quantidade de pontos próximos à linha da aresta, além de ser baseada na curvatura.

Weber *et al*. [Weber et al. 2010] utilizaram clusterização em mapas Gaussianos de vizinhanças locais para descartar todos os pontos não pertencentes a descontinuidades. Como nos

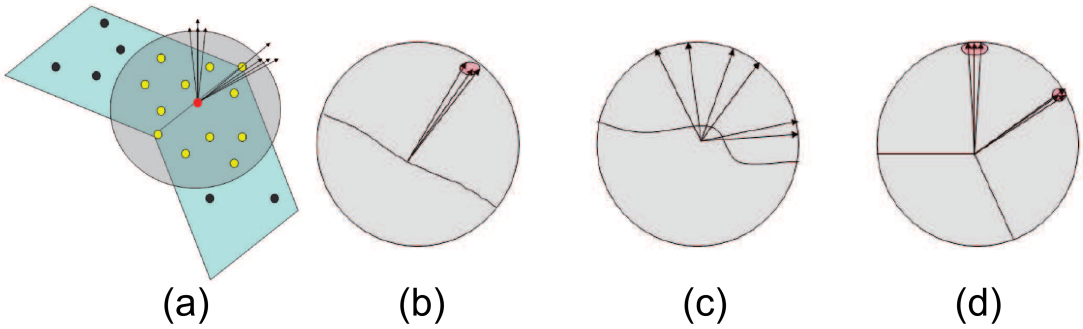

Figura 2.8: (a) Geração do mapa gaussiano da vizinhança de determinado ponto. (b) Mapa gaussiano de um ponto sobre uma região plana. (c) Mapa gaussiano sobre uma região arredondada. (d) Mapa gaussiano sobre uma descontinuidade. [Weber et al. 2010].

demais métodos, um parâmetro baseado na curvatura local é utilizado para descartar pontos muito distantes das arestas. Num segundo estágio, os candidatos a descontinuidades (*feature candidates*) restantes passam por um processo de seleção mais preciso. A principal vantagem do método deve-se à computação automática de um parâmetro de sensibilidade adaptativo, o que torna o método robusto na presença de ângulos obtusos e agudos.

Similar a várias outras técnicas, os autores utilizam os k vizinhos mais próximos. Para um determinado ponto $\mathbf{P}$ e seus k vizinhos, o método calcula todas as triangulações possíveis entre o ponto $\mathbf{P}$ e seus vizinhos, formando $k \cdot (k-1)$ triângulos, cada um com seu vetor normal. O mapa Gaussiano da vizinhança de $\mathbf{P}$ pode agora ser definido pelo conjunto de pontos da forma:

$$\mathbf{X}_{ij} = \mathbf{P} + \frac{\mathbf{n}_{ij}}{\|\mathbf{n}_{ij}\|}$$

onde $\mathbf{n}_{ij}$ é o vetor normal formado pelo triângulo $\Delta\mathbf{P}\mathbf{P}_i\mathbf{P}_j$ (Figura 2.8a). Um primeiro teste de nivelamento descarta os pontos presentes em áreas planas. Esse teste consiste em medir o desvio padrão dos vetores normais. Se esse valor for pequeno, assume-se que o ponto não pertence a uma descontinuidade (Figura 2.8b). Um valor sugerido para o desvio padrão limiar é 15%. Através de um algoritmo de clusterização, agrupa-se as diversas projeções $\mathbf{X}_{ij}$. Quando todos os clusters têm poucos pontos, isso indica que a região correspondente é curva (Figura 2.8c). Se entre dois a quatro clusters possuem a maioria dos pontos, isso indica que a região é uma aresta ou canto (Figura 2.8d). Se os pontos estão distribuídos por uma quantidade de clusters maior do que quatro, o método considera que não há descontinuidade.

É difícil encontrar a quantidade ideal do número de vizinhos k e da distância angular mínima entre dois clusters σ. Por esse motivo, Weber *et al.* desenvolveram uma forma adaptativa de escolher esses valores para cada ponto da geometria. Na primeira iteração, é escolhido um valor baixo para σ e alto para k. Isso garante que uma grande quantidade de candidatos sejam

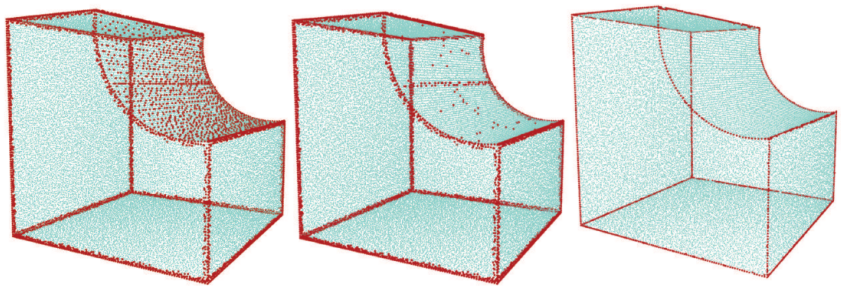

Figura 2.9: As figuras da esquerda e do centro aplicaram diferentes parâmetros globais (Esquerda: $\sigma = 0.1$; $k = 20$ e Centro: $\sigma = 0.5$; $k = 16$). A figura da direita é o resultado obtido com o método iterativo, que adapta automaticamente σ e k para cada candidato.

escolhidos para que não se perca qualquer ponto presente numa descontinuidade. As etapas seguintes removem os falsos positivos escolhidos na primeira etapa. No caso de um candidato estar presente em uma aresta ou em um canto, ele dividirá a vizinhança em duas partes, ou seja, a porcentagem de outros candidatos dentro dessa vizinhança é baixa. Uma porcentagem muito alta de outros candidatos significa que a sensibilidade foi muito alta, ou seja, o valor de σ foi muito baixo para essa vizinhança. Caso não haja nenhum outro candidato na vizinhança este ponto é um *outlier*. Dessa forma, aumenta-se cuidadosamente o valor de σ e diminui-se o tamanho da vizinhança. Os autores sugerem aumentar σ em 10% e diminuir k em 1 a cada iteração. Quanto ao critério de parada, aplica-se alguma heurística plausível, como o número mínimo de candidatos na vizinhança ou o número máximo de iterações ou ainda um limite inferior para k e um limite superior para σ. A Figura 2.9 mostra a dificuldade da escolha de parâmetros globais e a vantagem da parametrização adaptativa.

Todos os métodos de detecção de descontinuidades apresentados até aqui são realizados em modelos baseados em pontos e, dessa forma, não podem extrair vantagens dos modelos baseados em splats. Assim como uma malha poligonal, os elementos de um modelo baseado em splats apresentam normais e dimensões apesar da ausência de conectividade. Várias pesquisas na detecção de descontinuidades em malhas poligonais têm sido apresentadas, mas uma abordagem particularmente simples e eficiente é proposta por Kobbelt *et al.* em [Kobbelt et al. 2001]. Como o próprio autor afirma, se os pontos forem trocados por pequenas faces, os cálculos são bem similares aos utilizados para malhas poligonais. Seu trabalho tem por objetivo reconstruir superfícies, sejam elas implícitas, nuvens de pontos ou mesmo campos de distância escalares através da técnica de *marching cubes* adaptado. Essa adaptação detecta as descontinuidades e insere novos triângulos na malha gerada a fim de melhor representar arestas e cantos.

O método de detecção é bastante simples. Se o produto escalar entre duas normais $\mathbf{n}_i$

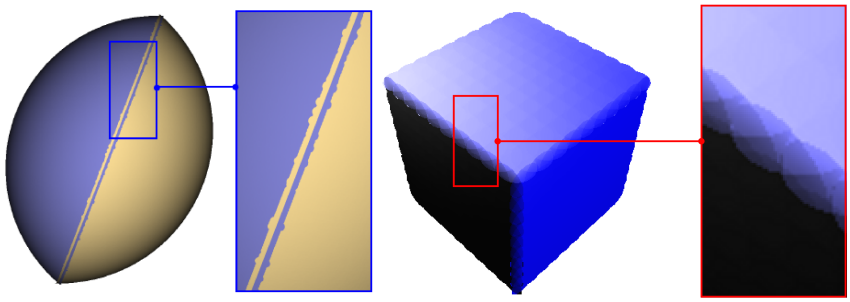

Figura 2.10: Artefatos de splats se cruzando aparecem nas arestas e cantos (esquerda), mesmo combinando estes splats, o sombreamento ou textura tornam evidente a tentativa de suavizar as descontinuidades (direita).

e $\mathbf{n}_j$ de amostras próximas for menor que um limiar θ_{sharp}, então espera-se que a superfície apresente alguma descontinuidade nessa área. Sejam $\mathbf{n}_0$ e $\mathbf{n}_1$ as duas normais que formam o maior ângulo e $\mathbf{n}^* = n_0 \times \mathbf{n}_1$ o vetor normal ao plano formado por $\mathbf{n}_0$ e $\mathbf{n}_1$. Para determinar se a descontinuidade é uma aresta ou um canto, estima-se o desvio máximo das normais $\mathbf{n}_i$ do plano formado. Ou seja, se o produto escalar de alguma normal $\mathbf{n}_i$ com $\mathbf{n}^*$ for maior do que um limiar φ_{corner}, então, a descontinuidade é um canto, caso contrário, é uma aresta. Os limiares sugeridos pelos autores foram $\theta_{sharp} = 0.9$ e $\varphi_{corner} = 0.7$.

Neste trabalho usa-se essa definição para realizar o teste que detecta se dois splats formam uma aresta entre eles. Ou seja, não se quer identificar um splat exatamente sobre uma aresta, mas sim identificar quando o recorte de um splat por outro é necessário. Identificar apenas se há uma descontinuidade é suficiente, pois arestas curvas são tratadas como cantos e os splats presentes nelas são recortados de forma a adaptar sua forma. Apesar dos benefícios dessa simples abordagem às necessidades, a utilização de um limiar global é uma limitação séria do método. Pretende-se desenvolver um método adaptativo, como proposto em [Weber et al. 2010].

2.3 Representação das Arestas

Para muitas aplicações, a habilidade de renderizar superfícies com descontinuidades, tais como cantos e arestas, é uma exigência. Na técnica de *Surface Splatting*, as amostras da geometria são discos em espaço de objeto, que projetados e combinados no espaço de imagem, reconstroem a superfície de forma contínua. Entretanto, próximo a arestas e cantos, deve-se ter uma atenção especial, pois a amostragem pode exibir buracos ou a sobreposição dos splats apresenta artefatos (Figura 2.10).

Um dos primeiros trabalhos a tratar esse tipo de característica foi o artigo de Adams e

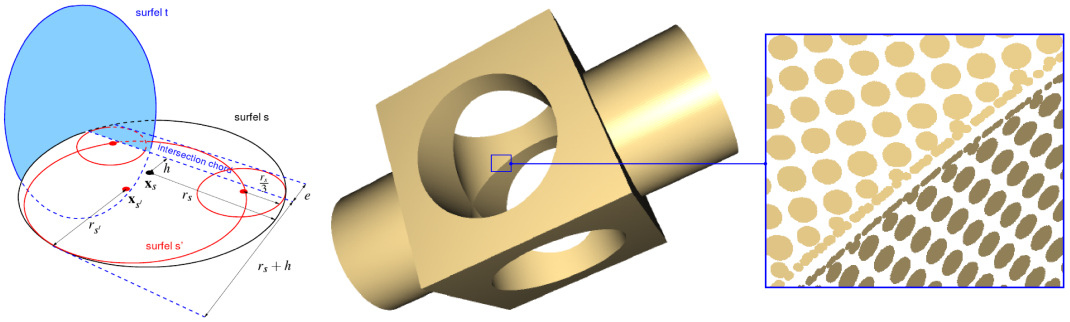

Figura 2.11: Interseção de um splat com o plano do splat mais próximo pertencente a outra superfície. O splat é trocado por três splats menores através de um operador de reamostragem. A direita vemos o resultado da reamostragem. Apesar de adicionar novos splats, ao ampliar a imagem, a estrutura dos splats é revelada. [Adams e Dutré 2003]

Dutré [Adams e Dutré 2003]. Nesse trabalho, primitivas necessariamente formadas por superfícies suaves são submetidas a operações booleanas com outras superfícies, todas formadas por splats. Como normalmente a interseção dessas superfícies formam arestas e cantos na superfície resultante da operação booleana, uma abordagem era necessária para uma boa renderização do modelo resultante. As curvas de interseção são detectadas através de uma octree, onde as folhas ainda apresentam amostras de ambas as superfícies da operação booleana. Todos os splats que cruzam essa curva são analisados e, possivelmente, substituídos por outros splats menores.

Seja S um splat e, T seu vizinho mais próximo, porém pertencente a uma superfície diferente da qual S pertence. O plano definido por T deve dividir S em duas partes. Dependendo das operações booleanas, pode-se manter a parte menor ou maior do splat S. Na hipótese de se manter a parte maior, a linha da aresta é aproximada através da troca do splat S por outro splat menor S'. O novo splat cruza a interseção até um determinado valor especificado pelo usuário, denominado de *overshoot error*. Para preencher completamente a linha de interseção, adicionam-se novos splats menores à esquerda e à direita com raios iguais a um terço do raio do splat original (Figura 2.11 a esquerda). Essa abordagem diminui a área borrada abaixo de um pixel, além de ser muito rápida, porém nunca esconderá completamente a natureza amostrada por pontos da aresta. Ampliar a aresta resultará em borramento (Figura 2.11 direita).

Ao notar o problema da técnica em [Adams e Dutré 2003], percebe-se que seria necessário uma quantidade infinita de splats para representar arestas de forma contínua. Entretanto, Pauly *et al*. [Pauly et al. 2003] introduziram uma classe especial de splats que representa arestas explicitamente. Inicialmente, encontram-se todos os pontos das duas superfícies da operação booleana que estão próximos à curva de interseção, por meio da avaliação da função de distância induzida pelo operador de projeção MLS. Então formam-se todos os pares mais próximos desses pontos e calcula-se um ponto na curva de interseção (Figura 2.12a-d). O método de Newton

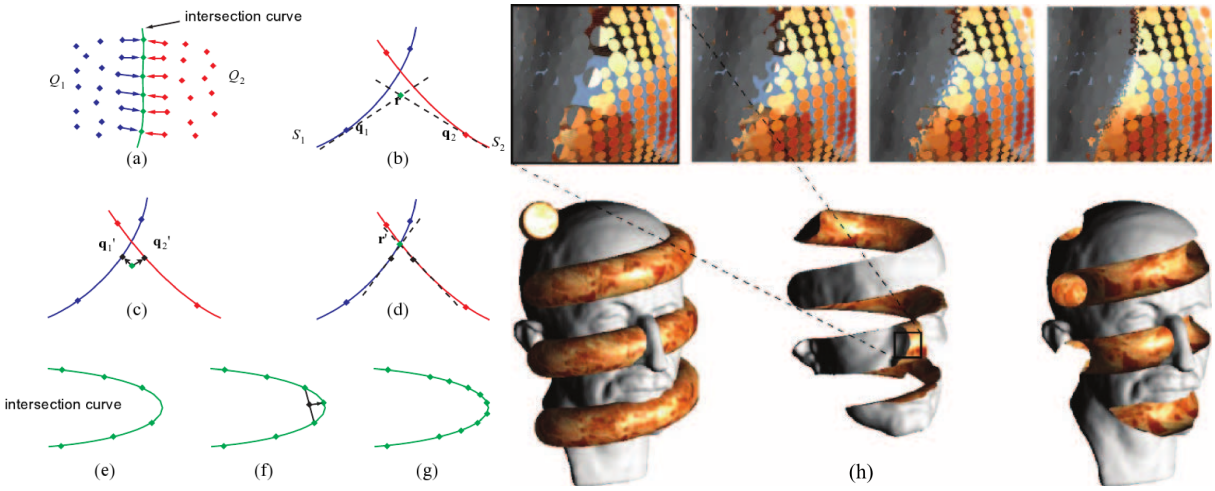

Figura 2.12: (a) Pares de pontos mais próximos. (b-d) Projeções para posicionar ponto sobre a curva. (e-g) Refinamento adaptativo da curva. (h) Modelos gerados com splats recortados dividindo mesmo centro. [Pauly et al. 2003]

também provê um fácil mecanismo para refinar adaptativamente a curva de interseção. A regra de subdivisão para criar um novo ponto inicial pode ser simples, por exemplo, calcular a média de dois pontos adjacentes na curva. Em seguida, aplicam-se as iterações para amostrá-lo sobre a curva de interseção (Figura 2.12e-g).

Após a projeção dos pontos sobre as arestas, é preciso renderizá-los por meio dos splats. Para renderizar corretamente arestas e cantos, Pauly *et al.* representam os pontos sobre a curva de interseção por dois splats que dividem o mesmo centro, mas com normais das superfícies de cada lado da aresta. Durante a rasterização, cada um desses splats é recortado contra o plano definido pelo outro para obter uma aproximação linear da curva de interseção em espaço de tela (Figura 2.12h). Apesar dessa abordagem apresentar melhor apresentação de descontinuidades, mesmo quando ocorre ampliação da imagem, assim como na técnica anterior, novos pontos são inseridos no modelo, necessitando de vários cálculos para a inserção de pontos sobre a aresta. Outra desvantagem ocorre em cantos, pois, nesses casos, são necessários cada vez mais normais associadas a um único splat.

Outra desvantagem dos métodos anteriores deve-se à prévia informação da localização das arestas. Considera-se que as descontinuidades só estão presentes nas curvas de interseção da operação booleana. Utilizando-se de informações prévias de métodos de detecção de descontinuidades ou informações de modelagem, Zwicker *et al.* [Zwicker et al. 2004] adaptaram os splats próximos às arestas por meio da inserção de linhas de corte (*clip lines*). As linhas de corte dividem o splat em duas partes. Em uma das partes, o núcleo de reconstrução é avaliado normalmente, enquanto que a outra parte é descartada, ou recortada. Essas linhas de corte são calculadas na interseção dos planos tangentes dos splats de cada lado da aresta. Como os pares

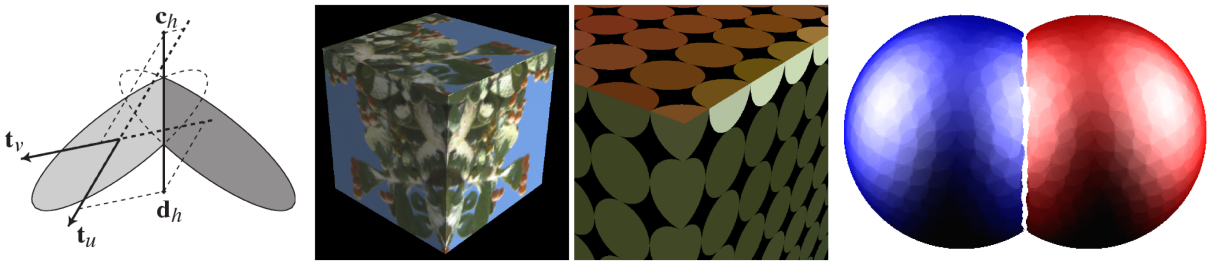

Figura 2.13: (a) Linha de corte localizada na interseção de dois planos tangentes. (b) Modelo renderizado com método proposto. (c) Detalhe na aresta. (d) Quando a amostragem é irregular de ambos os lados da aresta e adicionam-se várias linhas de recorte, buracos são inseridos na superfície. [Zwicker et al. 2004]

de splats são recortados pela mesma linha de corte, buracos não aparecem devido ao recorte. A linha de corte é representada por dois pontos sobre a interseção dos planos (Figura 2.13a). Esses pontos são projetados no espaço de imagem, permitindo que cada pixel da projeção do splat seja analisado, para saber em que lado da linha esse pixel se encontra e, dependendo do lado, o pixel é descartado. Apesar da abordagem realizar uma boa representação da aresta com poucos splats e sem projetá-los exatamente sobre a linha da descontinuidade (figuras 2.13b e c), quando várias linhas de corte são inseridas e cada uma é independente de outra, podem surgir ambiguidades e haver pixels recortados desnecessariamente, acarretando buracos (Figura 2.13d). Essas ocasiões ocorrem com mais frequência em cantos e em arestas curvas, quando as amostras não estão alinhadas, formando pares.

Para classificar um ponto de uma superfície como interno ou externo a outra superfície, necessária na técnica de CSG, [Adams e Dutré 2003] e [Pauly et al. 2003] utilizam apenas o ponto mais próximo da outra superfície para classificá-lo. Para classificar um ponto $\mathbf{X}$ como interno ou externo a um objeto representado pela superfície A, somente o splat de A mais próximo é utilizado na classificação. Seja S esse splat, $\mathbf{X}$ é considerado interno a A se S não "vê" $\mathbf{X}$:

$$\begin{aligned} \mathbf{X} \text{ é interno a } A &\Leftrightarrow \neg[S \text{ vê } \mathbf{X}] \\ &\Leftrightarrow (\mathbf{X} - \mathbf{C}_s) \cdot \mathbf{n}_s < 0 \end{aligned}$$

onde, $\mathbf{C}_s$ denota o centro e $\mathbf{n}_s$ a normal do splat S.

Como ilustrado na Figura 2.14a, isso leva a erros de classificação causando arestas serrilhadas. Pauly *et al.* [Pauly et al. 2003] utilizam o operador de projeção MLS para obter uma classificação exata, mas é computacionalmente muito caro calcular esse operador várias vezes para cada fragmento. Wicke *et al.* [Wicke et al. 2004] trataram esse prolema ao utilizar os dois splats mais próximos S_1 e S_2 para a classificação. O teste interno/externo é feito para cada um

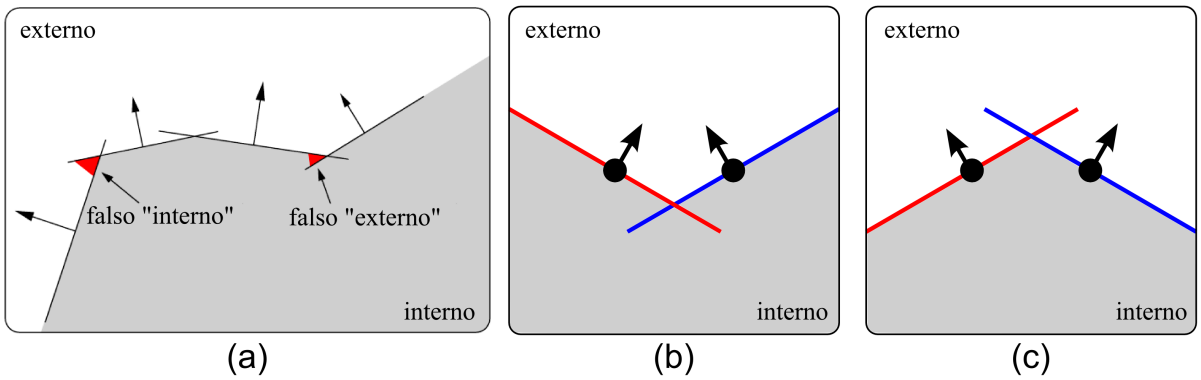

Figura 2.14: (a) Utilizando somente o splat mais próximo para classificação dentro/fora leva a erros de classificação. (b) As arestas criadas pela interseção de duas superfícies amostradas diferentemente, renderizadas usando somente o splat mais próximo na classificação. (c) a mesma aresta renderizada com teste dentro/fora proposto em [Wicke et al. 2004].

dos splats e seus resultados são combinados dependendo da configuração desses splats. Diz-se que S_1 e S_2, com centros $\mathbf{C}_1$ e $\mathbf{C}_2$, respectivamente, são *côncavos*, se S_1 "vê" $\mathbf{C}_2$ e S_2 "vê" $\mathbf{C}_1$ (Figura 2.14b), caso contrário, são chamados *convexos* (Figura 2.14c). Assim a classificação interno/externo ocorre dependendo da classificação côncavo/convexo. Se S_1 e S_2 são côncavos, então:

$$\mathbf{X} \text{ é interno a } A \Leftrightarrow \neg[S_1 \text{ vê } \mathbf{X}] \vee \neg[S_2 \text{ vê } \mathbf{X}] \text{ (Figura 2.14b)}$$

caso contrário, se S_1 e S_2 são convexos:

$$\mathbf{X} \text{ é interno a } A \Leftrightarrow \neg[S_1 \text{ vê } \mathbf{X}] \wedge \neg[S_2 \text{ vê } \mathbf{X}] \text{ (Figura 2.14c)}$$

Ou seja, no caso de splats côncavos, a união dos pontos não vistos por eles é interna à superfície. Caso os splats sejam convexos, a interseção dos pontos não vistos por eles é interna à superfície. Essa classificação é importante não somente para identificar os splats que serão removidos pela operação booleana, mas principalmente para adaptar os splats que devem ser recortados.

Zwicker *et al.* [Zwicker et al. 2004] removem todos os fragmentos que estão após a linha de corte, mas cada linha é independente de outra, ou seja, no caso da inserção de mais de uma linha de corte no mesmo splat, pode ocorrer a aparição de buracos em cantos complexos e arestas curvas (Figura 2.13d). Wicke cria uma lista para cada splat que estiver próximo da linha de interseção. Seja S um desses splats. Na lista de S, encontram-se todos os splats do outro objeto que estão suficientemente próximos de S para poder recortá-lo. Os elementos dessa lista são denominados *clip partners* de S. Um splat é considerado suficientemente próximo de outro para efeito de recorte, se a distância entre o centro desses splats for menor do que a soma de seus raios. Após a determinação dos *clip partners* de cada splat, o recorte é realizado durante a rasterização da elipse. Coordenadas de mundo do objeto são computadas para cada fragmento

e usadas para determinar se o fragmento está, ou não, na superfície resultante, a partir das operações booleanas armazenadas na árvore CSG. Muitas das ideias presentes em [Wicke et al. 2004] foram adaptadas para modelos baseados em splats em geral e sem a utilização de informações de modelagem neste trabalho.

2.4 Segmentação de modelos

A segmentação é o processo de particionar uma nuvem de pontos em regiões significativas ou extrair importantes características dos pontos dados como entrada. Essa técnica serve de motivação ou de entrada para outros métodos, tais como: reconstrução de superfície, aplicações de compressão, engenharia reversa, etc. A maioria dos métodos de segmentação de conjuntos de pontos podem ser classificados como: métodos de detecção de arestas, métodos de crescimento de regiões e métodos híbridos.

Os métodos de detecção de arestas tentam detectar descontinuidades nas superfícies que formam fronteiras fechadas de componentes nos pontos entrados. Os métodos de detecção de arestas tentam detectar descontinuidades nas superfícies, que dividem a superfície em vários componentes. Os métodos de crescimento de regiões, por outro lado, realizam a segmentação ao detectar superfícies contínuas que possuem homogeneidade ou propriedades geométricas similares. As abordagens híbridas tentam resolver as limitações envolvidas em ambas as abordagens anteriores. Devido à grande quantidade de métodos, nesta seção, são apresentados apenas aqueles mais similares ao método proposto neste trabalho.

Woo *et al*. [Woo et al. 2002] utilizaram a seguinte abordagem para segmentação de nuvens de pontos. Uma octree é construída sobre o modelo. O desvio padrão dos vetores normais aos pontos internos a uma célula indica os níveis de mudança na forma do pedaço da superfície interno à célula. Quando o desvio padrão é maior do que um limiar definido pelo usuário, a célula é subdividida e células vazias são eliminadas. Os tamanhos das células diferem de acordo com a forma do objeto. Em áreas altamente curvas, muitas pequenas células são geradas enquanto poucas células são necessárias em áreas relativamente planas.

Para realizar a segmentação, os pontos em áreas altamente curvas são extraídos através da remoção das células de menor tamanho. A remoção dessas células separa automaticamente a nuvem de pontos em várias regiões, deixando espaços espaços entre elas. O tamanho das células 3D a serem extraídas é definido também pelo usuário. Por razões de eficiência computacional, a estrutura de octree é utilizada novamente para a tarefa de separação. Uma célula da octree é unida a uma célula vizinha se o ângulo entre a normal média de cada célula for menor do

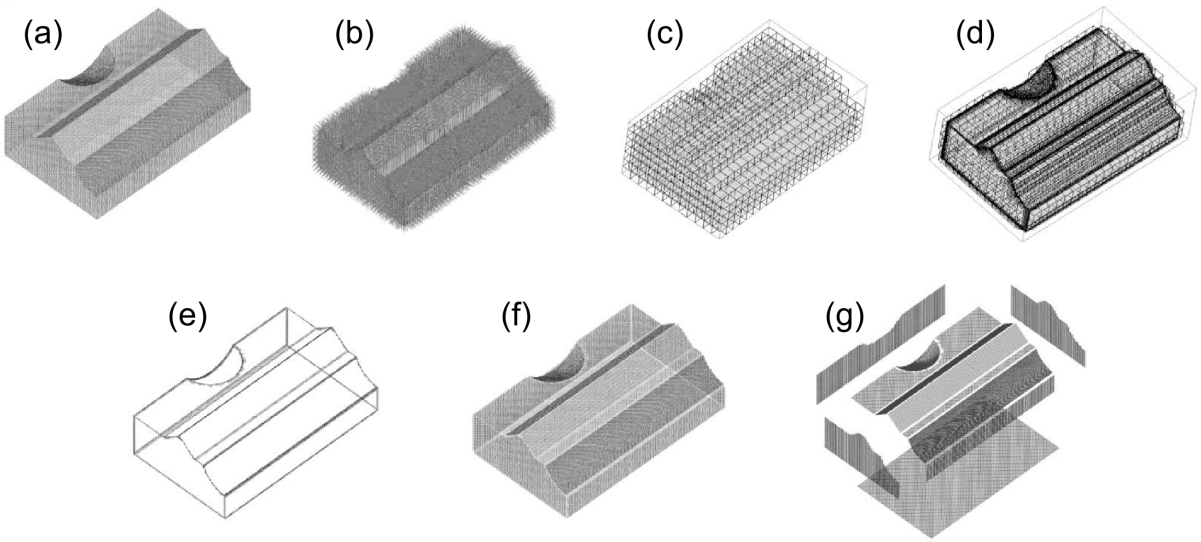

Figura 2.15: Pipeline de segmentação de nuvem de pontos proposto por Woo *et al.* em [Woo et al. 2002]. (a) Nuvem de pontos original. (b) Estimativa de normais dos pontos. (c) Octree inicial. (d) Octree final. (e) Pontos de aresta extraídos. (f) Pontos de aresta removidos. (g) Modelo segmentado.

que um valor pré-determinado. A Figura 2.15 explica o procedimento proposto em [Woo et al. 2002]. Apesar da octree tornar o método eficiente, e apesar da ausência da geração de grafo de vizinhança, o método necessita de várias restrições. A estimativa de normais depende fortemente do método utilizado na aquisição da nuvem de pontos, como o tipo de scanner utilizado. Além do usuário ter que definir limiares pouco intuitivos, a distribuição dos pontos deve ser bastante uniforme para um bom funcionamento do método.

Vanco e Brunnet [Vanco e Brunnett 2002] propuseram um método de segmentação para finalidades de engenharia reversa. A fase de segmentação agrupa os pontos em subconjuntos para facilitar os demais passos, tais como o passo de reconstrução de superfície. O primeiro passo do método proposto é baseado nos vetores normais aos pontos, pois as arestas do objeto possuem alta variação de vetores normais, e assim, podem ser detectadas. No segundo passo, as curvaturas principais são utilizadas para subdividir o conjunto de dados. Áreas contínuas, mas com curvatura descontínuas são detectadas e segmentos que estiverem na mesma superfície algébrica simples (como planos, cones, cilindros ou esferas) são detectados e unidos.

A primeira segmentação utiliza vetores normais para detectar arestas, regiões de alta curvatura, assim como áreas planas no objeto a ser reconstruído. Essa segmentação utiliza dois ângulos α e β para subdividir a superfície em clusters de forma que duas regras sejam respeitadas: 1) O ângulo entre dois vetores normais, $\mathbf{n}_r$ e $\mathbf{n}_i$, a dois pontos, $\mathbf{P}_r$ e $\mathbf{P}_i$, onde $\mathbf{P}_i$ está no conjunto dos k vizinhos mais próximos de $\mathbf{P}_r$, tem que ser menor do que o ângulo α. 2) O ângulo entre o vetor $\mathbf{n}_i$, normal à superfície no ponto $\mathbf{P}_i$, que é um candidato a ser adicionado

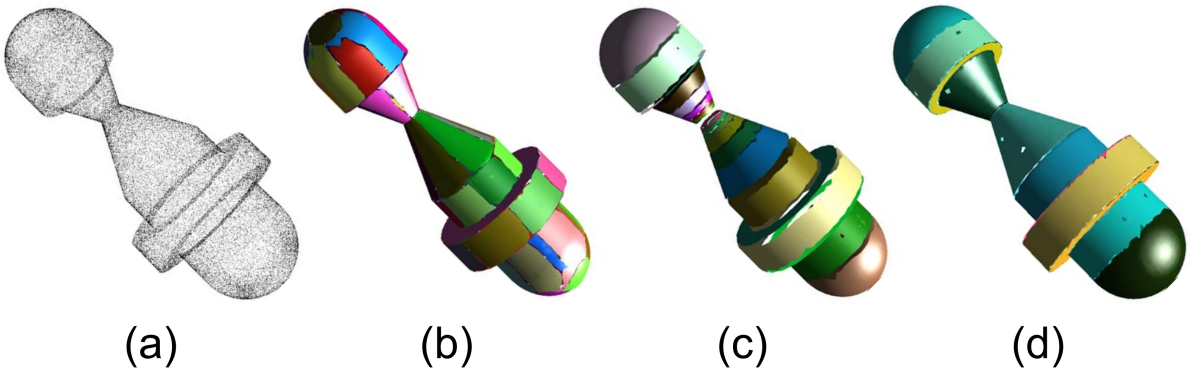

Figura 2.16: Pipeline de segmentação de nuvem de pontos proposto por Vanco *et al*. em [Vanco e Brunnett 2002]. (a) Nuvem de pontos original. (b) Segmentação baseada nos vetores normais. (c) Segmentação baseado na curvatura. (d) Classificação e extensão dos segmentos reconhecidos.

a um segmento existente, e o vetor normal de referência $\mathbf{n}_{ref}$ do segmento tem que ser menor do que o ângulo β. Aqui, o vetor normal de referência de um segmento é um vetor unitário na direção do vetor obtido por meio da soma dos vetores normais de todos os pontos do segmento. O ângulo α controla a divisão da aresta por especificar o ângulo de aceitação máximo entre vetores normais adjacentes. Se o objeto possuir arestas com ângulos variados, α deve ser o menor de todos eles. O ângulo β controla a planaridade dos segmentos. Para superfícies suaves, a segmentação é feita tomando como referência apenas os valores desse ângulo. Na vizinhança de arestas, a estimativa inicial de normais pode fornecer normais incorretas que realizam uma transição "suave" através da aresta. Nesse caso β indicará uma divisão do segmento.

Apesar dessa segmentação respeitar as descontinuidades do modelo, ela traz algumas propriedades indesejáveis. Apesar de segmentos maiores proverem uma segmentação razoável dos dados, uma grande quantidade de pequenos segmentos são produzidos. Para reduzir o número de pequenos segmentos, os autores implementaram três procedimentos para melhorar a segmentação. O primeiro procedimento mescla um segmento pequeno com algum segmento vizinho maior, caso o ângulo entre seus vetores normais de referência viole o critério do ângulo β, segundo um critério pré-definido pelo usuário. Os segmentos pequenos restantes são processados num segundo procedimento que pretende reduzir o tamanho de um segmento pequeno através da extração dos pontos de suas bordas. Por fim, um terceiro procedimento simplifica a segmentação ao mesclar os segmentos pequenos restantes. Através de testes, os autores sugerem o uso de $\alpha = 8^\circ$ e $\beta = 20^\circ$ como valores *default*. Sugerem também utilizar a segmentação para reestimar os vetores normais a fim de melhorar a estimativa desses vetores, principalmente na proximidade das arestas e aplicar a segmentação novamente.

Uma segmentação final (com normais corretas devido aos passos anteriores) pode ser feita

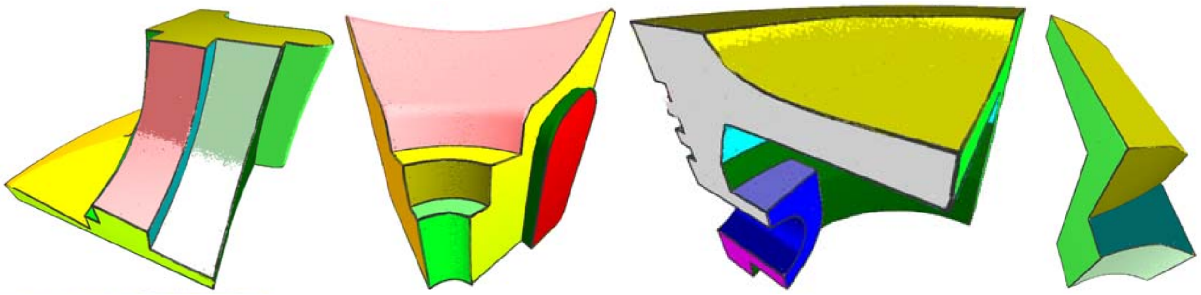

Figura 2.17: Segmentação proposto em [Daniels et al. 2008]. Utilizando-se do cálculo exato das linhas de descontinuidade, pode-se separar as diversas regiões.

utilizando apenas o ângulo α. Em seguida, uma segmentação de segunda ordem, que separa pontos com estimativas de curvatura destoantes, é feita. Essa segmentação tem por objetivo diferenciar um ponto pertencente a uma esfera de um ponto pertencente a um cilindro para finalidades de engenharia reversa. Essa segmentação é mais difícil, pois estimativas de curvaturas são mais sensíveis a ruído do que estimativas de normais. A Figura 2.16 mostra as etapas desta segmentação.

Unindo uma técnica de detecção de arestas com um algoritmo de crescimento de regiões, Daniels *et al*. [Daniels et al. 2008] propuseram uma segmentação com uma abordagem bastante simples. Utilizando-se das linhas de arestas construídas por meio do operador de projeção RMLS, eles identificam todos os pontos, cujas distâncias a essas linhas são menores do que um limiar d, como pontos de fronteira e os demais pontos, como pontos não visitados. A segmentação inicia-se ao escolher, ao acaso, um ponto não visitado e adicionar ao mesmo segmento todos os pontos não visitados que são seus vizinhos e que estão a uma distância menor do que d. Quando não for mais possível adicionar nenhum ponto a esse segmento, inicia-se outro segmento em outro ponto ainda não visitado. Quando não houver mais pontos não visitados, isso significa que faltam apenas os pontos de fronteira. Estes são adicionados ao segmento que contem o ponto mais próximo visitado. Apesar da abordagem relativamente simples, é preciso ter um cuidado na escolha da distância d. Essa distância não deve permitir que algum dos k vizinhos mais próximos de um determinado ponto pertença a um segmento diferente. Isso criaria uma ponte no grafo de vizinhança que ligaria dois segmentos. A Figura 2.17 mostra diversos modelos segmentados através do método proposto em [Daniels et al. 2008].

2.5 Reconstrução das Superfícies em Espaço de Tela

A técnica de surface splatting possui duas etapas principais. A primeira etapa se concentra na projeção das amostras sobre o espaço de tela. A segunda etapa escolhe os fragmentos visíveis,

e os combina para calcular a cor de um dado pixel, reconstruindo assim a superfície em espaço de tela.

Zwicker *et al*. [Zwicker et al. 2001] propuseram inicialmente uma adaptação do algoritmo de *z-buffer*. Todos os fragmentos são ordenados segundo sua profundidade e comparados em sequência. Entretanto, o teste de profundidade é ternário:

- o primeiro fragmento é escolhido.
- se o valor de profundidade do próximo fragmento for maior do que a profundidade do fragmento anterior acrescida de um valor, ε, especificado pelo usuário, então esse fragmento pertence a outra superfície e é descartado.
- se o valor de profundidade do próximo fragmento for menor do que a profundidade do fragmento anterior acrescita de ε, sua contribuição é adicionada a contribuição do fragmento anterior.

Apesar dessa estratégia ser uma boa estratégia em princípio, é complicado escolher um intervalo fixo para a imagem inteira. Buscando uma abordagem adaptativa, Räsänen [Räsänen 2002] propôs um intervalo adaptativo baseado nas propriedades locais da superfície. Nesse caso o intervalo é determinado utilizando valores máximos e mínimos de profundidade para a amostra projetada. Assim, splats são considerados como pertencentes à mesma superfície, em caso de sobreposição de seus intervalos.

3 *Grafo de Vizinhança*

Uma representação geométrica baseada em pontos pode ser considerada como uma amostragem de uma superfície contínua, resultando em posições 3D. Devido à ausência de conectividade, relações de vizinhança em nuvens de pontos são comumente usadas para estimar vetores normais à superfície. Computar a vizinhança é também necessário para estimar a curvatura local de uma superfície. Em certas aplicações, armazenar informações da vizinhança local pode ser adequado. Essa representação geométrica baseada em pontos pode ser comparada a um grafo, denominado *grafo de vizinhança*.

Neste capítulo, as estimativas de vizinhança mais utilizadas são descritas e uma análise de vantagens e desvantagens de cada abordagem em determinadas ocasiões é apresentada. A estimativa de vizinhança proposta é apresentada e explica-se a razão da sua escolha, mostrando as vantagens de utilizá-la na detecção de descontinuidades. Por fim, uma comparação é feita entre um método clássico de vizinhança e a abordagem proposta a fim de mostrar o propósito de sua escolha.

3.1 Motivação

Os métodos *r-ball* e *k-nearest* são os dois métodos mais relatados na literatura, para a estimativa de vizinhança de uma amostra dentro de uma geometria amostrada por pontos. Ambos os métodos utilizam distância Euclidiana. No método *r-ball*, todas as amostras internas à esfera de raio *r* centrada no ponto consultado são definidos como vizinhos deste ponto. Este método ingênuo é simples e as relações de vizinhança são simétricas, mas não é adequado para modelos amostrados irregularmente, pois uma quantidade muito grande ou muito pequena de pontos pode ser encontrada dentro da esfera [Kobbelt e Botsch 2004]. Tentando solucionar essa característica, Mitra *et al*. [Mitra e Nguyen 2003] estudaram métodos para escolher o raio adaptativamente em diferentes pontos da amostragem. Assumindo que o modelo apresentava ruído aleatório, eles propuseram métodos analíticos para limitar o erro da normal como função do raio e um algoritmo para estimar a curvatura local e o nível de ruído dos dados a fim de permitir a escolha

do raio ótimo.

Em contraste, o método *k-nearest*, que escolhe os k pontos mais próximos, provê uma abordagem adaptativa, pois mantém uma quantidade fixa de vizinhos. Entretanto, esse método requer ordenação das amostras e as relações de vizinhança não são simétricas. Essa abordagem é comparável com a primeira, onde o raio da esfera centrada na dada amostra é igual à distância ao seu k-ésimo vizinho. Definir o tamanho do raio r e o número k são escolhas críticas para ambos os métodos, pois, quando esses parâmetros são determinados arbitrariamente, vizinhos errados podem ser selecionados. Para evitar esse problema, Pauly *et al*. [Pauly et al. 2003] aumentam gradualmente os valores desses parâmetros enquanto monitoram os desvios padrões dos vetores normais nas amostras. Quando o desvio padrão aumenta abruptamente, os valores dos parâmetros param de crescer (Figura 2.2). Entretanto, esse procedimento consome muito tempo de processamento.

Os principais fatores para a escolha de uma estimativa de vizinhança são a distribuição das amostras fornecidas e o objetivo da aplicação que utilizará essa estimativa. Por exemplo, quando todos os pontos estão distribuídos em linhas ou fatias sobre a superfície, todos os k pontos mais próximos podem pertencer à mesma fatia, o que leva a uma estimativa ruim da normal à superfície naquele ponto. Utilizar a vizinhança *r-ball* com um raio fixo poderia solucionar alguns desses casos, mas acarretaria as mesmas dificuldades. A Seção 2.2 mostrou, que dependendo da aplicação, uma determinada estimativa de vizinhança pode ser vantajosa em relação às outras. Neste trabalho, o objetivo é encontrar os splats localizados nas arestas para recortá-los e adaptá-los ao formato da curva da descontinuidade. Uma boa vizinhança para esse tipo de aplicação deve ser local, para estimar com mais precisão os elementos presentes na descontinuidade, e representar bem a superfície local, no sentido de que representem a curvatura local da superfície de forma satisfatória, independente da forma da amostragem.

Em [Gumhold et al. 2001], utiliza-se a tetraedralização de Delaunay dos pontos do modelo, aplica-se uma série de filtros para remover triângulos dos tetraedros não pertencentes à superfície, e as arestas da malha resultante tornam-se relações de vizinhança entre os pontos. Apesar dessa vizinhança apresentar características bastante locais e funcionar bem para amostragens não uniformes, os filtros de triângulos da tetraedralização dependem do modelo, pois buracos podem ser conectados levando a disparidades de vizinhanças, mesmo entre dois pontos próximos. Outra vizinhança que atende a esses requisitos é a vizinhança em anel. Utilizada por [Guennebaud et al. 2004] para refinar a amostragem do modelo, essa vizinhança, após selecionar os k vizinhos mais próximos e calcular a distância geodésica desses ao ponto consultado, aplica uma seleção fuzzy que diminui o conjunto dos vizinhos e deixa o conjunto resultante

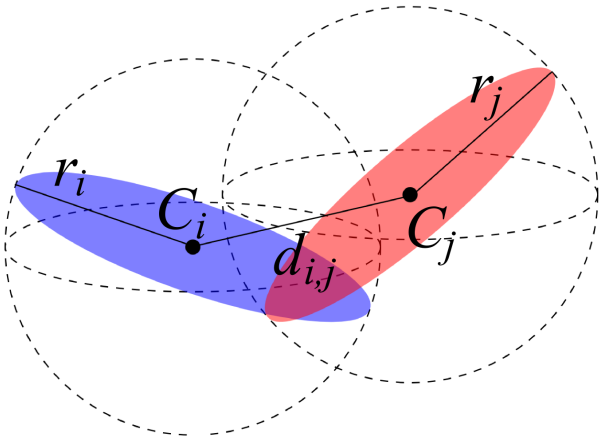

Figura 3.1: Condição proposta para dois *splats* serem vizinhos.

espaçado quase uniformemente ao redor do ponto. Mesmo apresentando melhores resultados do que as demais vizinhanças, o seu ponto fraco deve-se à complexidade de formá-la.

3.2 Vizinhança Através de Esferas Limitantes

As vizinhanças propostas anteriormente são utilizadas para modelos baseados em pontos e, consequentemente, podem ser aplicadas a modelos baseados em splats. Entretanto, apesar dessas vizinhanças serem robustas, elas não exploram as vantagens da orientação e da dimensionalidade dos splats. Além de explorar vantagens do tipo de modelo, deve-se escolher uma vizinhança que sirva bem ao objetivo de identificar os splats que necessitam de recorte. Todos o splats que precisam ser recortados, cruzam outros splats do outro lado da aresta. Dessa forma, é possível definir uma vizinhança de um splat através da união dos splats que o interceptam. Essa estimativa identifica, para cada splat, os splats que o recortam, pois esses certamente o cruzam. Entretanto, muita computação é gasta para identificar se dois discos no espaço de objeto se cruzam.

Uma alternativa mais simples e eficiente consiste em adaptar a abordagem anterior para detectar quando dois splats não são vizinhos. Se as esferas limitantes de dois splats não se tocam, certamente esses splats não se cruzam. Dessa forma, dois splats são ditos vizinhos se a distância entre os centros de suas esferas limitantes é menor do que a soma de seus raios (Figura 3.1). O cálculo da esfera limitante de um splat é muito simples. A esfera limitante de um splat circular tem o mesmo centro e o mesmo raio do splat. Para um splat elíptico, a esfera limitante tem o mesmo centro, mas seu raio é igual ao maior semi-eixo da elipse.

Um splat localizado longe das arestas não tem qualquer um de seus vizinhos localizado em outra superfície. Entretanto, amostras em regiões próximas às arestas, aos cantos e às inter-

seções de superfícies que se cruzam podem ser classificadas como vizinhas, mesmo quando elas não se tocam. Isso resulta em um maior número de candidatos a recorte do que o necessário, entretanto, o custo computacional envolvido em determinar exatamente se dois splats se cruzam é muito alto comparado ao ganho.

Baseado em nossa definição de vizinhança, um grafo de vizinhança, $G = (V,E)$, pode ser construído. Neste grafo, V representa todo o conjunto de splats do modelo, e E o conjunto de arestas:

$$E = \{(i,j)|d(C_i,C_j) = d_{ij} < r_i + r_j; S_i \in V; S_j \in V\}, \tag{3.1}$$

onde, S_i e S_j são splats do modelo com centros C_i e C_j, respectivamente, d_{ij} é a distância entre os centros dos splats citados, e r_i e r_j são os raios das esferas limitantes dos respectivos splats.

3.3 Detecção de Arestas e Cantos

Diversos métodos de detecção de arestas e cantos utilizam uma função para estimar a curvatura de um ponto e ponderam cada amostra de acordo com sua proximidade de uma aresta. Um limiar identifica os pontos pertencentes a descontinuidades. Assim, as arestas que os conectam no grafo de vizinhança também são ponderadas. Então, uma árvore geradora mínima é computada para estimar a linha da aresta [Gumhold et al. 2001, Pauly et al. 2003]. Entretanto, o objetivo não é identificar as arestas que formam a descontinuidade, mas sim as arestas do grafo de vizinhança que conectam splats de uma superfície de um lado da aresta a um splat pertencente ao outro lado da descontinuidade.

Para encontrar essas arestas, utiliza-se a estratégia adotada em [Kobbelt et al. 2001], pois os vetores normais dos splats estão disponíveis desde o princípio. Sabe-se que quanto maior o ângulo entre os vetores normais de splats que se interceptam, maior o número de fragmentos distantes que aparecem na etapa de reconstrução de superfície. Combinar esses fragmentos distantes durante a etapa de reconstrução de superfície exibirá borramento nas descontinuidades (Figura 2.10). Por isso, o método limita o ângulo máximo entre dois splats que se cruzam. O ângulo é monitorado através do produto escalar dos vetores normais aos splats. Quanto mais próximo de 1.0 for o produto escalar desses vetores normais, mais plana é a região entre os splats analisados. Dessa forma, seguindo o caminho inverso dos trabalhos anteriores, as arestas são ponderadas utilizando o produto escalar dos vetores normais aos splats conectados por essa aresta, ou seja:

$$w_{ij} = n_i \cdot n_j, (i,j) \in E, \tag{3.2}$$

onde w_{ij} é o peso da aresta (i,j) e n_i e n_j são os vetores unitários normais aos splats S_i e S_j,

respectivamente.

Pode-se definir um subconjunto de arestas $E_s \subset E$ que contêm todas as arestas com peso menor do que um limiar w_s escolhido pelo usuário. Os dois splats associados a uma aresta em E_s podem ser interpretados como pertencentes a diferentes superfícies e, possivelmente, há uma descontinuidade entre eles. Essas arestas são chamadas *sharp-edges*, e todas as outras arestas são chamadas *continuous-edges*. Matematicamente, E_s é definido como

$$E_s = \{(i, j) | w_{ij} < w_s\}. \tag{3.3}$$

Uma escolha típica para o limiar w_s é $0,9$, como dito em [Kobbelt et al. 2001]. Entretanto, não é necessário utilizar o outro limiar proposto por Kobbelt *et al.* para verificar se o splat pertence a um canto. Isso acontece naturalmente, pois a limiarização ocorre sobre as arestas e não sobre as amostras. Caso um splat apresente mais de uma sharp-edge, este splat é interpretado como um canto realizando diversos recortes.

3.4 Análise da Vizinhança de Esferas Limitantes

Como dito na Seção 3.1, uma estimativa de vizinhança pode ser melhor do que outra, dependendo do tipo de amostragem e da aplicação que a utilizará. Nesta aplicação, onde o objetivo é detectar os splats a serem recortados, busca-se uma vizinhança que conecte o mínimo necessário para evitar posteriores cálculos de interseção de discos sem deixar passar splats presentes nas arestas. Devido ao cálculo simples usado na detecção de splats pertencentes a superfícies diferentes, a vizinhança utilizada deve atender essas características. Nesta seção, comparam-se as estimativas *k-nearest*, uma das estimativas mais relacionadas na literatura, e as esferas limitantes.

A quantidade de arestas no grafo é uma forma de analisar a quantidade de produtos escalares e, principalmente, a quantidade de detecções de colisão necessárias para identificar candidatos a recorte. Seja m o número médio de arestas por splat determinado por meio do método de esferas limitantes. Assim a quantidade de arestas do grafo é aproximadamente:

$$|E| \approx m \cdot \frac{|V|}{2}. \tag{3.4}$$

Essa quantidade de arestas analisadas é dividida por 2, pois elas são simétricas, por isso são contadas duas vezes para cada splat. No método *k-nearest*, as arestas não são simétricas, e

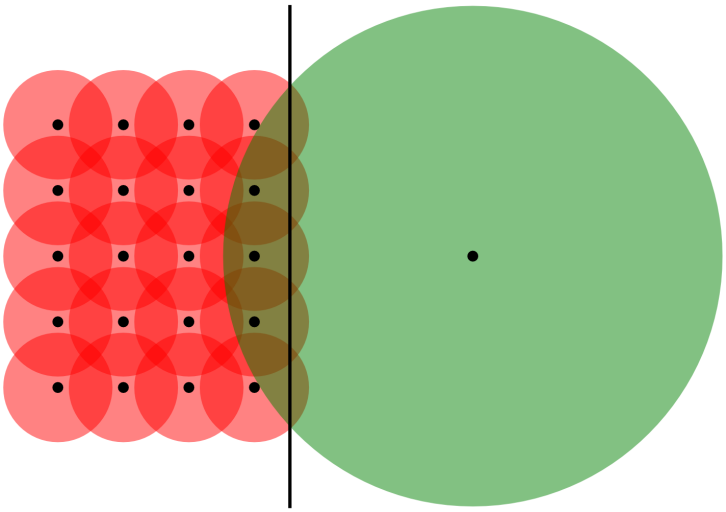

Figura 3.2: Mesmo com densidades de amostragem diferentes em torno de uma aresta, a estimativa de vizinhança utilizando esferas limitantes garante arestas entre os splats em torno da descontinuidade.

assim, o número de arestas a serem analisadas é igual a

$$|E| \approx k \cdot |V|. \tag{3.5}$$

A partir das equações 3.4 e 3.5, conclui-se que $k \approx \frac{m}{2}$. Entretanto, em diversos testes feitos, verificou-se que os modelos cujas amostragens são aproximadamente uniformes, e no qual os raios dos splats são calculados como a distância média aos cinco pontos mais próximos, apresentam $m \approx 13$. Ou seja, para que o grafo de vizinhança que usa o método *k-nearest* possuir um número total de arestas aproximado do grafo de vizinhança que usa o método de esferas limitantes, k deve ser aproximadamente 7 ou 8, um valor muito baixo segundo a maioria dos autores. Ao reduzir em 10% o raio de todos os splats dos mesmos modelos utilizados nos testes anteriores, o número médio de arestas usando o método de esferas limitantes, m, é aproximadamente igual a 8. Mesmo com essa baixa quantidade de arestas, a estimativa de esferas limitantes consegue detectar todos os splats presentes nas descontinuidades, pois todos os splats presentes nessas áreas continuaram se interceptando.

Outra preocupação deve-se às densidades de amostragem ao redor de uma aresta. Considerando-se o exemplo mostrado na Figura 3.2, é possível que, ao utilizar o método de *k-nearest*, todos os splats vermelhos não apresentem uma única aresta ligando-os ao splat verde, acarretando o não recorte do splat e artefatos. Mesmo aumentando o valor de k, esse valor dependerá da amostragem. Entretanto, ao utilizar o método de esferas limitantes, esse caso sempre será solucionado, pois independente da amostragem, se os splats não se interceptarem, haverá buracos próximos às arestas por causa das amostras.

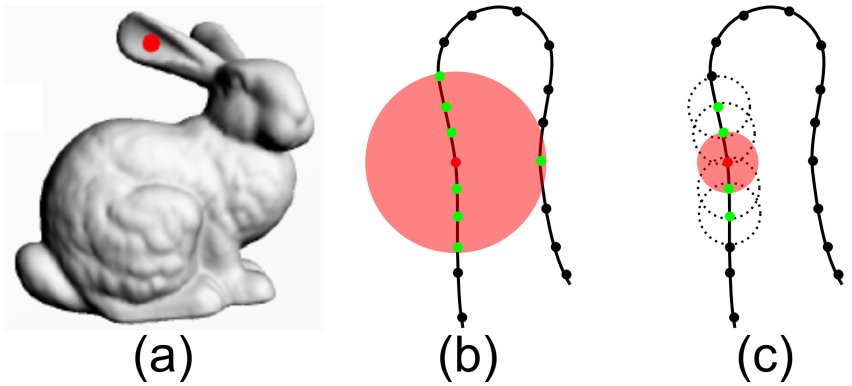

Figura 3.3: Comparação entre vizinhanças *k-nearest* e esferas limitantes. (a) Para um splat localizado na orelha do Stanford Bunny. (b) Falsos positivos podem ocorrer devido um splat longe estar entre os k vizinhos mais próximos. (c) Caso os splats apresentem raios menores do que o espaçamento entre as superfícies que estes formam, as vizinhanças não cruzam as superfícies não formando falsos positivos fora das proximidades das descontinuidades.

Apesar da estimativa proposta detectar as amostras presentes em arestas, independente das diferenças na densidade de amostragem, uma boa vizinhança deveria conectar o mínimo de falsos positivos, ou seja, aqueles splats que foram detectados como splats de aresta, quando, na verdade, não precisam de recorte. Um falso positivo pode acontecer no método de *k-nearest* devido à escolha crítica do valor k, tornando-se mais complicado próximo às descontinuidades (Figura 3.3b). No método de esferas limitantes, um falso positivo pode acontecer se duas superfícies não formarem aresta, mas um splat exageradamente grande formar uma esfera que entre em contato com a outra superfície. Esse problema é ruim para qualquer método, e, geralmente, o método de esferas limitantes não conecta splats de superfícies diferentes desde que as superfícies estejam mais distantes umas das outras do que o tamanho dos splats que as formam (Figura 3.3c). Mesmo que essa condição seja satisfeita nas superfícies em geral, elas não podem ser satisfeitas próximo das descontinuidades. Sendo assim, o método de esferas limitantes depende do ângulo formado pelas superfícies na aresta. Quanto mais agudo for o ângulo, mais esferas se interceptam, gerando mais falsos positivos.

A Figura 3.4 ilustra a relação entre ângulo entre superfícies e quantidade de falsos positivos determinados. Para visualizar a disposição dos splats, esses foram renderizados com 40% de seu raio original. Splats vermelhos apresentam pelo menos uma *sharp-edge*, ou seja, são candidatos a recorte. Splats verdes apresentam apenas *continuous-edges*. Mesmo para um ângulo relativamente grande nas arestas, ocorre o aparecimento de falsos positivos devido à sobreposição dos splats (Figura 3.4a), mas quando o ângulo torna-se mais agudo, esferas limitantes dos splats próximos à aresta começam a se interceptar, aumentando a quantidade de falsos positivos (Figura 3.4b). Achatar a esfera limitante na direção do vetor normal do splat, tornaria

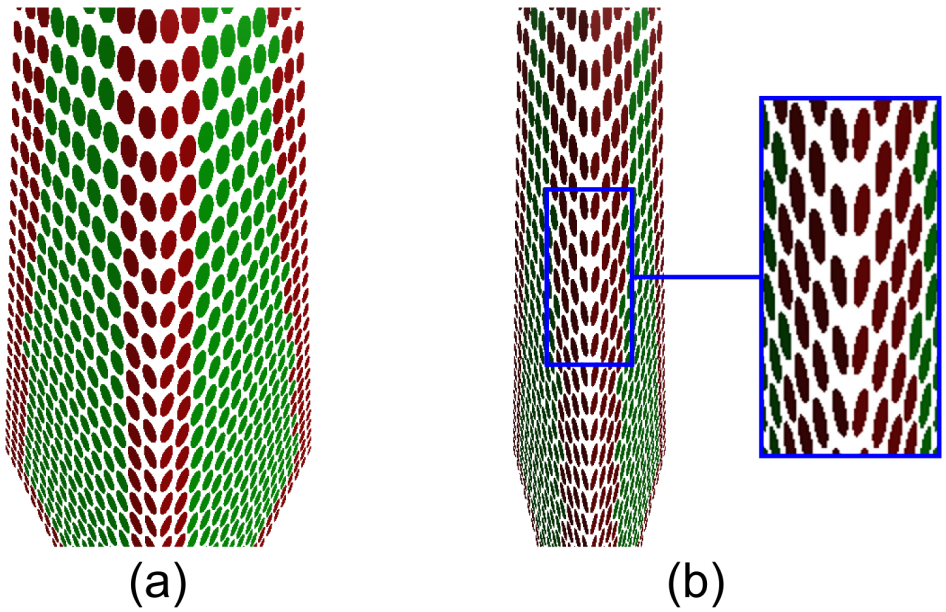

Figura 3.4: Relação ângulo da descontinuidade e inserção de falsos positivos na estimativa de esferas limitantes. (a) Para ângulo maiores, a inserção de falsos positivos ocorre devido a sobreposição dos splats. (b) Mas quando o ângulo torna-se muito agudo, mais esferas limitantes se interceptam causando o aparecimento de mais falsos positivos.

a estimativa de vizinhança mais precisa e encontraria um menor número de falsos positivos em ângulos mais agudos. Entretanto, os cálculos das interseções entre esses elipsóides começaria a se tornar complexos em comparação ao ganho com eficiência.

3.5 Considerações Finais

O método de detecção de arestas proposto em [Kobbelt et al. 2001] é bastante simples e não robusto, necessitando que a estimativa de vizinhança seja a mais fiel possível à superfície implícita. Podem-se utilizar outros métodos de detecção de arestas mais complexos no mesmo grafo de vizinhança proposto, mas um modelo formado por splats permite uma série de simplificações devido a sua orientação pelo vetor normal e pela sua dimensão fornecida pelo raio, no caso de splats circulares, ou eixos, no caso de splats elípticos. Explorar essas simplificações para alcançar eficiência sem perder a robustez da detecção é o principal objetivo para realizar a detecção e a renderização de arestas em tempo real.

4 *Recorte de Splats*

No Capítulo 3, um grafo de vizinhança e um método de detecção de descontinuidades foram definidos. Agora os splats precisam ser adaptados a essas descontinuidades tomando por entrada a estrutura de dados gerada e a classificação das arestas do grafo em *sharp-edges* e *continuous-edges*. Para representar descontinuidades com apenas discos, seria necessário uma quantidade infinita de amostras, por esse motivo, os métodos existentes recortam splats, uns contra os outros, para adaptar a linha da descontinuidade [Pauly et al. 2003, Zwicker et al. 2004, Wicke et al. 2004]. No entanto, antes de recortar um splat contra todos os splats ligados a ele através de uma *sharp-edge*, deve-se ter em mente que a vizinhança deixou passar alguns falsos positivos, ou seja, splats que não precisam ser recortados, mas que foram detectados como candidatos a recorte. Além disso, deve-se identificar quando uma interseção de superfícies foi detectada, pois interseções de superfícies também formam diversas *sharp-edges* no grafo, sem que os splats envolvidos precisem ser recortados.

Após a remoção de todas as amostras que não precisam ser recortadas, o recorte deve adaptar-se à curva da descontinuidade. Para isso, é feita uma adaptação do método de Wicke *et al*. [Wicke et al. 2004] realizando recortes côncavos ou convexos para uma melhor adaptação à aresta. Resumindo, o método de adaptação dos splats apresenta quatro etapas: 1) identificação de uma descontinuidade para diferenciar o que é uma aresta e o que é uma interseção de superfícies; 2) identificação de interseção de splats para remoção dos falsos positivos; 3) classificação dos parceiros de recorte; e, finalmente, 4) remoção dos fragmentos do splat durante sua rasterização.

4.1 Identificando uma Descontinuidade

Após a construção do grafo de vizinhança e a aplicação de pesos às arestas, as descontinuidades podem ser tratadas. Entretanto, primeiro precisam-se diferenciar as diversas características detectadas. Um canto pode ser interpretado como uma simples descontinuidade, porque cada splat pode armazenar mais de uma *sharp-edge* com outras amostras que, por sua vez, podem ter

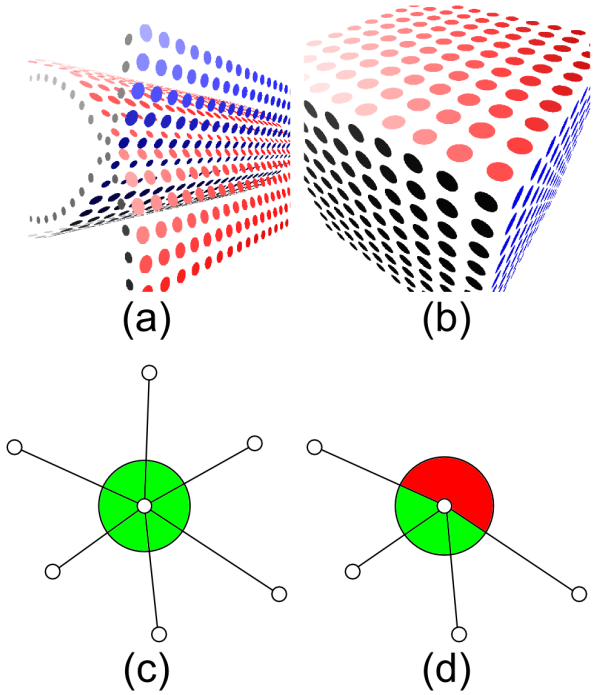

Figura 4.1: Identificação de aresta pelo método de Linsen [Linsen 2001]. (a) Superfícies que se interceptam. (b) Arestas. (c) Superfície contínua - todos os ângulos consecutivos são menores do que um limite. (d) Identificação de aresta - um ângulo consecutivo é maior do que o limite pré-definido.

sharp-edges entre elas. Entretanto, um problema mais difícil é distinguir uma aresta, de uma interseção de superfícies.

Para resolver esse problema, pode-se utilizar a técnica de Linsen [Linsen 2001] para detecção de fronteiras. As *sharp-features* localizadas em uma aresta dividem o modelo em superfícies suaves, formando fronteiras (Figura 4.1b). Isso é diferente das *sharp-edges* localizadas em uma interseção de superfícies (Figura 4.1a). Considere que um splat S_i seja adjacente a uma *sharp-edge* e que $N_c(S_i)$ seja o conjunto dos splats conectados a S_i através de uma *continuous-edge*. Considere ainda que $S_j \in N_c(S_i)$, que $\mathbf{C}_j$ seja o centro do splat S_j, e que $\mathbf{P}_j$ seja a projeção de $\mathbf{C}_j$ sobre o plano definido por S_i. Assim, assumindo-se que os vetores $\mathbf{P}_j - \mathbf{C}_i$ foram ordenados por algum método de ângulo orientado, se o ângulo entre dois vetores consecutivos for maior do que um limite α pré-determinado, S_i é classificado como pertencente a uma fronteira da superfície suave, ou seja, S_i está localizado numa aresta (figuras 4.1b e 4.1d). Por outro lado, se todos os ângulos consecutivos forem menores do que o limite α, então S_i está numa superfície contínua, mas a razão para estar conectado a uma *sharp-edge* é sua interseção com outra superfície suave (figuras 4.1a e 4.1c). O parâmetro α controla o tamanho dos buracos que são detectados como fronteiras.

Por razões de eficiência, os valores das projeções $\mathbf{P}_j$ não precisam ser calculadas com ex-

atidão, pois essas projeções são necessárias apenas para ordenar os ângulos dos vetores. Dessa forma, o vetor $\mathbf{v}_j \equiv \overrightarrow{\mathbf{C}_i\mathbf{P}_j}$ pode ser calculado como

$$\mathbf{v}_j = \mathbf{n}_i \times ((\mathbf{C}_j - \mathbf{C}_i) \times \mathbf{n}_i). \tag{4.1}$$

Para fins de ordenação, o vetor $\mathbf{v'}_j$ dado por

$$\mathbf{v'}_j = (\mathbf{C}_j - \mathbf{C}_i) \times \mathbf{n}_i. \tag{4.2}$$

pode ser usado em lugar do vetor $\mathbf{v}_j$, já que $\mathbf{v'}_j$ está no plano do splat, S_i, e é defasado de 90 graus em relação ao vetor $\mathbf{v}_j$. Assim, $\mathbf{v'}_j$ pode ser usado eficientemente no método de detecção de fronteiras.

4.2 Identificando Interseção de Splats

O próximo passo é descobrir se dois splats se interceptam. Os cálculos envolvidos nesse processo visam excluir os falsos positivos e permitir o recorte apenas dos splats que realmente se cruzem e formem um ângulo maior do que um limiar pré-determinado. Nesse estágio, esses cálculos são bastante eficientes por levarem em conta apenas os splats associados a *sharp-edges*, que, em geral, são poucos quando comparados ao número total de splats.

Para identificar a interseção entre dois discos no espaço de objeto, alguns passos são necessários. Inicialmente, detecta-se a reta de interseção dos planos formados pelos splats envolvidos. Essa reta de interseção deve interceptar ambos os discos formando segmentos de reta. Para concluir o método de interseção, verifica-se se esses segmentos de interseção apresentam algum trecho em comum. No caso em que a interseção persista até esse último teste, então conclui-se que os discos interceptam-se, caso contrário, essa aresta do grafo é desconsiderada e os splats não se recortam.

4.2.1 Reta de Interseção de Planos

O primeiro passo para realizar a detecção de colisão entre dois círculos ou elipses é descobrir se seus planos se interceptam e calcular a reta de interseção. Para uma *sharp-edge* (i, j), os planos dos splats S_i e S_j são definidos como

$$\begin{aligned} \alpha_i &= \{\mathbf{P} \mid \mathbf{n}_i \cdot \mathbf{P} = \mathbf{n}_i \cdot \mathbf{C}_i\} \\ \alpha_j &= \{\mathbf{P} \mid \mathbf{n}_j \cdot \mathbf{P} = \mathbf{n}_j \cdot \mathbf{C}_j\}. \end{aligned} \tag{4.3}$$

A direção da reta de interseção é calculada como

$$\mathbf{d} = \mathbf{n}_i \times \mathbf{n}_j. \tag{4.4}$$

Quando o produto vetorial na Equação 4.4 é zero, os dois planos são paralelos ou coincidentes. Entretanto, essa condição não deve ser satisfeita, pois é contrária à definição de *sharp-edge* e sabe-se que o ângulo entre os vetores normais $\mathbf{n}_i$ e $\mathbf{n}_j$ é maior do que o limiar pré-definido. Então, os vetores $\mathbf{d}$, $\mathbf{n}_i$ e $\mathbf{n}_j$ formam uma base não-ortogonal para o espaço Euclidiano $\mathbb{R}^3$. Dessa forma, usando a origem canônica e esses vetores-base para formar um espaço de referência, qualquer ponto no espaço pode ser escrito da forma

$$\mathbf{P} = t_i\mathbf{n}_i + t_j\mathbf{n}_j + t\mathbf{d}. \tag{4.5}$$

A Equação 4.5 pode ser usada para representar a equação paramétrica da reta de interseção, se t for definido como o parâmetro da reta e $t_i\mathbf{n}_i + t_j\mathbf{n}_j$ como o vetor posição, relativo à origem canônica, do ponto $\mathbf{P}_o$ onde a linha corta o plano $\mathbf{n}_i\mathbf{n}_j$. Assume-se que t seja igual a zero nesse ponto. Dessa forma, ao utilizar o produto escalar de $\mathbf{P}_o$ com $\mathbf{n}_i$ e $\mathbf{n}_j$, obtém-se o seguinte sistema de duas equações a duas incógnitas t_i e t_j.

$$\begin{cases} \mathbf{P}_o \cdot \mathbf{n}_i = t_i(\mathbf{n}_i \cdot \mathbf{n}_i) + t_j(\mathbf{n}_j \cdot \mathbf{n}_i) \\ \mathbf{P}_o \cdot \mathbf{n}_j = t_i(\mathbf{n}_i \cdot \mathbf{n}_j) + t_j(\mathbf{n}_j \cdot \mathbf{n}_j) \end{cases} \tag{4.6}$$

Ao encontrar as soluções do sistema 4.6, $\bar{t}_i$ e $\bar{t}_j$, e substituí-las na Equação 4.5, encontram-se todos os pontos localizados na interseção dos planos formados pelos splats S_i e S_j, ou seja, a reta

$$\mathbf{P} = \mathbf{P}_o + t\mathbf{d} \text{ ,onde } \mathbf{P}_o = \bar{t}_i\mathbf{n}_i + \bar{t}_j\mathbf{n}_j \tag{4.7}$$

4.2.2 Interseção entre Reta e Splat

Quando duas esferas limitantes não se tocam, é garantido que os splats não se intersectam. Entretanto, quando o oposto acontece, não se pode dizer com certeza que a interseção ocorrerá. Por esse motivo, próximo de arestas, é possível que um splat possua *sharp-edges* com outras amostras sem intersectá-las, realizando um recorte desnecessário. Também é possível que a reta de interseção dos planos de dois splats cruze um deles, mas não o outro. Nesse caso, recortar apenas um dos splats pode gerar buracos na superfície. Por essa razão, é muito importante saber quando uma reta cruza um splat circular ou elíptico.

A reta de interseção dos planos é definida pelo ponto $\mathbf{P}_o$ e pelo vetor d encontrados na

Equação 4.7. Os vetores unitários $\mathbf{e}_u$ e $\mathbf{e}_v$ ao longo dos eixos principais de um splat elíptico, o vetor unitário $\mathbf{n}$, normal à superfície, e seu centro, $\mathbf{C}$, formam um espaço de referência (espaço de referência do splat) onde os parâmetros da reta de interseção dos planos podem ser escritos. Note que essa reta pertence ao plano do splat. Dessa forma, após a transformação para o espaço de referência do splat, as componentes na direção da normal ao splat são iguais a zero. Assim, $\mathbf{P}_o$ e $\mathbf{d}$ podem ser reescritos como

$$\begin{aligned} &\mathbf{P}_o = (P_{o_u}, P_{o_v}) \leftrightarrow \mathbf{P}_o = P_{o_u}\mathbf{e}_u + P_{o_v}\mathbf{e}_v \\ &\mathbf{d} = (d_u, d_v) \leftrightarrow \mathbf{d} = d_u\mathbf{e}_u + d_v\mathbf{e}_v. \end{aligned} \tag{4.8}$$

No sistema de coordenadas 2D definido pelos vetores-base $\mathbf{e}_u$ e $\mathbf{e}_v$ (espaço de referência do splat), a equação paramétrica da reta de interseção pode ser escrita como

$$\mathbf{P}(t) = \mathbf{P}_o + t\mathbf{d} \tag{4.9}$$

e a equação da elipse, como

$$\frac{u^2}{r^2} + \frac{v^2}{s^2} = 1 \tag{4.10}$$

onde r e s são os semi-eixos da elipse nas direções de $\mathbf{e}_u$ e $\mathbf{e}_v$, respectivamente. Os pontos de interseção da reta com o splat elíptico são as raízes da seguinte equação quadrática em t

$$At^2 + 2Bt + C = 0, \text{ onde } \begin{cases} A = \frac{d_u^2}{r^2} + \frac{d_v^2}{s^2} \\ B = \frac{P_{o_u} d_u}{r^2} + \frac{P_{o_v} d_v}{s^2} \\ C = \frac{P_{o_u}^2}{r^2} + \frac{P_{o_v}^2}{s^2} - 1 \end{cases} \tag{4.11}$$

Um splat circular é um caso particular de um splat elíptico onde $r = s$. Dessa forma, a interseção de uma reta com o splat pode ser calculada com uma versão simplificada da Equação 4.11 como

$$At^2 + 2Bt + C = 0, \text{ onde } \begin{cases} A = \mathbf{d} \cdot \mathbf{d} \\ B = \mathbf{P}_o \cdot \mathbf{d} \\ C = \mathbf{P}_o \cdot \mathbf{P}_o - r^2 \end{cases} \tag{4.12}$$

As soluções da Equação 4.11 ou da Equação 4.12, t_1 e t_2, são utilizadas para determinar o segmento de reta resultante da interseção entre a reta de interseção dos planos e um splat.

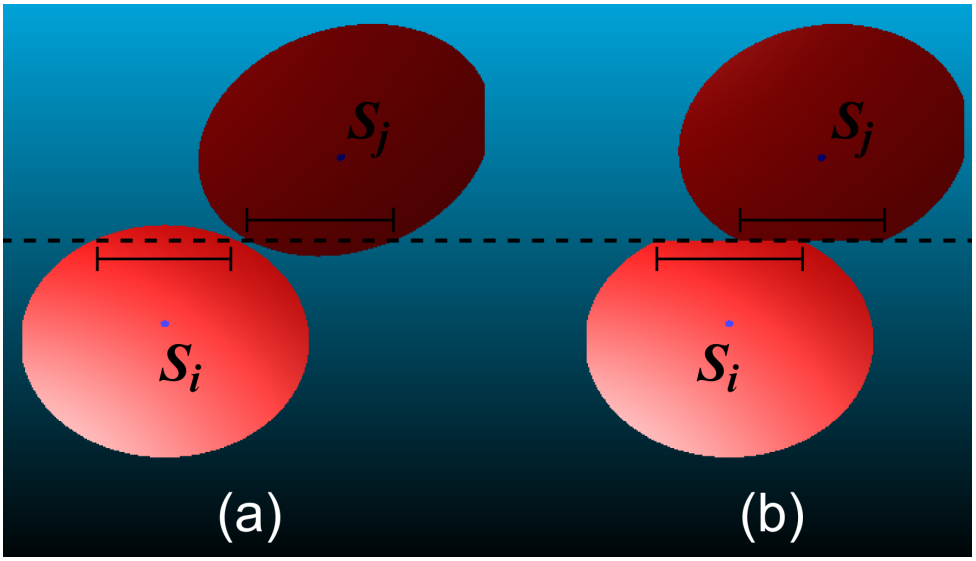

Figura 4.2: Necessidade do teste de interseção de segmentos para detecção de falsos positivos. (a) Mesmo ambos os splats interceptando a reta de interseção, se os segmentos não se interceptarem, eles não se cruzam, não precisando de recorte. (b) Caso os segmentos apresentem alguma interseção, o recorte é necessário e feito sobre a reta de interseção.

4.2.3 Interseção entre Segmentos

Uma vez que a reta de interseção dos dois planos dos splats e os segmentos de sua interseção com cada um dos splats utilizados para calculá-la tenham sido determinados, um último teste de interseção é feito para verificar o cruzamento desses dois splats. Esse teste é necessário, pois, mesmo que as esferas limitantes dos splats se interceptem, e mesmo que esses splats interceptem a reta de interseção dos seus planos, é possível que esses splats não se toquem (Figura 4.2a).

Seja t_{i_1} e t_{i_2} as soluções da Equação 4.11 resultantes da interseção da reta com o splat S_i, onde $t_{i_2} > t_{i_1}$. Da mesma forma, seja t_{j_1} e t_{j_2} as soluções da Equação 4.11 resultantes da interseção da reta com o splat S_j, onde $t_{j_2} > t_{j_1}$. Se os intervalos $[t_{i_1}, t_{i_2}]$ e $[t_{j_1}, t_{j_2}]$ não forem disjuntos, os splats se cruzam e precisam ser recortados (Figura 4.2b), caso contrário, um está na lateral do outro, mas não há cruzamento, identificando um falso positivo.

Após esse último teste, os vizinhos, através de *sharp-edges*, que realmente interceptam um determinado splat são denominados seus parceiros de recorte, ou, *clip partners* [Wicke et al. 2004].

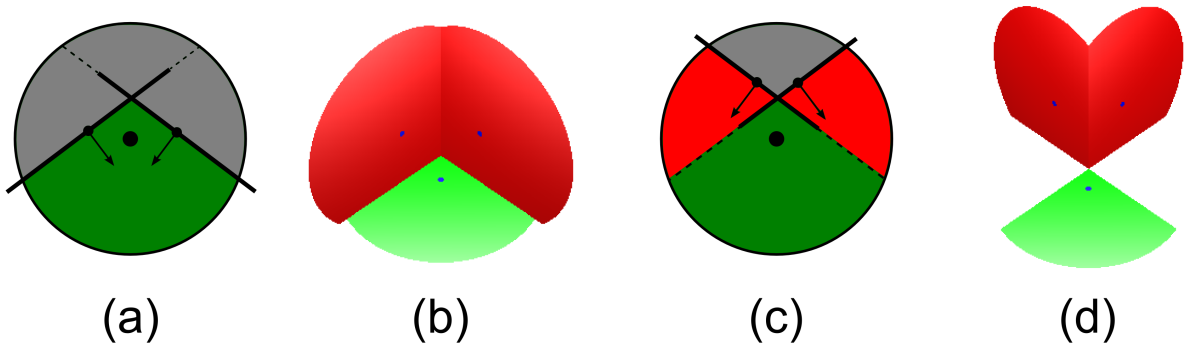

Figura 4.3: Caso de ambiguidade não tratado pelo método de Zwicker [Zwicker et al. 2004]. (a) Esquema onde o recorte contra ambos os *clip partners* adapta bem o splat. (b) Modelo que exemplifica o esquema (a). (c) Esquema onde o recorte contra ambos os *clip partners* gera buracos no splat. (d) Modelo que exemplifica o esquema (c).

4.3 Classificação dos Planos de Corte

Para adaptar splats a arestas, esses devem ser recortados pelo plano de outro splat vizinho que o intercepta. Esses splats podem compartilhar o mesmo centro [Pauly et al. 2003] ou não [Zwicker et al. 2004, Wicke et al. 2004]. Assim, como em [Wicke et al. 2004], será utilizado o termo *clip partner* para denominar o vizinho que participará do processo de recorte do splat. Em [Pauly et al. 2003], as amostras são posicionadas sobre a linha da descontinuidade, e, então, dois splats com mesmo centro recortam-se um ao outro, ou seja, uma amostra apresenta apenas um único *clip partner*. No caso de um canto, o processo de posicionamento da amostra é similar, porém mais splats são posicionados sobre o mesmo ponto e recortados entre si. Além do processo não ser geral, ou seja, precisa-se diferenciar arestas e cantos, ele necessita da inserção de amostras que não existem no modelo original.

Zwicker *et al.* [Zwicker et al. 2004] generalizaram a ideia de recortar splats, retirando a necessidade dos splats possuírem centros coincidentes. Nesse método, um splat detectado como pertencente a uma aresta é recortado pela reta de interseção do plano onde está inserido com o plano do *clip partner*. A quantidade de *clip partners* usada para recortar a amostra fica a critério do usuário, entretanto, essa escolha não é trivial. Utilizar apenas o *clip partner* mais próximo além de não representar adequadamente os cantos, apresenta problemas na representação de descontinuidades curvas, pois o recorte é uma reta. Recortar contra todos os *clip partners* traz um problema ainda mais sério, a ambiguidade. A Figura 4.3 exemplifica dois casos tratados de forma idêntica pelo método de Zwicker, mas que necessitam de tratamento diferenciado. No caso ilustrado nas figuras 4.3a e 4.3b, a união das áreas recortadas por cada um dos *clip partners* vermelhos adapta de forma correta o splat verde. Entretanto, o caso exemplificado nas figuras 4.3c e 4.3d, que apresenta as mesmas retas de interseção do caso anterior, não pode

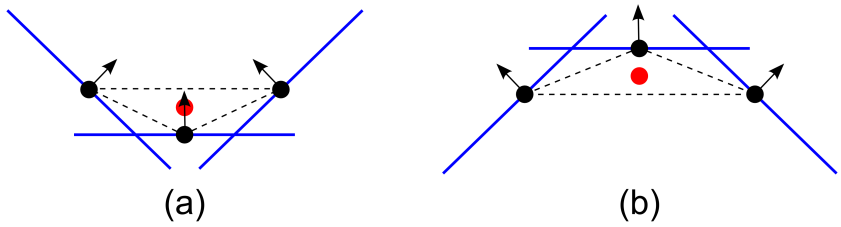

Figura 4.4: Classificação entre região côncava e convexa. (a) Em regiões côncavas, todos os elementos "veem" o ponto médio de seus centros. (b) Em regiões convexas, nenhum elemento "vê" o ponto médio.

ser tratado de forma idêntica, pois algumas áreas recortadas estão na frente dos splats da outra superfície. Nesse caso, uma forma de adaptar corretamente o splat verde é recortar apenas a interseção das áreas de recorte dos *clip partners*. Ao realizar um recorte independente dos demais, um fragmento é removido se estiver na área de recorte de pelo menos um dos *clip partners*. Dessa forma, independente do caso, um fragmento é removido se estiver na união das áreas de recorte. Para evitar essas ambiguidades os *clip partners* devem ser previamente analisados para realizar um recorte adaptado sobre o splat.

Wicke *et al*. [Wicke et al. 2004] utilizaram os dois splats mais próximos para classificar quando um ponto $3D$ está dentro ou fora de um objeto representado por splats. Essa função é necessária para realizar operações CSG com mais precisão. A Seção 2.3 mostra a classificação de dois splats formando regiões côncavas e convexas. Se os splats formam uma região côncava, um ponto é externo ao objeto se, e somente se, estiver na interseção dos semi-espaços formados pelos centros e pelas normais desses splats. Se eles formam uma região convexa, um ponto é externo ao objeto, se, e somente se, estiver na união dos semi-espaços formados pelos centros e pelas normais dos splats. Essa classificação recorta os splats de forma correta, pois para cada fragmento, recupera-se sua posição em espaço de objeto e verifica-se se esse ponto pertence ao objeto final através dessa classificação e da árvore CSG.

Assim como em [Wicke et al. 2004], os *clip partners* precisam ser classificados segundo sua orientação para decidir entre interseção ou união das áreas de recorte. Para identificar áreas côncavas ou convexas entre o conjunto de *clip partners*, pode-se calcular o ponto médio dos centros dos elementos desse conjunto. Em um conjunto côncavo, o ponto médio é "visto" por todos os elementos do conjunto (Figura 4.4a). Em um conjunto convexo, o ponto médio não é "visto" por nenhum dos elementos (Figura 4.4b). Um ponto é "visto" por um splat se estiver no semi-espaço definido pelo centro e pela normal do splat, ou seja, se $(\mathbf{P} - \mathbf{C}_i) \cdot \mathbf{n}_i > 0$, onde $\mathbf{P}$

é um ponto qualquer e $\mathbf{C}_i$ e $\mathbf{n}_i$ são o centro e a normal de um splat S_i, respectivamente. Ou seja:

$$\begin{cases} N_{cp} \text{ é côncavo} & \Leftrightarrow & \forall S_i \in N_{cp}; (\mathbf{M} - \mathbf{C}_i) \cdot \mathbf{n}_i > 0 \\ N_{cp} \text{ é convexo} & \Leftrightarrow & \forall S_i \in N_{cp}; (\mathbf{M} - \mathbf{C}_i) \cdot \mathbf{n}_i < 0 \end{cases} \tag{4.13}$$

onde:

- N_{cp} é o conjunto dos *clip partners* de um splat S;
- $\mathbf{C}_i$ e $\mathbf{n}_i$ são o centro e o vetor normal do *clip partner* S_i, respectivamente;
- $\mathbf{M}$ é a média dos centros dos *clip partners*.

Apenas a classificação dos *clip partners* em região côncava ou convexa não é suficiente para escolher entre união e interseção das áreas de recorte. Precisam-se verificar casos onde S forma regiões côncavas ou convexas com os elementos de N_{cp}. A mesma classificação descrita na Equação 4.13 pode ser utilizada nesse caso. Dessa forma:

$$\begin{cases} S \cup N_{cp} \text{ é côncavo} & \Leftrightarrow & \forall S_i \in N_{cp}; (\mathbf{C}_i - \mathbf{C}) \cdot \mathbf{n} > 0 \\ S \cup N_{cp} \text{ é convexo} & \Leftrightarrow & \forall S_i \in N_{cp}; (\mathbf{C}_i - \mathbf{C}) \cdot \mathbf{n} < 0 \end{cases} \tag{4.14}$$

onde, $\mathbf{C}$ e $\mathbf{n}$ são o centro e o vetor normal do splat S.

Cruzando as informações obtidas nas classificações descritas nas equações 4.13 e 4.14, a escolha entre união e interseção dos recortes pode ser feita de forma correta. As escolhas são feitas para cada combinação de casos como descrito a seguir:

$$\begin{cases} N_{cp} \text{ é côncavo} \wedge S \cup N_{cp} \text{ é côncavo} & \Rightarrow & \text{União dos recortes (figuras 4.5a e 4.5b)} \\ N_{cp} \text{ é côncavo} \wedge S \cup N_{cp} \text{ é convexo} & \Rightarrow & \text{Interseção dos recortes (figuras 4.5c e 4.5d)} \\ N_{cp} \text{ é convexo} \wedge S \cup N_{cp} \text{ é côncavo} & \Rightarrow & \text{Interseção dos recortes (figuras 4.5e e 4.5f)} \\ N_{cp} \text{ é convexo} \wedge S \cup N_{cp} \text{ é convexo} & \Rightarrow & \text{União dos recortes (figuras 4.5g e 4.5h)} \end{cases} \tag{4.15}$$

4.4 Rasterização de Splats Recortados

Todas as etapas mencionadas anteriormente são realizadas *offline*, ou seja, durante um pré-processamento. Entretanto, o recorte do splat é realizado de fato durante sua rasterização. O primeiro passo na renderização de um splat é sua rasterização, isto é, a determinação dos pixels da imagem que são atingidos pelo splat projetado.

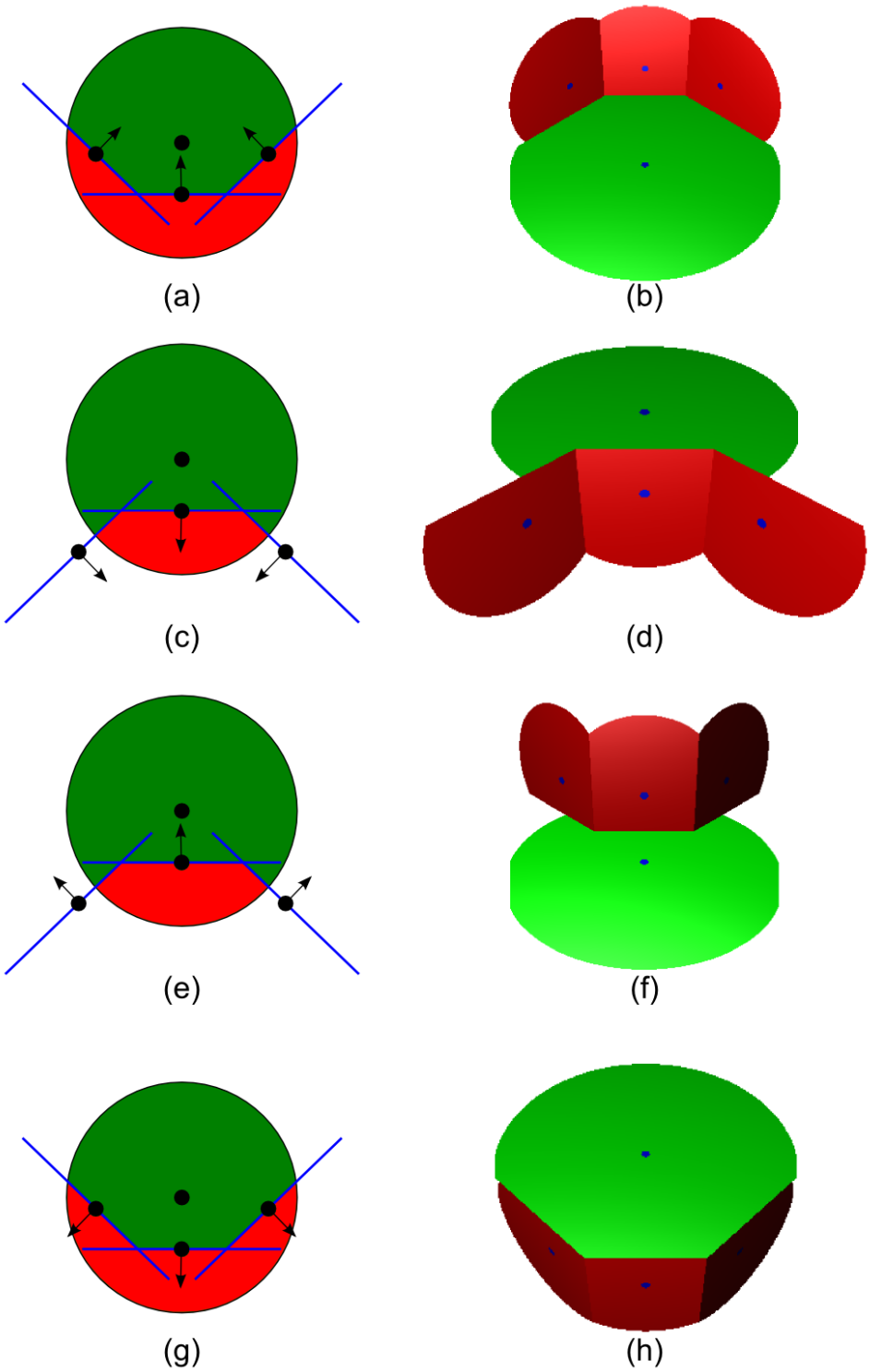

Figura 4.5: Escolha entre união ou interseção dos recortes segundo as classificações de concavidade e convexidade entre *clip partners* e entre superfícies.

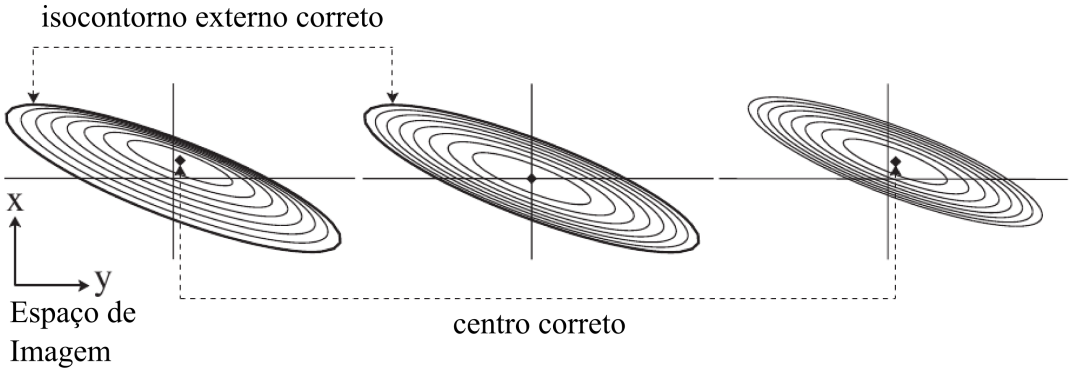

Figura 4.6: Esquerda: isocontornos de uma gaussiana mapeada respeitando a perspectiva. Centro: Aproximação afim de Zwicker *et al*. [Zwicker et al. 2004]; o isocontorno mais externo está correto, mas há erros de perspectiva no interior. Direita: Aproximação afim proposta no *EWA Splatting* original [Zwicker et al. 2001]; a projeção do centro está correta, mas todos os outros isocontornos apresentam erro de perspectiva.

As aproximações afins para a projeção de splats utilizado no *EWA Splatting* original [Zwicker et al. 2001] transformam corretamente o centro de um splat, mas não seu contorno mais externo, o que pode causar pequenos buracos na imagem renderizada (Figura 4.6 direita). A primeira abordagem a tratar essas imprecisões foi o *splatting* de projeção precisa proposto por Zwicker *et al*. [Zwicker et al. 2004]. Eles usaram uma nova aproximação afim que transforma corretamente o contorno externo do splat, mas possui erros de projeção no seu interior (Figura 4.6 meio).

Uma técnica de rasterização mais eficiente e que não apresenta erros de perspectiva foi apresentada por Botsch *et al*. [Botsch et al. 2004]. A ideia principal dessa abordagem é determinar o ponto 3D correspondente ao ponto 2D do pixel baseado em um *ray casting* local. O tamanho exato do splat projetado é complicado de ser calculado, mas uma aproximação eficiente é utilizar o raio do splat r e o valor de profundidade do centro do splat $\mathbf{C} = (c_x, c_y, c_z)$ em coordenadas de câmera:

$$s = 2r \cdot \frac{n}{c_z} \cdot \frac{h}{t-b} \tag{4.16}$$

onde, n, t e b são os parâmetros *near*, *top* e *botton* do *frustum* do observador e h denota a altura (em pixels) da *viewport*. Caso o splat seja elíptico, o valor do maior eixo é utilizado como r. Assim, é realizado um *ray casting* local entre todos os pixels de um quadrado de tamanho $s \times s$ centrado no pixel atingido pela projeção do centro do splat (Figura 4.7 esquerda). Considerando-se $\mathbf{P}$ o ponto de interseção de um raio com o plano do splat, caso o splat seja circular, se a distância de $\mathbf{P}$ ao centro do splat $\mathbf{C}$ for maior do que r, então sua contribuição é descartada; caso o splat seja elíptico, com os vetores unitários $\mathbf{u}$ e $\mathbf{v}$ na direção dos dois eixos principais, se as coordenadas (u, v) do ponto $\mathbf{P}$ forem tais que $u^2 + v^2 \leq 1$, o pixel é renderizado,

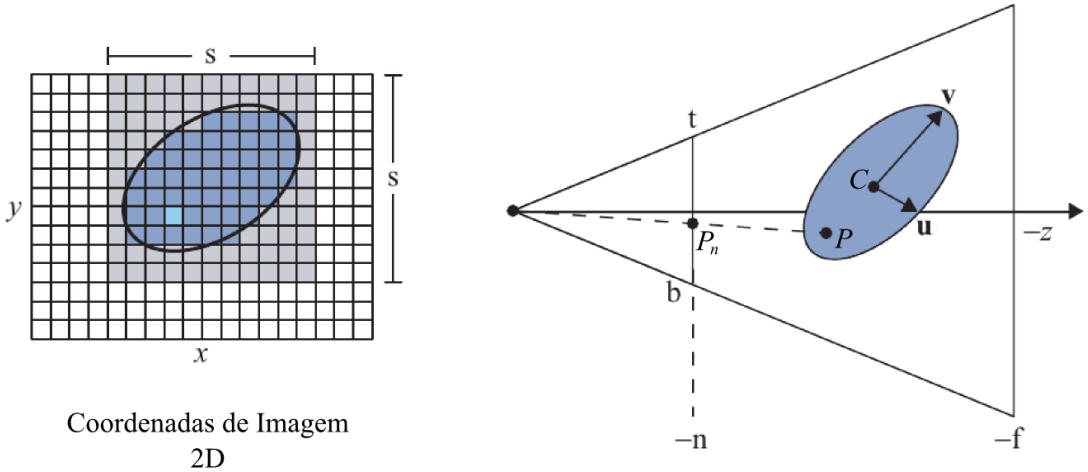

Figura 4.7: A esquerda, rasterização do splat e, a direita, *ray casting*.

caso $u^2 + v^2 > 1$, o pixel é descartado (Figura 4.7 direita).

Para splats que apresentam *clip partners*, o ponto **P** é ainda testado contra os planos de recorte. Caso o splat S "veja" os seus *clip partners*, ou seja, $S \cup N_{cp}$ é côncavo, então os vetores normais aos planos de corte estão voltados para o centro de S (figuras 4.5a e 4.5e) e a área de recorte consiste dos pontos não "vistos" pelos planos. Caso $S \cup N_{cp}$ seja convexo (figuras 4.5c e 4.5g), ocorre o contrário, os vetores normais dos planos de corte não estão voltados para o centro do splat S e a área de recorte são os pontos "vistos" pelos planos. Em suma:

$$\begin{cases} S \cup N_{cp} \text{ é côncavo} & \Rightarrow \quad \mathbf{P} \text{ é recortado por } S_j \Leftrightarrow (\mathbf{P} - \mathbf{C}_j) \cdot \mathbf{n}_j < 0 \\ S \cup N_{cp} \text{ é convexo} & \Rightarrow \quad \mathbf{P} \text{ é recortado por } S_j \Leftrightarrow (\mathbf{P} - \mathbf{C}_j) \cdot \mathbf{n}_j > 0 \end{cases} \tag{4.17}$$

onde $S_j \in N_{cp}$.

Agora que foi definida a orientação das áreas de recorte de cada *clip partner* de acordo com a concavidade ou convexidade da aresta, resta realizar a união ou a interseção das áreas de recorte, dependendo da classificação definida na Equação 4.15. Nos casos de união das áreas de recorte, o ponto é recortado se estiver na área de recorte de pelo menos um dos *clip partners*. Nos casos de interseção, o ponto deve estar na área de recorte de todos os *clip partners* para ser descartado. A classificação descrita na Equação 4.18 adapta a classificação apresentada na Equação 4.15 para decidir quando um pixel deve ser descartado da rasterização de um splat presente numa descontinuidade:

$$\begin{cases} N_{cp} \text{ é côncavo } \wedge\ S \cup N_{cp} \text{ é côncavo} & \Rightarrow \quad \mathbf{P} \text{ é recortado } \Leftrightarrow \exists S_j \in N_{cp}; (\mathbf{P} - \mathbf{C}_j) \cdot \mathbf{n}_j < 0 \\ N_{cp} \text{ é côncavo } \wedge\ S \cup N_{cp} \text{ é convexo} & \Rightarrow \quad \mathbf{P} \text{ é recortado } \Leftrightarrow \forall S_j \in N_{cp}; (\mathbf{P} - \mathbf{C}_j) \cdot \mathbf{n}_j > 0 \\ N_{cp} \text{ é convexo } \wedge\ S \cup N_{cp} \text{ é côncavo} & \Rightarrow \quad \mathbf{P} \text{ é recortado } \Leftrightarrow \forall S_j \in N_{cp}; (\mathbf{P} - \mathbf{C}_j) \cdot \mathbf{n}_j < 0 \\ N_{cp} \text{ é convexo } \wedge\ S \cup N_{cp} \text{ é convexo} & \Rightarrow \quad \mathbf{P} \text{ é recortado } \Leftrightarrow \exists S_j \in N_{cp}; (\mathbf{P} - \mathbf{C}_j) \cdot \mathbf{n}_j > 0 \end{cases} \tag{4.18}$$

4.5 Considerações Finais

O método proposto neste capítulo para o recorte de splats, a fim de adaptá-los às descontinuidades do modelo, supõe que as orientações dos *clip partners* mantenham-se consistentes, ou seja, todos os *clip partners* formem áreas côncavas ou convexas, não havendo inflexões. Esse tipo de restrição não é muito forte, pois os modelos gerados por scanners 3D geram dezenas a centenas de milhares de pontos e mesmo para modelos gerados por modelagem, áreas em torno de descontinuidades devem possuir uma maior quantidade de elementos para uma boa representação, tornando esses casos geralmente difíceis de ocorrer. Entretanto, quanto menor a quantidade de splats, maior é a possibilidade de ocorrerem casos mais complexos, como a existência de áreas côncavas e convexas no mesmo conjunto de *clip partners*. Uma solução ideal seria construir uma polilinha que conectasse todos os segmentos de interseção, adaptando, de forma ótima, o recorte. Entretanto, o custo excessivo desse cálculo para casos tão raros em modelos extraídos de scanners 3D ou mesmo gerados por modelagem tornam esse tipo de abordagem não atraente.

5 *Reconstrução da Superfície*

O algoritmo de *surface splatting* é dividido em dois passos. No primeiro passo, em ordem do objeto, cada splat é transformado para o espaço de tela e rasterizado. Os fragmentos rasterizados de um splat são armazenados em um *A-Buffer* [Carpenter 1984]. Cada fragmento armazena cor, profundidade e peso e, possivelmente, outros atributos tais como, transparência. No segundo passo, em ordem de imagem, para cada pixel, os fragmentos são ordenados de forma ascendente, segundo sua profundidade. Então, para cada superfície, encontram-se os fragmentos que a formam, e combinam-se esses fragmentos por meio de uma média ponderada por seus pesos.

A argumentação construída ao longo deste capítulo inicia com uma breve introdução ao problema de reconstrução de superfícies, seguida de uma descrição do algoritmo do *A-Buffer* e de uma apresentação e análise das soluções clássicas para o problema de reconstrução de superfícies. Após essa exposição, amostra-se a utilização da segmentação de um modelo para facilitar o processo de reconstrução da superfície adaptando as soluções clássicas.

5.1 O Problema da Reconstrução de Superfícies

Cada splat junto com seus vizinhos formam uma superfície, e para cada pixel há um conjunto de fragmentos de splats. Para reconstruir uma superfície, é necessário encontrar os splats que formam a superfície e, então, somar e normalizar suas contribuições. Como não há informações explícitas de quais fragmentos pertencem a quais superfícies, ou mesmo de quantas superfícies existem ao todo, o uso de heurísticas é inevitável. Essas heurísticas são definidas com base em informações implícitas que podem estar disponíveis ou que, simplesmente, podem ser assumidas a partir dos dados dos dados fornecidos. Algumas dessas informações são:

- As superfícies tendem a estar separadas por uma distância maior do que os tamanhos dos splats que as formam. Assim, os fragmentos de uma superfície tendem a estar agrupados sobre o eixo de profundidade.
- Dois splats se sobrepõem se suas esferas limitantes ou os intervalos de suas profundidades

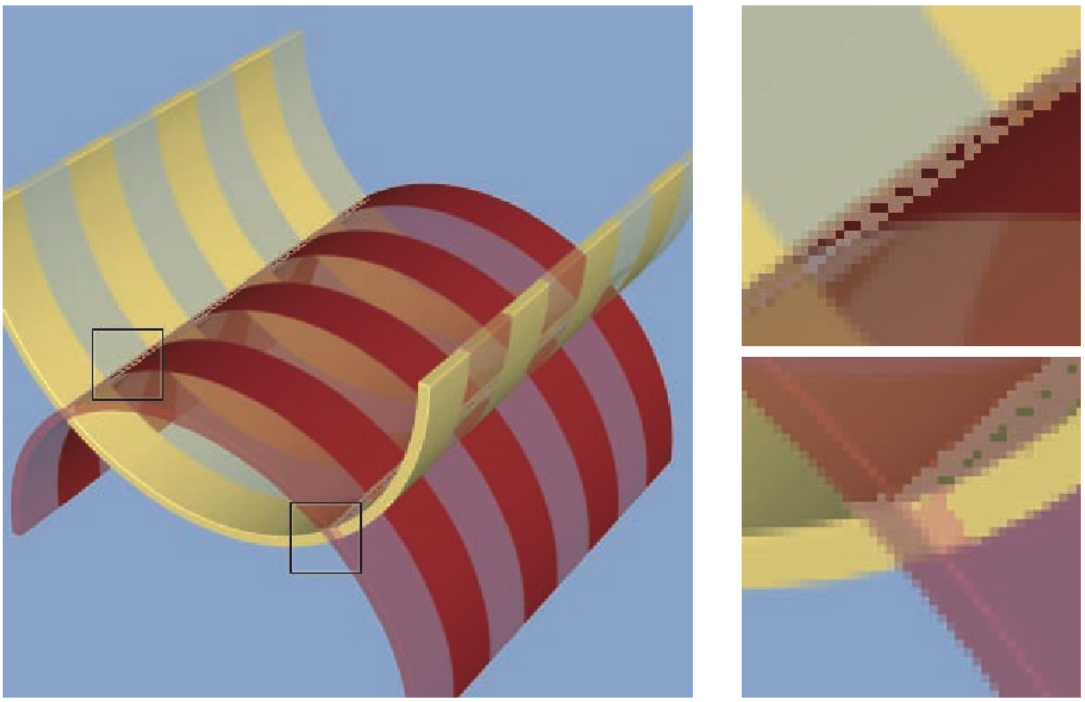

Figura 5.1: À esquerda, superfícies semi-transparentes intersectando-se. À direita, detalhes dos artefatos gerados nas linhas de interseção dos objetos.

se interceptam.

- Cada fragmento pertence a apenas uma superfície. Especificamente, o fragmento mais próximo pertence à superfície mais próxima, e o fragmento mais distante pertence à superfície mais distante. Entretanto, não é possível saber se essas superfícies são a mesma.
- A amostragem de objetos produz superfícies que consistem de aproximadamente o mesmo número de fragmentos em cada pixel.

Entretanto, problemas ocorrem em superfícies intersectantes, pois elas não podem ser identificadas sem informações explícitas. Nas áreas de interseção, as superfícies estão mais próximas umas das outras do que os splats que as formam, tornando difícil a identificação da superfície à qual um dado fragmento pertence (Figura 5.1). Essa situação ocorre também próximo a arestas e cantos. Mesmo após o recorte remover fragmentos não pertencentes a uma superfície, os fragmentos de um lado da descontinuidade estão próximos de outros fragmentos do outro lado. Combinar esses fragmentos, que geralmente apresentam sombreamentos bastante distintos, pode levar a borramento.

5.2 *A-Buffer*

Antes de apresentar as técnicas de reconstrução de superfície, para fins didáticos, será descrito como os dados para cada pixel são armazenados e compostos.

Um *A-Buffer* [Carpenter 1984] é um *frame buffer* formado por um vetor de pixels, cada um possuindo uma lista de fragmentos, desenvolvido para evitar *aliasing* e remover superfícies ocultas. Em renderização de polígonos, cada fragmento é uma amostra de um polígono na posição do pixel. Em *splatting*, cada fragmento é uma amostra de um splat na posição do pixel. Entretanto, a diferença para os polígonos é que o fragmento já é uma amostra da superfície que ele representa, enquanto para *splatting*, cada superfície é formada pela combinação de vários fragmentos, pois vários splats se sobrepõem para formar a superfície.

Assumindo que as superfícies já foram encontradas e que cada superfície possui sua cor, transparência e profundidade, então o algoritmo de *A-Buffer* funciona da seguinte forma:

1. Ordenar as superfícies de forma ascendente segundo seus valores de profundidade.
2. Inicializar a cor e a transparência do pixel atual: $c_0 = (0,0,0), a_0 = 0$.
3. Partindo da superfície com menor profundidade, adiciona-se a contribuição de cada superfície à cor atual do pixel usando a equação

$$c_{k+1} = c_k + c_s(1 - a_k) \qquad a_{k+1} = a_k + a_s(1 - a_k) \tag{5.1}$$

   onde c_s e a_s são a cor e a transparência de cada superfície, respectivamente, e c_k e a_k são a cor e a transparência atuais do pixel.
4. Se, em qualquer estágio, o algoritmo atingir $a_k = 1$, isto é, o pixel está completamente opaco, então o restante das superfícies não serão visíveis.
5. Se $a_k < 1$ após todas as superfícies terem sido combinadas, então a cor de fundo é combinada com a cor do pixel.

O problema com *A-Buffer* é que ele provê alocação dinâmica de novos fragmentos de acordo com a demanda. Dessa forma, a memória requerida nesse tipo de técnica é virtualmente ilimitada. Esse é um sério problema para implementações em hardware. Implementações do algoritmo *A-Buffer* utilizando uma quantidade limitada de memória foram propostas por Winner *et al* [Winner et al. 1997] e Jouppi e Chang [Jouppi e Chang 1999].

5.3 Métodos de Reconstrução Clássicos

Os dois métodos de reconstrução de superfície mais conhecidos são *Z-Threshold* de Zwicker et al. [Zwicker et al. 2001] e *Z-Range* de Räsänen [Räsänen 2002]. Ambos os métodos utilizam-se da proximidade dos fragmentos para inferir se eles pertencem à mesma superfície. A seguir, uma descrição mais detalhada dos métodos é feita e uma análise mostra as vantagens e desvantagens de cada métodos.

5.3.1 *Z-Threshold*

Zwicker *et al*. [Zwicker et al. 2001] apresentaram um algoritmo de reconstrução de superfície baseado no tradicional algoritmo *z-buffer*. Utilizando-se da hipótese de que os fragmentos de uma superfície tendem a estar agrupados sobre o eixo de profundidade, dois fragmentos são considerados como pertencentes à mesma superfície se estão mais próximos do que uma constante especificada pelo usuário.

O algoritmo começa ordenando os fragmentos de um pixel em ordem ascendente de profundidade. Se f for o primeiro fragmento dessa lista, a primeira superfície S acumula os valores desse fragmento. Se o próximo fragmento f' estiver mais próximo de f do que uma constante ε seus valores são combinados ao da superfície atual S e repete-se o processo utilizando agora o fragmento f' como referência para o próximo da lista. Quando o próximo fragmento estiver mais distante do fragmento anterior do que a constante ε, o laço é parado e a superfície S é concluída, ou seja, sua cor é normalizada segundo o peso acumulado e a cor da superfície é combinada com a cor atual do pixel. Por fim, o processo inteiro é repetido com os fragmentos restantes para a detecção da próxima superfície.

Como a soma dos pesos das contribuições dos fragmentos não necessariamente é igual a um, a normalização das contribuições é necessária. A cor da superfície c_s é calculada usando as cores e pesos dos fragmentos como:

$$c_s = \sum_k \frac{c_k w_k}{\sum_j w_j} \tag{5.2}$$

onde os somatórios são realizados sobre todos os fragmentos de uma mesma superfície. Os valores de transparência ou qualquer outro atributo são calculados de maneira similar.

A Figura 5.2 mostra o algoritmo *Z-Threshold* aplicado ao mesmo caso, utilizando três valores diferentes para ε. Os fragmentos f_1, f_2, f_3 e f_4 são aqueles associados ao pixel ilustrado, ordenados de forma ascendente. Observa-se na figura que os fragmentos f_1 e f_2 pertençam

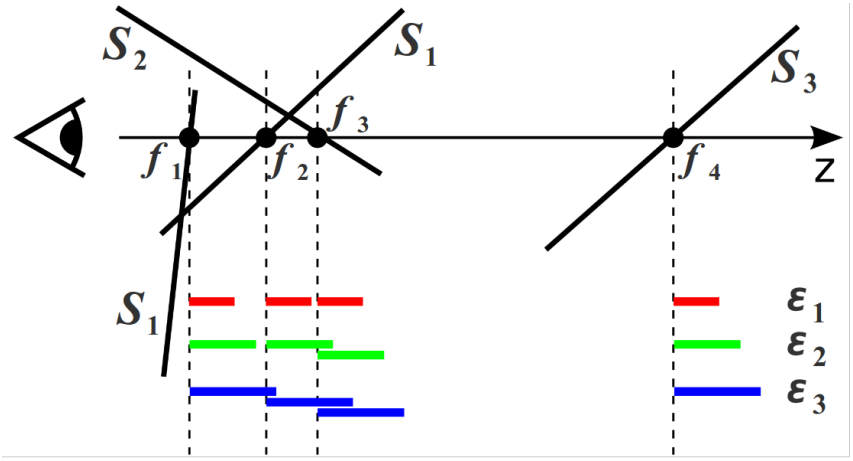

Figura 5.2: Algoritmo *Z-Threshold* [Zwicker et al. 2001]. (S_i: superfícies; f_i: fragmentos; ε_i: valores de limiar)

à mesma superfície, S_1; o fragmento f_3 pertence à superfície S_2, e o fragmento f_4 pertence à superfície S_3. A figura mostra a dificuldade na escolha do limiar, pois nenhum dos valores de ε consegue realizar a renderização correta. Para ε_1, nenhum dos fragmentos são combinados, quando, na verdade, os fragmentos f_1 e f_2 deveriam ser combinados. Para ε_2, os fragmentos f_2 e f_3 são combinados, quando não deveriam, pois pertencem a superfícies diferentes. Para ε_3, todos os três primeiros fragmentos são combinados, quando apenas f_1 e f_2 deveriam ser combinados. Combinar esses fragmentos gera artefatos como ilustrados na Figura 5.1.

Os benefícios desse algoritmo são sua simplicidade e eficiência. Em casos mais gerais, que satisfazem a hipótese de que as superfícies estão mais distantes entre si do que o tamanho de um splat, o algoritmo funciona corretamente (como o caso do fragmento f_4, corretamente separado numa superfície distinta das outras). Entretanto, na interseção de superfícies, onde a hipótese não é válida, não há escolha de limiar ótima. Além disso, a escolha não trivial de um parâmetro constante para a imagem inteira significa que os objetos devem ter splats com aproximadamente o mesmo tamanho, o que é uma restrição severa à usabilidade do *splatting*.

5.3.2 *Z-Range*

Outra solução para o problema da reconstrução de superfícies baseando-se na noção de que superfícies tendem a ser separadas por uma distância maior do que as amostras de uma mesma superfície é o algoritmo do *Z-Range* proposta por Räsänen [Räsänen 2002]. O método de *Z-Range* armazena em cada fragmento o intervalo $[z_{min}, z_{max}]$ de todas as profundidades do splat que o originou. A etapa de reconstrução de superfície combina todos os fragmentos que possuem splats com intervalos de profundidade sobrepostos. Esse algoritmo não necessita de parâmetros especificados pelo usuário e adapta-se à amostragem do modelo.

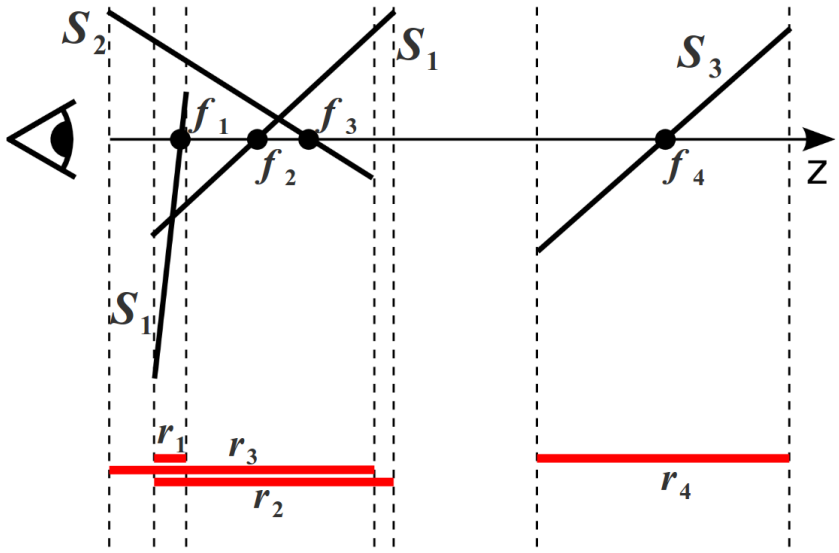

Figura 5.3: Algoritmo *Z-Range* [Räsänen 2002]. (f_i: fragmentos; r_i: intervalos dos splats que formaram os fragmentos)

A Figura 5.3 mostra o resultado do algoritmo *Z-Range* para o mesmo caso ilustrado na Figura 5.2. Como o método independe de um limiar do usuário, há apenas uma solução possível. Nesse caso, os três primeiros fragmentos são combinados, pois seus intervalos de profundidade possuem alguma interseção, acarretando, mais uma vez, em mistura de superfícies.

Esse algoritmo mantém a simplicidade e eficiência do algoritmo *Z-Threshold*, além da ausência do parâmetro global escolhido pelo usuário. Entretanto, o problema de superfícies intersectantes continua sem solução e aparecem artefatos nas silhuetas, pois os splats localizados nessas áreas apresentam intervalos de profundidade maiores e muitos fragmentos são combinados.

5.4 Segmentação do Modelo Aplicada à Reconstrução de Superfícies

Uma característica comum a ambos os métodos clássicos descritos na Seção 5.3 é a hipótese utilizada como justificativa de suas abordagens. A distância entre superfícies, geralmente, é maior do que o tamanho dos splats que as formam, mas essa hipótese é inválida em áreas de interseção de superfícies e áreas próximas a descontinuidades, Assim, utilizar apenas a distância entre os fragmentos não resolve o problema nesses casos. Identificar as superfícies previamente, através de informações de modelagem ou através da identificação manual poderia solucionar o problema, mas limitaria a aplicabilidade do *splatting*.

A solução proposta é adicionar informações da vizinhança dos splats para a decisão de com-

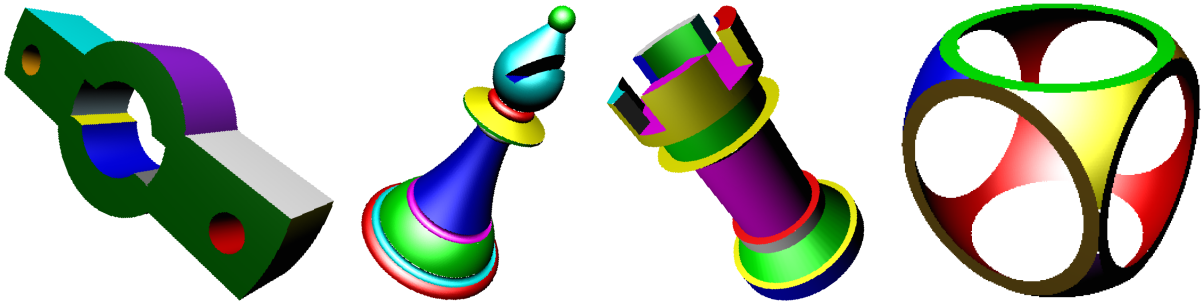

Figura 5.4: Segmentação de modelos utilizando buscas em largura no grafo de vizinhança.

binar os fragmentos. As *sharp-edges* do grafo de vizinhança ligam não só os splats presentes em descontinuidades, mas também os splats localizados na linha de interseção de superfícies que se cruzam. Os artefatos gerados nas áreas de interseção de superfícies acontecem por causa da combinação de fragmentos pertencentes a splats cujos vetores normais são muito diferentes. Uma solução seria adicionar, ao fragmento, o identificador do splat de onde foi gerado e uma lista dos identificadores de todos os splats conectados ao primeiro através de uma *sharp-edge*. Dessa forma, os algoritmos anteriores poderiam adicionar a seguinte condição para o próximo fragmento ser adicionado ao antecessor: o próximo fragmento é combinado apenas se não tiver seu identificador presente na lista do fragmento antecessor. Essa solução apresenta o mesmo problema do *A-Buffer*, pois também utiliza uma quantidade de memória variante e ilimitada. Isso não é problema para uma implementação em software, mas torna a implementação em hardware um desafio.

Uma saída, é utilizar a segmentação do modelo. Se o modelo apresenta apenas arestas fechadas, as *sharp-edges* particionam o grafo de vizinhança em diversas áreas sem descontinuidade. A abordagem é semelhante à utilizada por Daniels *et al.* [Daniels et al. 2008] apresentada na Seção 2.4. A segmentação tem início pela escolha aleatória de um splat, ao qual aplica-se um identificador único antes de adicioná-lo a uma fila W. O método entra em um laço, onde S, o primeiro elemento da fila, é removido, enquanto todos os seus vizinhos, que estiverem conectados através de uma *continuous-edge* e que ainda estiverem sem identificador, recebem o mesmo identificador de S e são inseridos na fila W. O laço termina quando a fila W ficar vazia. Se ainda houver splat sem identificador, o processo é repetido escolhendo um splat ainda não visitado até que todos os splats do modelo apresentem algum identificador. A Figura 5.4 mostra alguns modelos segmentados utilizando a técnica proposta.

5.5 Adaptação dos Métodos Clássicos

Da mesma forma como a construção do grafo de vizinhança, a identificação das *sharp-edges* e das *continous-edges* e a classificação dos *clip partners*, a segmentação do modelo é feita em um pré-processamento. Após essa etapa, cada splat carrega um identificador que indica a superfície à qual o splat pertence. Os fragmentos gerados durante a rasterização de um splat recebem seu identificador. Isso impede que fragmentos com identificadores distintos sejam combinados.

Um ideia ingênua e errônea, é identificar o fragmento com menor profundidade e combiná-lo com todos os fragmentos do mesmo pixel que apresentem o mesmo identificador. Apesar desses fragmentos pertencerem a uma mesma superfície, essa abordagem pode combinar fragmentos que estejam muito distantes entre si. Por exemplo, uma esfera apresenta dois fragmentos distantes em cada pixel onde é renderizada e esses fragmentos não podem ser combinados.

Os métodos clássicos são simples e eficientes, entretanto, apresentam dificuldades de reconstruir corretamente superfícies na proximidade de interseções de superfícies. Uma boa forma de reconstruir a superfície é adaptar esses métodos, adicionando a informação do identificador de superfícies presente nos fragmentos. Ambos os métodos ordenam os fragmentos de um pixel segundo sua profundidade e realizam uma varredura sequencial combinando-os em sequência. Porém, é possível que fragmentos de superfícies diferentes estejam intercalados. Dessa forma, o algoritmo de reconstrução adaptado funciona como segue:

1. Encontrar f_1, o fragmento de menor profundidade;
2. Remover temporariamente todos os fragmentos cujos identificadores são diferentes do identificador do fragmento f_1;
3. Aplicar qualquer método de reconstrução para os fragmentos restantes;
4. Combinar os fragmentos identificados como pertencentes à mesma superfície de f_1 pelo método utilizado;
5. Adicionar novamente os fragmentos removidos no passo 2;
6. Se ainda houver fragmentos, voltar para o passo 1.

A Figura 5.5 ilustra o funcionamento do novo método em combinação com os métodos de *Z-Threshold* e *Z-Range* para o mesmo caso das figuras 5.2 e 5.3. A Figura 5.5a mostra a adaptação do método de *Z-Threshold* utilizando o mesmo limiar ε_3 da Figura 5.2. O primeiro fragmento f_1 determina o identificador da superfície S_1. Pelo método de Zwicker, os fragmentos

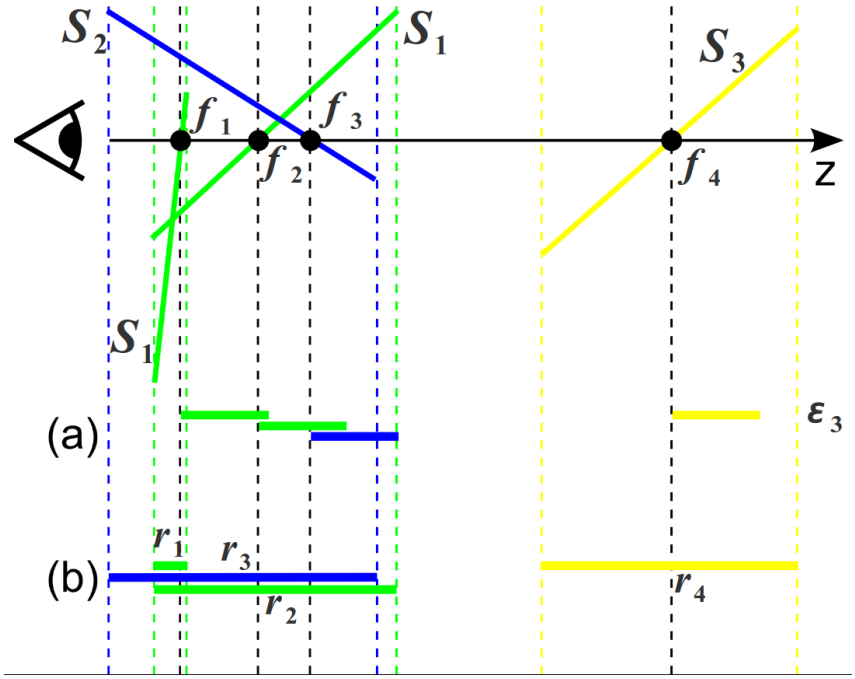

Figura 5.5: Adaptação dos algoritmos clássicos. (a) *Z-Threshold*. (b) *Z-Range*.

f_2 e f_3 devem ser combinados, porém f_3 possui um identificador diferente de S_1 e é removido do cálculo no passo 2 do algoritmo proposto. Assim, apenas os fragmentos f_1 e f_2 são combinados produzindo uma renderização correta. O mesmo acontece com o método de *Z-Range*, ilustrado na Figura 5.5b.

A escolha de um limiar ε para o método *Z-Threshold* continua sendo um problema delicado. Um valor muito pequeno pode separar fragmentos de uma mesma superfície e renderizá-los como pertencentes a superfícies distintas. Entretanto, essa escolha torna-se mais fácil, visto que depende apenas da curvatura da superfície e do tamanho dos splats além de corrigir um caso complicado como a interseção de superfícies. A Figura 5.6 ilustra a diferença entre o método de *Z-Threshold* original e o método adaptado. Na Figura 5.6a, os pixels próximos à linha de interseção das esferas apresentam cor magenta, pois fragmentos de splats da esfera vermelha foram combinados com os fragmentos de splats da esfera azul devido à sua proximidade. Na Figura 5.6b, esses fragmentos não são combinados devido a seus identificadores, mesmo utilizando o mesmo valor de limiar.

Apesar do método proposto funcionar bem para a maioria dos casos, ele está limitado às restrições dos métodos de segmentação de modelos. Casos onde arestas não são fechadas (Figura 5.7a) e onde superfícies se auto interceptam (Figura 5.7b) permitem que a busca em largura encontre caminhos de *continuous-edges* no grafo de vizinhança conectando splats que apresentem *sharp-edges* entre si, atribuindo-lhes o mesmo identificador de superfície. Apesar desses casos apresentarem erros devidos à segmentação, as *sharp-edges* continuam identifi-

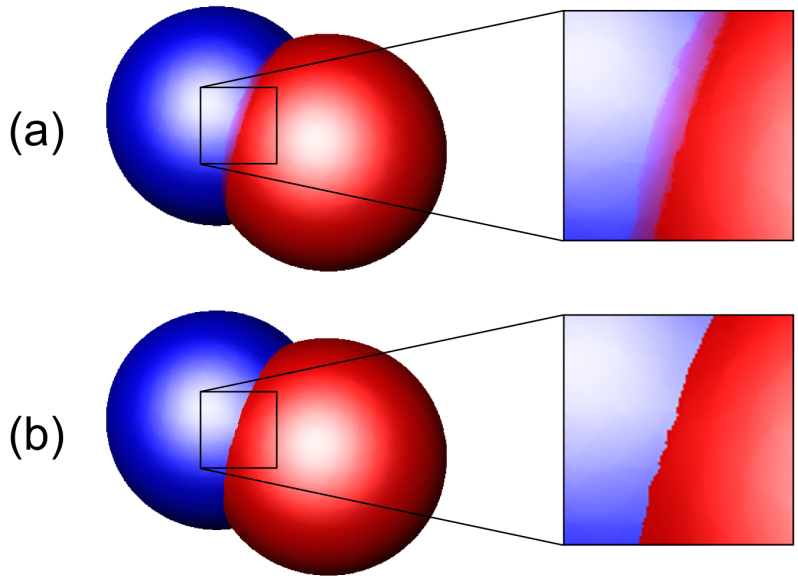

Figura 5.6: Duas esferas intersectantes renderizadas utilizando o algoritmo *Z-Threshold* clássico (a) e o algoritmo *Z-Threshold* adaptado (b) utilizando o mesmo limiar.

Figura 5.7: Casos complexos. (a)Arestas não fechadas. (b)Superfícies auto-intersectantes.

cando splats que não devem ser combinados em espaço de tela.

5.6 Considerações Finais

A etapa de reconstrução de superfícies utiliza várias heurísticas para determinar quais fragmentos combinar e assim reamostrar corretamente a superfície em espaço de tela. Essas heurísticas são baseadas nas possíveis características do modelo. Dependendo dessas características, uma abordagem pode ser mais eficiente do que outra. Além disso, informações adicionais sobre o modelo ajudam a determinar as características da superfície que foi projetada sobre determinado pixel. Essas informações podem ser sobre: a vizinhança local, a estimativa de curvatura, o raio do splat, a orientação da normal ao splat, a modelagem, a forma de aquisição, etc. Alguns desses dados podem ser obtidos automaticamente ou por meio do usuário, como o aplicativo *PointShop 3D* [Zwicker et al. 2002] , que realiza a identificação de superfícies através do usuário e informações de modelagem para melhor reconstruir a superfície durante a renderização do modelo.

6 *Análise de Resultados*

O objetivo deste trabalho é renderizar eficientemente e corretamente arestas, cantos e interseções de superfícies usando splats como primitivas de modelagem e renderização. A geração de splats, normalmente, é a adição de normais e raios ou eixos às amostras de modelos baseados em pontos. Os vetores normais podem ser estimados por diversos métodos, desde que mantenham orientação consistente e respeitem descontinuidades, nas as suavizando [Li et al. 2010]. Com relação às dimensões dos splats, diversas heurísticas, dependendo da amostragem do modelo, podem ser utilizadas. Neste capítulo, todos os modelos utilizados nos testes (Figura 6.1) são formados por splats circulares, cujos raios são iguais à distância média a seus cinco vizinhos mais próximos.

O processo de renderização é dividido nas etapas de pré-renderização e de renderização. A pré-renderização consiste na geração do grafo de vizinhanças; na identificação e classificação dos *clip partners* e na segmentação. A etapa de renderização, por sua vez, consiste na projeção dos splats, no recorte de splats e na reconstrução de superfícies.

Os testes apresentados neste capítulo estão divididos em três grupos que visam comparar os métodos propostos neste trabalho para a geração do grafo de vizinhança, para classificação e adaptação dos recortes de splats e para reconstrução de superfícies através de segmentação do modelo, com métodos existentes. Todos os testes foram feitos em um computador com processador Intel Core2Duo 2.8GHz no sistema operacional Ubuntu 10.10.

6.1 Geração do Grafo de Vizinhança

O grafo de vizinhança é essencial para o bom desempenho das fases subsequentes do processo de renderização dos splats. A análise feita na Seção 3.4 mostrou as principais vantagens da estimativa de esferas limitantes sobre a estimativa *k-nearest*: a independência de um valor escolhido pelo usuário para estimar a vizinhança e a detecção de splats nas proximidades de descontinuidades, mesmo apresentando densidades diferentes em torno da linha da aresta.

Figura 6.1: Modelos utilizados nos testes. Da esquerda para a direita: (em cima) bispo, cadeira, peça, torre; (em baixo) sinuca, smiling face.

Tabela 6.1: *Comparação de tempos de geração de grafos de vizinhança*

Modelo	**Nº de pontos**	**Nº de arestas**		**Tempo de geração (segundos)**	
		k-Nearest	**Esferas Limitantes**	**k-Nearest**	**Esferas Limitantes**
Bispo	15970	111790 ($k=7$)	112954 ($m \approx 14,15$)	47,81	11,56
Cadeira	41737	292159 ($k=7$)	275879 ($m \approx 13,22$)	327,25	76,41
Peça	7478	52346 ($k=7$)	51208 ($m \approx 13,70$)	10,43	2,48
Sinuca	24785	173495 ($k=7$)	167362 ($m \approx 13,51$)	115,37	27,18
Smiling Face	4305	34440 ($k=8$)	32434 ($m \approx 15,07$)	3,63	0,84
Torre	10638	74466 ($k=7$)	72938 ($m \approx 13,71$)	20,71	5,02

A Tabela 6.1 compara as diferentes tempos de geração do grafo de vizinhança sem a utilização de qualquer estrutura de particionamento espacial. Para que a comparação entre o método dos *k-nearest* e o método das esferas limitantes seja justa, o valor de k utilizado é aproximadamente igual a metade do número médio, m, de arestas incidentes em uma amostra, no grafo de vizinhança obtido com o método de esferas limitantes.

O tempo de geração do grafo de vizinhança usando a estimativa de esferas limitantes é menor devido, principalmente, à simetria das arestas do grafo. Essa simetria reduz o número de comparações entre as amostras pela metade, e, assim, o número, NC_{el}, de comparações feitas durante a geração do grafo de vizinhança é

$$NC_{el} \approx \frac{n^2}{2}. \tag{6.1}$$

Por outro lado, a estimativa *k-nearest* realiza n pesquisas a uma lista de k elementos para verificar se a amostra analisada encontra-se entre os k mais próximo. Mesmo adotando busca binária na lista de k elementos, cada amostra realiza $n \cdot \log k$ comparações. Assim, o número total, NC_{kn}, de comparações feitas por essa estimativa é

$$NC_{kn} \approx n^2 \cdot \log k. \tag{6.2}$$

A geração do grafo de vizinhança é um dos principais gargalos em técnicas baseadas em pontos e, consequentemente, em splats. Para acelerar o processo de busca por elementos próximos, necessários à construção da vizinhança, estruturas de particionamento espacial, como *octrees*, *k-d trees*, etc., eliminam a necessidade de comparações com todos os elementos do modelo. Vários trabalhos foram propostos para acelerar o processo de geração de grafo de vizinhança, como o grafo usando a estimativa *k-nearest* [Connor e Kumar 2010]. Assim, métodos similares podem facilitar e acelerar ainda mais o processo de geração do grafo usando a estimativa de esferas limitantes.

6.2 Recorte de Splats

A escolha da melhor opção de recorte de um splat depende da classificação dos splats chamados seus *clip partners*. Um *clip partner* precisa cruzar efetivamente o splat a ser recortado. Assim, para a correta classificação, é preciso analisar se dois splats que apresentam uma *sharp-edge* efetivamente se cruzam e devem ser recortados. A Tabela 6.2 mostra a taxa de acerto das *sharp-edges* em detectar splats que precisam ser recortados. A coluna do número de *sharp-edges*, N_{sf}, mostra a taxa dessas arestas em relação ao número total de arestas, E. Apesar da baixa

Tabela 6.2: *Taxa de* sharp-edges *que detectaram pares de* clip partners

Modelo	**Limiar**	**Nº de arestas (E)**	**Nº de *sharp-edges* (E_{sf}) (E_{sf}/E)**	**Nº de interseções reais (E_{ir}) (E_{ir}/E_{sf})**
Bispo	0,70	112954	12579 (11,14%)	4281 (34,03%)
Cadeira	0,80	275879	62739 (22,74%)	19438 (30,98%)
Peça	0,98	51208	8802 (17,19%)	2341 (26,59%)
Sinuca	0,70	167362	30878 (18,45%)	7278 (23,57%)
Smiling Face	0,90	32434	7715 (23,79%)	1808 (23,43%)
Torre	0,90	72938	13333 (18,28%)	4381 (32,85%)

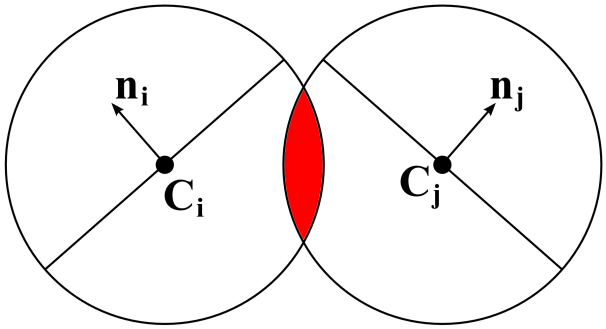

Figura 6.2: Só existe possibilidade de dois splats se cruzarem se os dois splats interceptarem o volume de interseção das esferas limitantes.

quantidade relativa de arestas detectadas, muitos dos testes de interseção resultaram no não cruzamento dos splats, ou seja, falsas detecções. Em torno de 28% das *sharp-edges*, os splats realmente se cruzavam.

Essa alta taxa de *sharp-edges* desnecessárias ocorre devido ao pequeno volume de interseção das esferas limitantes de splats localizados nas proximidades de uma descontinuidade. A Figura 6.2 ilustra o motivo do grande número de *sharp-edges* desnecessárias. Quanto menor o volume de interseção de duas esferas limitantes, menor a probabilidade de dois splats se cruzarem, porém próximo de descontinuidades, esses casos tornam-se muito comuns. Outro fator que influencia nessa taxa de acerto é o ângulo entre as superfícies na descontinuidade, como mostrado na Seção 3.4.

Tabela 6.3: *Taxa de splats detectados que necessitaram de recorte*

Modelo	**Limiar**	**Nº de splats (P)**	**Nº de candidatos (P_c/P)**	**Nº de recortados (P_r/P_c)**
Bispo	0,70	15970	5391 (33,76%)	2823 (52,36%)
Cadeira	0,80	41737	22196 (53,18%)	14833 (66,83%)
Peça	0,98	7478	3862 (51,64%)	1954 (50,59%)
Sinuca	0,70	24785	13159 (53,09%)	6132 (46,60%)
Smiling Face	0,90	4305	4120 (95,70%)	1412 (34,27%)
Torre	0,90	10638	5177 (48,66%)	2910 (56,21%)

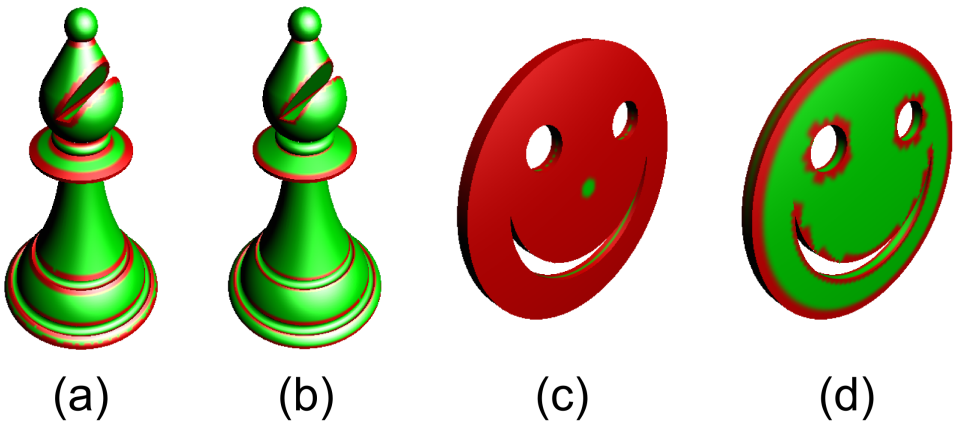

Figura 6.3: Relação entre splats detectados através das *sharp-edges* e splats recortados. (a) Bispo, onde todos os splats em vermelho possuem pelo menos uma *sharp-edge*. (b) Bispo, onde apenas os splats recortados receberam cor vermelha. (c) Smiling face renderizado da mesma forma que o bispo em (a). (d) Smiling face renderizado da mesma forma que o bispo em (b).

Apesar da, relativamente, baixa taxa de detecção das *sharp-edges*, a detecção de splats localizados nas descontinuidades apresenta maior fidelidade. Esse efeito ocorre, pois muitos dos splats presentes em descontinuidades apresentam várias arestas com splats com os quais não têm interseção, mas apresentam algumas arestas com splats com os quais precisam ser recortados. A Tabela 6.3 mostra a taxa de splats recortados em relação aos splats detectados como pertencentes a descontinuidades. Em média, aproximadamente 50% dos splats detectados necessitam de recorte. Para modelos do "bispo"e do "smiling face", a Figura 6.3 mostra a relação entre os splats detectados pelo método como pertencentes a descontinuidade e os splats realmente recortados. O exemplo da "smiling face" apresentou discrepância dos demais, pois a hipótese de que a distância entre as superfícies é maior do que o tamanho do splats não foi respeitada. Todos os splats presentes na face frontal do modelo apresentam arestas com splats da face traseira acarretando numa grande quantidade de *sharp-edges* e candidatos a recorte desnecessários.

Quanto ao tratamento do recorte, a classificação dos *clip partners* realiza o recorte adaptado à curva da descontinuidade sem deixá-la serrilhada ou apresentando buracos. A Figura 6.4 compara três renderizações do modelo Torre utilizando três estratégias para recortar splats localizados nas arestas e cantos: 1) todos os *clip partners* de um splat o recortam cada um independentemente dos demais (Figura 6.4a); 2) apenas o *clip partner* mais próximo do splat o recorta (Figura 6.4b); 3) os recortes são adaptados através da classificação dos *clip partners* (Figura 6.4c). No primeiro caso, a ambiguidade dos recortes acarreta no aparecimento de buracos ao longo de arestas curvas (buracos foram evidenciados em vermelho). No segundo caso, as arestas são serrilhadas, embora disfarçadas por causa da combinação dos splats na etapa

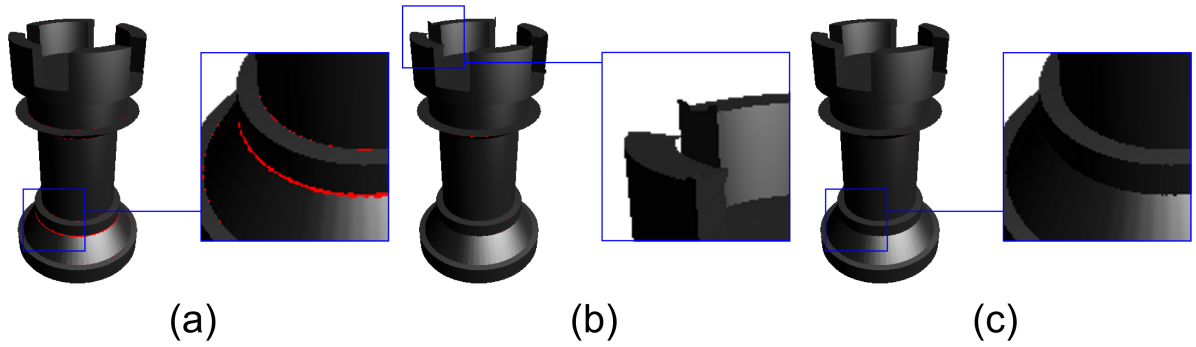

Figura 6.4: Comparação entre recortes. (a) Cada *clip partner* recorta independente dos demais. (b) Apenas o *clip partner* mais próximo recorta o splat. (c) Adaptação dos recortes através da classificação dos *clip partners*

Tabela 6.4: *Tempos de execução das etapas de renderização*

Modelo	**Tempos de Execução (segundos)**			**Nº de splats projetados**	**Nº de pixels atingidos**
	Projeção	**Recorte**	**Reconstrução**		
Bispo	5,85	0,25	0,74	8947	52983
Cadeira	9,42	0,52	0,95	20691	70163
Peça	11,10	0,46	1,28	3506	95500
Sinuca	8,02	0,40	0,98	12674	73223
Smiling Face	6,04	0,40	0,98	2094	77477
Torre	8,03	0,48	0,93	5088	67572

de reconstrução da superfície, artefatos aparecem nos cantos do modelo. No terceiro caso, os recortes adaptam-se à curva da descontinuidade, não permitindo o borramento das arestas e fazendo a adaptação correta dos splats.

Entretanto, existem casos onde há *clip partners* oriundos de superfícies diferentes, o que pode levar a erros na classificação dos recortes. Uma solução é agrupar os *clip partners* em subconjuntos de elementos de mesma superfície e, então, cada subconjunto é classificado e realiza seu recorte sobre o splat independentemente dos demais subconjuntos.

6.3 Renderização

A etapa de renderização deve buscar a maior eficiência possível a fim de alcançar bons *frame rates*. Várias formas de otimização tentam acelerar as duas principais etapas da renderização: a projeção dos splats e a reconstrução da superfície. A Tabela 6.4 mostra os tempos de execução das etapas de renderização dos modelos. Em nossa implementação, foi utilizada a técnica de projeção de splats proposta por Botsch *et al*. [Botsch et al. 2004] mostrada na Seção 4.4. Como essa etapa está fora do escopo das contribuições deste trabalho, exceto pela etapa de remoção de fragmentos recortados, a Tabela 6.4 mostra os tempos de execução da rasterização e da

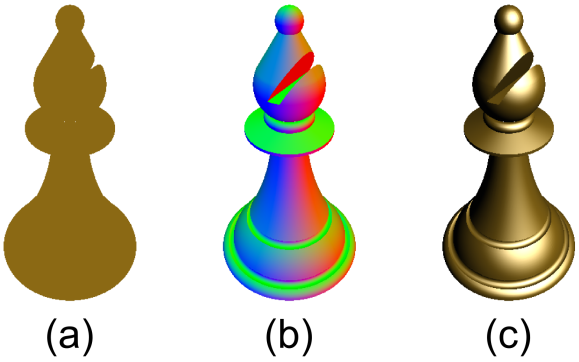

Figura 6.5: Estratégia de sombreamento. (a) Fragmentos são recortados e combinados segundo suas superfície detectadas. (b) Vetores normais são combinados de forma similar às cores. (c) Sombreamento realizado sobre os pixels atingidos usando o mapa de normais descrito em (b).

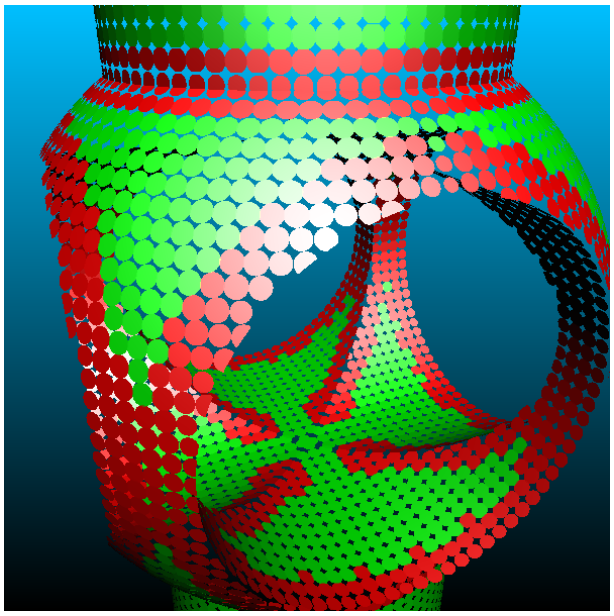

Figura 6.6: Caso complexo para o algoritmo de segmentação proposto. Splats vermelhos apresentam pelo menos uma *sharp-edge*. Splats verdes apresentam apenas *continuous-edges*.

remoção dos fragmentos separadamente. Nota-se que o tempo de renderização é influenciado principalmente pela etapa de projeção.

O sombreamento utilizado é o *deferred shading* proposto em [Botsch et al. 2004]. Nessa estratégia, o sombreamento não é feito por splat, mas por pixel, similar ao sombreamento de *Phong*. Os vetores normais dos splats são armazenados junto com os fragmentos e são combinados da mesma forma que as cores, usando a Equação 5.2. A Figura 6.5 mostra a estratégia do *deferred shading* utilizado. Essa estratégia, além de melhorar o sombreamento do modelo, mesmo com poucos splats, é facilmente implementada em hardware, pois o mapa de normais formado ao final da reconstrução da superfície pode ser transformado em uma textura e enviado a uma GPU programável para sombreamento do modelo.

A reconstrução da superfície depende da segmentação para realizar a separação de superfícies corretamente e assim combinar de maneira correta os fragmentos de cada pixel. A restrição para o bom funcionamento do algoritmo de segmentação é que o ângulo entre as superfícies intersectantes seja maior do que o limiar definido para classificar as *sharp-edges*. Mesmo que essa estratégia simples e eficiente realize a segmentação, corretamente, na maioria dos casos, casos mais complexos podem ocorrer como no exemplo mostrado na Figura 6.6. Esses casos ocorrem devido à utilização de apenas vetores normais na estimativa de detecção de splats próximos a descontinuidades.

6.4 Considerações Finais

Dentre as contribuições deste trabalho, a estimativa de vizinhança de esferas limitantes e a classificação dos *clip partners* para realização de um recorte adaptado unem eficiência e corretude no tratamento de descontinuidades do modelo. Apesar das demais técnicas apresentarem erros mínimos, devido a grande amostragem dos modelos gerados por scanners, deve-se ter em mente que a complexidade do *surface splatting* reside na quantidade de amostras do modelo. Métodos de redução de amostragem são utilizados para acelerar o processo de renderização, e, é nesse sentido, que métodos que renderizam modelos com poucas amostras devem tratar os problemas que não são visualizados com altas amostragens.

7 *Conclusão e Trabalhos Futuros*

Neste capítulo, os métodos e resultados apresentados nos capítulos anteriores são sumarizados. Finalmente, a dissertação é concluída por apontar as direções para possíveis trabalhos futuros.

7.1 Sumário

Nesta dissertação, investigaram-se métodos para renderização correta e eficiente de descontinuidades e interseções de superfícies em modelos baseados em splats. Nessa busca, diversos temas de computação gráfica voltados para modelos formados por pontos ou splats foram visitados: estimativas de vizinhança, detecção de descontinuidades, recorte de splats, segmentação de modelos e reconstrução de superfícies em espaço de imagem.

Os métodos dividem-se em duas etapas: uma antes e outra durante a renderização do modelo. Na primeira etapa, um grafo de vizinhança é construído utilizando a estimativa de esferas limitantes proposta para diminuir a quantidade de arestas do grafo, e assim reduzir custo computacional na busca por descontinuidades, sem deixar de detectar splats que necessitam de recorte. Isso é possível pelo fato de que há sempre uma aresta no grafo ligando os splats que se cruzam em arestas e cantos. O método de detecção de descontinuidades de Kobbelt *et al.* [Kobbelt et al. 2001] é utilizado para detectar pares de splats que pertencem a lados diferentes de uma descontinuidade, e que, possivelmente, serão recortados. Esses pares são conectados por arestas no grafo de vizinhança denominadas *sharp-edges*.

Após essa triagem inicial, para cada *sharp-edge*, um método de detecção de colisão entre círculos ou elipses é executado e, no caso de interseção, cada elemento ligado pela *sharp-edge* é inserido na lista de parceiros de recorte do outro. Essa lista é denominada de lista dos *clip partners*. O conjunto de *clip partners* são os vizinhos do splat que o recortarão adaptando-o à descontinuidade. Para a realização do recorte, o conjunto é classificado como côncavo ou convexo e sua interação com o splat também é classificada como côncava ou convexa. Dependendo dessas classificações são determinados os sentidos dos recortes de cada splat e é escolhido o

tipo de recorte: união ou interseção das áreas de corte de cada *clip partner*.

Ainda antes da renderização, um algoritmo de segmentação é executado sobre o grafo de vizinhança. Essa segmentação visa diferenciar os splats segundo a superfície a qual cada um pertence. Essa distinção entre os splats é necessária, pois os métodos de reconstrução da superfície em espaço de tela existentes utilizam apenas a distância entre os fragmentos como critério para combiná-los. Com a identificação das superfícies às quais cada splat pertence, os fragmentos são diferenciados de forma que não sejam combinados, e gerem artefatos nas áreas de interseção de superfícies.

Durante a renderização do modelo, a técnica de *Surface Splatting* é executada normalmente com algumas modificações. A renderização é dividida em dois passos: a projeção dos splats e a reconstrução da superfície. No primeiro passo, os splats são projetados e rasterizados segundo a técnica de Botsch *et al*. [Botsch et al. 2004]. Cada fragmento gerado é inserido em um *A-Buffer* e analisado contra os *clip partners* do splat que o gerou. Se o ponto 3D correspondente ao fragmento estiver na área de recorte seguindo a classificação realizada no pré-processamento, o fragmento é removido do *A-Buffer*.

Para a reconstruir a superfície em espaço de tela, o método de Zwicker [Zwicker et al. 2001] e de Räsänen [Räsänen 2002] são adaptados ao utilizar os identificadores de superfícies presentes nos fragmentos. A adaptação, basicamente, não permite a combinação de fragmentos com identificadores distintos, não permitindo a mistura de superfícies nas áreas de interseção. Essa adaptação pode ser utilizada com qualquer método de sombreamento dos splats.

7.2 Pontos positivos e negativos

A seguir são apresentadas breves análises dos pontos positivos e negativos das contribuições propostas nesta dissertação.

- **Grafo de vizinhança.** A estimativa de esferas limitantes certamente detecta splats presentes em descontinuidades, mesmo em casos de diferentes densidades de amostragem dos lados da aresta, sem a necessidade de um parâmetro fornecido pelo usuário. Outra vantagem deve-se à simetria de suas arestas. Essa característica acelera a geração do grafo e reduz a quantidade de arestas armazenadas. Entretanto, a quantidade de arestas depende exclusivamente do modelo. Quanto mais sobreposição dos splats, maior o número de arestas geradas, porém melhor é a reconstrução da superfície (Figura 7.1).

- **Recorte dos splats.** A classificação dos clip partners e o recorte dos splats nas proximi-

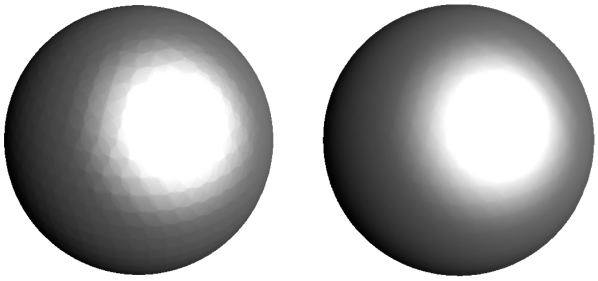

Figura 7.1: Relação entre número de arestas do grafo e qualidade de reconstrução da superfície. A esfera da direita apresenta a mesma quantidade de splats da esfera da direita, mas com raios 50% maiores. Esse aumento causou um aumento de quase 130% no número de arestas do grafo, mas melhorou a qualidade de reconstrução da superfície.

dades de descontinuidades adaptam splats às arestas e evitam o aparecimento de artefatos mesmo quando o observador está próximo do modelo. Essa adaptação parte do pressuposto de que a o mesmo conjunto de *clip partners* não apresente uma inflexão na curva da descontinuidade. Essa restrição não é muito severa como esclarecido na Seção 4.5. Entretanto, um caso bastante complexo e ainda não tratado ocorre quando alguns *clip partners* são vistos pelo splat e outros não. Esses casos ocorrem próximo de cantos, pois parte dos parceiros de recorte são de uma superfície e a outra pertence a uma terceira superfície.

- **Segmentação e Reconstrução da Superfície.** A adaptação dos métodos de Z-Threshold [Zwicker et al. 2001] e Z-Range [Räsänen 2002] é simples e reconstrói corretamente as superfícies por diferenciá-las. Entretanto, os métodos, antes sequenciais, tornaram-se laços, onde o número de voltas é igual ao número de superfícies detectadas no pixel. A segmentação do modelo funciona para o propósito de reconstruir as superfícies em espaço de tela na maioria dos casos, exceto para superfícies suaves que se alto-interceptam ou superfícies que se cruzam formando um ângulo menor do que um limiar definido. A utilização de um limiar fixo para o modelo inteiro não permite uma segmentação mais precisa e correta.

7.3 Trabalhos Futuros

Algumas desvantagens apontadas na Seção 7.2 permitem o surgimento de ideias para melhorias e adaptações do método proposto. Algumas sugestões de melhorias a serem feitas em trabalhos futuros são:

- Adicionar outras informações extraídas dos dados das amostras, além dos vetores normais, na escolha da função-peso. Assim, poderiam ser estudadas as inclusões de efeitos

como as distâncias entre amostras, e a distância de um vizinho ao plano definido pelo splat consultado, dentre outras.

- Certamente a escolha de um limiar fixo para o modelo inteiro é um problema a ser tratado. Uma descontinuidade pode formar um ângulo agudo num local e um ângulo obtuso mais adiante. Além disso, aumentar demais o limiar, pode detectar splats localizados em áreas com curvatura alta como pertencentes à descontinuidades. Um limiar adaptativo como em [Connor e Kumar 2010], além de livrar o usuário da escolha, é melhor aplicado, apesar do possível aumento da complexidade.

- A escolha dos recortes falha apenas em alguns casos próximos a cantos, onde alguns *clip partners* podem estar acima do splat e outros abaixo, não permitindo a classificação correta. Separar os *clip partners* em subconjuntos, onde cada subconjunto possuiria apenas *clip partners* pertencentes a uma mesma superfície, e classificar esses subconjuntos poderia realizar o recorte correto nesses casos.

Referências Bibliográficas

[Adams e Dutré 2003] ADAMS, B.; DUTRé, P. Interactive boolean operations on surfel-bounded solids. In: *ACM SIGGRAPH 2003 Papers*. [S.l.]: ACM, 2003. (SIGGRAPH '03), p. 651–656.

[Botsch et al. 2004] BOTSCH, M.; SPERNAT, M.; KOBBELT, L. Phong splatting. In: *Proceedings of Symposium on Point-Based Graphics 2004*. [S.l.: s.n.], 2004. p. 25–32.

[Carpenter 1984] CARPENTER, L. The a-buffer, an antialiased hidden surface method. In: *Computer Graphics (Proceedings of SIGGRAPH 84)*. [S.l.: s.n.], 1984. v. 18, p. 103–108.

[Connor e Kumar 2010] CONNOR, M.; KUMAR, P. Fast construction of k-nearest neighbor graphs for point clouds. *IEEE Transactions on Visualization and Computer Graphics*, v. 13, n. 4, p. 599–608, July 2010.

[Daniels et al. 2007] DANIELS, J. I.; HA, L. K.; OCHOTTA, T.; SILVA, C. T. Robust smooth feature extraction from point clouds. *Shape Modeling and Applications, International Conference on*, IEEE Computer Society, v. 0, p. 123–136, 2007.

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

[Demarsin et al. 2006] DEMARSIN, K.; VANDERSTRAETEN, D.; VOLODINE, T.; ROOSE, D. *Detection of Closed Sharp Feature Lines in Point Clouds for Reverse Engineering Applications*. May 2006.

[Fleishman et al. 2005] FLEISHMAN, S.; COHEN-OR, D.; SILVA, C. T. Robust moving least-squares fitting with sharp features. *ACM Trans. Graph.*, ACM, v. 24, p. 544–552, July 2005.

[Floater e Reimers 2001] FLOATER, M. S.; REIMERS, M. Meshless parameterization and surface reconstruction. *Computer Aided Geometric Design*, Elsevier Science Publishers B. V., v. 18, p. 77–92, March 2001.

[Gross e Pfister 2007] GROSS, M.; PFISTER, H. *Point-Based Graphics*. [S.l.]: Morgan Kaufmann Publishers, 2007.

[Guennebaud et al. 2004] GUENNEBAUD, G.; BARTHE, L.; PAULIN, M. Dynamic surfel set refinement for high-quality rendering. *Computer & Graphics*, Pergamon Press, Inc., v. 28, p. 827–838, December 2004.

[Gumhold et al. 2001] GUMHOLD, S.; WANG, X.; MACLEOD, R. Feature extraction from point clouds. In: *In Proceedings of the 10 th International Meshing Roundtable*. [S.l.: s.n.], 2001. p. 293–305.

[Jenke et al. 2006] JENKE, P.; WAND, M.; BOKELOH, M.; SCHILLING, A.; STRABER, W. Bayesian point cloud reconstruction. *Computer Graphics Forum*, v. 25, p. 379–388, September 2006.

[Jones et al. 2004] JONES, T. R.; DURAND, F.; ZWICKER, M. Normal improvement for point rendering. *IEEE Computer Graphics Applications*, IEEE Computer Society Press, v. 24, p. 53–56, July 2004.

[Jouppi e Chang 1999] JOUPPI, N.; CHANG, C.-F. Z3: an economical hardware technique for high-quality antialiasing and transparency. *1999 SIGGRAPH / Eurographics Workshop on Graphics Hardware*, p. 85–93, August 1999.

[Kobbelt e Botsch 2004] KOBBELT, L.; BOTSCH, M. A survey of point-based techniques in computer graphics. *Computers & Graphics*, v. 28, n. 6, p. 801 – 814, 2004.

[Kobbelt et al. 2001] 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*. [S.l.]: ACM, 2001. (SIGGRAPH '01), p. 57–66.

[Lalonde et al. 2005] LALONDE, J.-F.; UNNIKRISHNAN, R.; VANDAPEL, N.; HEBERT, M. Scale selection for classification of point-sampled 3-d surfaces. In: *Proceedings of the Fifth International Conference on 3-D Digital Imaging and Modeling*. [S.l.]: IEEE Computer Society, 2005. p. 285–292.

[Li et al. 2010] LI, B.; SCHNABEL, R.; KLEIN, R.; CHENG, Z.; DANG, G.; SHIYAO, J. Robust normal estimation for point clouds with sharp features. *Computers & Graphics*, v. 34, n. 2, p. 94–106, abr. 2010.

[Linsen 2001] LINSEN, L. *Point Cloud Representation*. [S.l.], 2001.

[Mérigot et al. 2009] MéRIGOT, Q.; OVSJANIKOV, M.; GUIBAS, L. Robust voronoi-based curvature and feature estimation. In: *2009 SIAM/ACM Joint Conference on Geometric and Physical Modeling*. [S.l.]: ACM, 2009. (SPM '09), p. 1–12.

[Mitra e Nguyen 2003] MITRA, N. J.; NGUYEN, A. Estimating surface normals in noisy point cloud data. In: *Proceedings of the nineteenth annual symposium on Computational geometry*. [S.l.]: ACM, 2003. p. 322–328.

[Oztireli et al. 2009] OZTIRELI, C.; GUENNEBAUD, G.; GROSS, M. Feature preserving point set surfaces based on non-linear kernel regression. *Computer Graphics Forum*, v. 28, n. 2, 2009.

[Pauly et al. 2005] PAULY, M.; KEISER, R.; ADAMS, B.; DUTRé, P.; GROSS, M.; GUIBAS, L. J. Meshless animation of fracturing solids. In: *ACM SIGGRAPH 2005 Papers*. [S.l.]: ACM, 2005. (SIGGRAPH '05), p. 957–964.

[Pauly et al. 2003] PAULY, M.; KEISER, R.; GROSS, M. Multi-scale feature extraction on point-sampled surfaces. *Computer Graphics Forum*, v. 22, p. 281–289, September 2003.

[Pauly et al. 2003] PAULY, M.; KEISER, R.; KOBBELT, L. P.; GROSS, M. Shape modeling with point-sampled geometry. In: *ACM SIGGRAPH 2003 Papers*. [S.l.]: ACM, 2003. (SIGGRAPH '03), p. 641–650.

[Räsänen 2002] RäSäNEN, J. *Surface Splatting: Theory, Extensions and Implementation*. Dissertação (Mestrado) — Helsinki University of Technology, 2002.

[Vanco e Brunnett 2002] VANCO, M.; BRUNNETT, G. Direct segmentation for reverse engineering. In: *Cyber Worlds, 2002. Proceedings. First International Symposium on*. [S.l.: s.n.], 2002. p. 24–31.

[Weber et al. 2010] WEBER, C.; HAHMANN, S.; HAGEN, H. Sharp feature detection in point clouds. In: *Proceedings of the 2010 Shape Modeling International Conference*. [S.l.]: IEEE Computer Society, 2010. (SMI '10), p. 175–186.

[Wicke et al. 2004] WICKE, M.; TESCHNER, M.; GROSS, M. Csg tree rendering for point-sampled objects. In: *Proceedings of the Computer Graphics and Applications, 12th Pacific Conference*. [S.l.]: IEEE Computer Society, 2004. (PG '04), p. 160–168.

[Winner et al. 1997] WINNER, S.; KELLEY, M.; PEASE, B.; RIVARD, B.; YEN, A. Hardware accelerated rendering of antialiasing using a modified a-buffer algorithm. In: *Proceedings of SIGGRAPH 97*. [S.l.: s.n.], 1997. p. 307–316.

[Woo et al. 2002] WOO, H.; KANG, E.; WANG, S.; LEE, K. H. A new segmentation method for point cloud data. *International Journal of Machine Tools and Manufacture*, v. 42, p. 167–178, January 2002.

[Wu e Kobbelt 2004] WU, J.; KOBBELT, L. Optimized sub-sampling of point sets for surface splatting. *Computer Graphics Forum*, v. 23, p. 643–652, September 2004.

[Zwicker et al. 2002] ZWICKER, M.; PAULY, M.; KNOLL, O.; GROSS, M. Pointshop 3d: an interactive system for point-based surface editing. In: *Proceedings of SIGGRAPH 2002*. [S.l.: s.n.], 2002.

[Zwicker et al. 2001] ZWICKER, M.; PFISTER, H.; BAAR, J. van; GROSS, M. Surface splatting. In: *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*. [S.l.]: ACM, 2001. p. 371–378.

[Zwicker et al. 2004] ZWICKER, M.; RäSäNEN, J.; BOTSCH, M.; DACHSBACHER, C.; PAULY, M. Perspective accurate splatting. In: *Proceedings of Graphics Interface 2004*. [S.l.]: Canadian Human-Computer Communications Society, 2004. (GI '04), p. 247–254.