Programa Interdisciplinar de Pós-Graduação em

Computação Aplicada

Mestrado Acadêmico

Fabrício Müller da Silva

Reamostragem Adaptativa para Simplificação de Nuvens de Pontos

São Leopoldo, 2015

Fabrício Müller da Silva

REAMOSTRAGEM ADAPTATIVA PARA SIMPLIFICAÇÃO DE NUVENS DE PONTOS

Dissertação apresentada como requisito parcial para a obtenção do título de Mestre, pelo Programa Interdisciplinar de Pós-Graduação em Computação Aplicada da Universidade do Vale do Rio dos Sinos — UNISINOS

Orientador:
Prof. Dr. Luiz Gonzaga da Silveira Jr.

São Leopoldo
2015

S586r Silva, Fabrício Müller da
Reamostragem adaptativa para simplificação de nuvens de pontos / por Fabrício Müller da Silva. – 2015.
73 f. : il., 30 cm.

Dissertação (mestrado) — Universidade do Vale do Rio dos Sinos, Programa de Pós-Graduação em Computação Aplicada, 2015.
Orientação: Prof. Dr. Luiz Gonzaga da Silveira Jr.

1. Nuvem de pontos. 2. Simplificação. 3. Reconstrução de superfícies. 4. Análise de componentes principais. I. Título.

CDU 004.421

Catalogação na Fonte:
Bibliotecária Vanessa Borges Nunes - CRB 10/1556

Fabrício Müller da Silva

Reamostragem Adaptativa para Simplificação de Nuvens de Pontos

Dissertação apresentada à Universidade do Vale do Rio dos Sinos – Unisinos, como requisito parcial para obtenção do título de Mestre em Computação Aplicada.

Aprovado em 31/08/2015

BANCA EXAMINADORA

Prof. Dr. Luiz Paulo Luna de Oliveira – UNISINOS

Prof. Dr. Manuel Menezes de Oliveira Neto – UFRGS

Prof. Dr. Luiz Gonzaga da Silveira Jr. (Orientador)

Visto e permitida a impressão
São Leopoldo,

Prof. Dr. Cristiano André da Costa
Coordenador PPG em Computação Aplicada

Para minha mulher, Gislaine, e meu filho, Pedro.

Em memória de minha mãe, Maria Reni.

AGRADECIMENTOS

Agradeço aos professores Luiz Gonzaga Jr. e Luiz Paulo Luna pelo apoio e dedicação na orientação deste trabalho.

RESUMO

Este trabalho apresenta um algoritmo para simplificação de nuvens de pontos baseado na inclinação local da superfície amostrada pelo conjunto de pontos de entrada. O objetivo é transformar a nuvem de pontos original no menor conjunto possível, mantendo as características e a topologia da superfície original. O algoritmo proposto reamostra de forma adaptativa o conjunto de entrada, removendo pontos redundantes para manter um determinado nível de qualidade definido pelo usuário no conjunto final. O processo consiste em um particionamento recursivo do conjunto de entrada através da Análise de Componentes Principais (PCA). No algoritmo, PCA é aplicada para definir as partições sucessivas, para obter uma aproximação linear (por planos) em cada partição e para avaliar a qualidade de cada aproximação. Por fim, o algoritmo faz uma escolha simples de quais pontos serão mantidos para representar a aproximação linear de cada partição. Estes pontos formarão o conjunto de dados final após o processo de simplificação. Para avaliação dos resultados foi aplicada uma métrica de distância entre malhas de polígonos, baseada na distância de Hausdorff, comparando a superfície reconstruída com a nuvem de pontos original e aquela reconstruída com a nuvem filtrada. Os resultados obtidos com o algoritmo conseguiram uma taxa de até 95% de compactação do conjunto de dados de entrada, diminuindo o tempo total de execução do processo de reconstrução, mantendo as características e a topologia do modelo original. A qualidade da superfície reconstruída com a nuvem filtrada também é atestada pela métrica de comparação.

Palavras-chave: Nuvem de Pontos. Simplificação. Reconstrução de Superfícies. Análise de Componentes Principais.

ABSTRACT

This paper presents a simple and efficient algorithm for point cloud simplification based on the local inclination of the surface sampled by the input set. The objective is to transform the original point cloud in a small as possible one, keeping the features and topology of the original surface. The proposed algorithm performs an adaptive resampling of the input set, removing unnecessary points to maintain a level of quality defined by the user in the final dataset. The process consists of a recursive partitioning in the input set using Principal Component Analysis (PCA). PCA is applied for defining the successive partitions, for obtaining the linear approximations (planes) for each partition, and for evaluating the quality of those approximations. Finally, the algorithm makes a simple choice of the points to represent the linear approximation of each partition. These points are the final dataset of the simplification process. For result evaluation, a distance metric between polygon meshes, based on Hausdorff distance, was defined, comparing the reconstructed surface using the original point clouds and the reconstructed surface using the filtered ones. The algorithm achieved compression rates up to 95% of the input dataset, while reducing the total execution time of reconstruction process, keeping the features and the topology of the original model. The quality of the reconstructed surface using the filtered point cloud is also attested by the distance metric.

Keywords: Point Cloud. Simplification. Surface Reconstruction. Principal Component Analysis.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

Atualmente os processos e equipamentos de captura 3D têm alto grau de precisão e geram nuvens muito densas e detalhadas para representar modelos de objetos reais. Estes grandes conjuntos de dados ocupam mais recursos para armazenar, transferir e processar. Além disso, nestes conjuntos de dados existem pontos que podem ser redundantes de acordo com a aplicação do modelo em 3 dimensões. Desta forma, o desenvolvimento de algoritmos para simplificação e remoção de pontos redundantes do conjunto de amostras é uma área de pesquisa importante em processamento de imagens. Segundo o trabalho de (NGUYEN; BAC; DANIEL, 2013), apesar do longo tempo de pesquisa na área de obtenção e processamento de modelos em 3 dimensões, o processo de simplificação de nuvens de pontos ainda oferece desafios.

A simplificação de nuvens de pontos tem o objetivo de reduzir a quantidade de pontos amostrados de uma superfície para reduzir o consumo de memória e otimizar o processamento do modelo. Após a aplicação do processo de filtragem e remoção dos pontos, a topologia e as características de superfície devem ser mantidas. A simplificação do conjunto de entrada é um passo importante para diminuir o esforço computacional, mantendo a fidelidade do modelo reconstruído. Aplicações como a captura de modelos para jogos ou animação, processamento e visualização em dispositivos móveis, computadores com configurações distintas, transferência de dados entre dispositivos, processamento de grandes nuvens de pontos, entre outras para modelagem de objetos 3D, podem beneficiar-se do processo de simplificação.

Os algoritmos de simplificação podem ser classificados em duas categorias, os métodos baseados na malha de polígonos e os métodos baseados na nuvem de pontos (SHI; LIANG; LIU, 2011) e (LI; XU; SHEN, 2014). Este trabalho apresenta um algoritmo para simplificação, onde o processo é aplicado diretamente na nuvem de pontos, sem a construção de uma malha de polígonos. Neste caso, o processo é executado sem conhecimento da topologia e dos vetores normais da superfície amostrada. O objetivo é remover pontos redundantes a partir do conjunto de entrada, sem reduzir a qualidade do processo de reconstrução da superfície. É importante lembrar que a qualidade visual será um critério subjetivo do usuário, assim como o quanto será importante abrir mão da qualidade para aumentar a taxa de simplificação, de acordo com a aplicação em uso.

A abordagem proposta aplica a Análise de Componentes Principais, ou em inglês, Principal Component Analysis (PCA), na nuvem de pontos de entrada. Neste contexto, as componentes principais (Principal Components - PCs) são usadas para dividir o conjunto de entrada sucessivamente em planos, buscando pela melhor aproximação de uma certa região da nuvem. Determinar um plano de aproximação para um conjunto de pontos permite determinar uma célula de onde podem ser extraídos os pontos que representam a melhor aproximação linear para aquela região da superfície amostrada, descartando os demais do conjunto de entrada. No algoritmo desenvolvido não há necessidade de pré-processamento do conjunto de entrada, organização dos pontos em uma estrutura de dados e definição prévia de vizinhança dos pontos. O processo

de particionamento usa diretamente as componentes principais, dividindo o conjunto de entrada recursivamente, obtendo as regiões que serão simplificadas. Como resultado, o processo consome menos memória e tempo de processamento. O processo de simplificação traz algumas questões a serem abordadas, como o custo computacional do pré-processamento e a definição de métricas adequadas para a remoção dos pontos durante o processo de reamostragem.

Para avaliação dos resultados dois algoritmos para reconstrução de superfícies diferentes são aplicados nos conjuntos de pontos gerados após a filtragem, Poisson Surface Reconstruction (KAZHDAN; BOLITHO; HOPPE, 2006a) e Smooth Signed Distance Surface Reconstruction (SSD) (CALAKLI; TAUBIN, 2011a). As malhas de polígonos geradas pelos processos de reconstrução aplicados nas nuvens originais e simplificadas são comparadas numericamente comparando a distância de Hausdorff (CIGNONI; ROCCHINI; SCOPIGNO, 1998) entre as superfícies reconstruídas.

As principais contribuições deste trabalho são:

- Algoritmo para simplificação da nuvem de pontos baseado na aplicação de PCA ao conjunto de entrada para definir as partições sucessivas, obter a aproximação linear em cada partição e avaliar a qualidade de cada aproximação. Esta abordagem permite classificar de forma mais natural uma nuvem de pontos em regiões bem aproximadas por planos.

- Particionamento do conjunto de entrada através de uma divisão do espaço com máxima independência linear, obtendo uma forma ótima de aproximação geométrica. Como resultado, os agrupamentos finais são os melhores candidatos para simplificação, fazendo com que a forma da seleção dos pontos para o conjunto simplificado tenha menor influência sobre o resultado.

- Aplicação de uma métrica para comparação dos resultados. A definição de uma métrica de distância entre as malhas de polígonos dos modelos reconstruídos nos permite verificar a diferença geométrica entre eles. Esta medida permite ao usuário comparar os resultados da simplificação usando diferentes abordagens (variando os parâmetros do algoritmo, aplicando diferentes algoritmos de reconstrução ou diferentes algoritmos de simplificação). Contudo, a definição de o que é uma boa, ou pequena, distância entre o modelo original e aquele simplificado dependerá da aplicação dada pelo usuário.

O texto deste trabalho apresenta no capítulo 2 os trabalhos relacionados com as técnicas já existentes para simplificação de nuvens de pontos. No capítulo 3 os conceitos básicos usados no trabalho, mostrando a definição de Análise de Componentes Principais, nuvens de pontos e reconstrução de superfícies. O capítulo 4 apresenta o algoritmo desenvolvido, com aplicação de PCA na nuvem de pontos, e a métrica usada para comparação dos resultados. Os resultados obtidos com o algoritmo estão no capítulo 5 e, por fim, no capítulo 6 estão a conclusão e considerações finais.

2 TRABALHOS RELACIONADOS

Existem diferentes trabalhos que abordam o problema da simplificação, diretamente na nuvens de pontos ou na malha de polígonos. O uso de PCA, autovalores e autovetores, para medir a inclinação da superfície é comum nas diferentes abordagens. Um dos primeiros trabalhos que aplicaram a simplificação diretamente na nuvem de pontos foi o de Alexa et al. (2001). Depois deste, Pauly, Gross e Kobbelt (2002) fazem uma análise de diferentes algoritmos para simplificação. Os métodos atuais, em sua maioria, usam clusterização, vizinhos locais, voxels e análise da inclinação da superfície para determinar quais pontos podem ser removidos do conjunto de entrada. Existem também aqueles que geram um conjunto com novos pontos, corrigindo sua posição no espaço, a partir dos originais. Os métodos podem ser adaptativos ou não.

Os autores no artigo (NGUYEN; BAC; DANIEL, 2013) apresentam um método de simplificação baseado na curvatura da superfície e na densidade dos pontos. O método aplica o processo de simplificação a partir da divisão do conjunto de entrada em células regulares, com diâmetro definido como parâmetro de entrada. As células são voxels que agrupam subconjuntos de pontos com seus vizinhos locais. O processo de simplificação é divido em três etapas. A primeira simplifica as bordas da superfícies avaliando a densidade dos pontos e sua inclinação. A segunda etapa apenas executa uma adaptação da resolução, mantendo o ponto médio de cada voxel. A última etapa é um processo adaptativo da simplificação, que executa a divisão das células em novos voxels avaliando a inclinação da superfície e densidade de pontos dentro de cada célula. Por fim, são mantidos apenas os pontos médios das células finais. O método tem o diferencial de tratar as bordas da superfície em um processo separado.

No artigo (SHI; LIANG; LIU, 2011) o modelo apresentado usa o algoritmo K-means Clustering para simplificar um conjunto de pontos, usando a inclinação do vetor normal como limite para subdivisão de cada agrupamento. Regiões com maior curvatura são subdivididas em novas células. A abordagem usa uma K-D Tree como estrutura de dados para organizar as células do processo de particionamento e seleciona o centroide para representar cada divisão. Os pontos são agrupados nas novas células de acordo com a similaridade da inclinação de seu vetor normal. Ao final do processo de particionamento são mantidos os centroides de cada célula no conjunto de pontos final. O resultado é uma reamostragem adaptativa, mas com distribuição uniforme dos pontos de acordo com o particionamento do conjunto de entrada. É necessário o pré-processamento do conjunto de amostras para estimativa do vetor normal.

O trabalho desenvolvido por Li, Xu e Shen (2014) apresenta um algoritmo para simplificação baseado no vetor normal de cada ponto à superfície. O algoritmo também usa uma K-D Tree como estrutura de dados para particionar o conjunto de entrada obtendo os k vizinhos mais próximos de um ponto e um raio de busca. O resultado é a divisão do conjunto de entrada em voxels. O primeiro passo é a redução da amostra, com a redução da resolução de pontos do conjunto de entrada, mantendo apenas aqueles mais próximos do centroide de cada voxel (similar a (NGUYEN; BAC; DANIEL, 2013)). O resultado é uma reamostragem regular da nuvem de

pontos. Depois, o algoritmo estima o vetor normal de cada ponto e, através dos autovalores, usa aqueles com maior inclinação como pontos característicos da superfície. O algoritmo executa o particionamento do conjunto de entrada com raios de busca diferentes para pontos característicos e não característicos, gerando uma divisão adaptativa do conjunto de entrada para execução da reamostragem.

Miao, Pajarola e Feng (2009) desenvolveram um modelo adaptativo baseado na curvatura da superfície onde os pontos estão localizados. Esta abordagem executa a reamostragem do conjunto de entrada mantendo maior densidade de pontos em regiões de maior curvatura e menor densidade em regiões de menor curvatura. O algoritmo é baseado na vizinhança de um dado ponto p, usando a posição dos vizinhos e a variação do vetor normal. Assim, a vizinhança é definida por um raio de distância entre os pontos e um limite no ângulo de desvio do vetor normal. O processo de particionamento constrói uma árvore binária, onde as folhas são as células do conjunto de entrada. O centroide da célula é mantido no conjunto final. A implementação usa PCA para estimar o vetor normal do centroide de cada célula para verificação do limite de angulação no processo de particionamento.

Existem, ainda, outros métodos desenvolvidos com abordagem semelhante a estes já vistos. Li et al. (2009) usam o método Affinity Propagation Clustering para particionamento do conjunto de entrada, com a obtenção dos vizinhos mais próximos através da semelhança do ângulo do vetor normal de cada ponto. Yu et al. (2010) constroem um particionamento hierárquico do conjunto de entrada baseado no método K-means, aplicando PCA para obter os pontos característicos de cada célula. E Lee e Huang (2011) obtêm os pontos característicos da superfície para particionamento através dos planos de aproximação, calculados pela estimativa da curvatura aplicando o método Discrete Shape Operator (DSO).

Na abordagem desenvolvida neste trabalho, PCA é aplicada para definir a divisão sucessiva dos dados de entrada e para obter a aproximação linear dos pontos do conjunto. Com base nas componentes principais menos significativas, o algoritmo determina a qualidade de cada aproximação, que define se o algoritmo fará ou não novas divisões locais subsequentes para a obtenção de aproximações adicionais. Desta forma, o particionamento do conjunto de entrada não será em um formato regular e os planos utilizados no processo de particionamento terão diferentes inclinações. Como consequência, o algoritmo produz uma reamostragem mais adaptativa da nuvem de pontos de entrada, avaliando as curvaturas locais do modelo original, dadas pela variação espacial das terceiras componentes principais em cada passo da aproximação. Além disso, a implementação não usa estruturas de dados, como Octrees ou KD-Trees, para particionar o conjunto de dados de entrada. Outra diferença é a forma de seleção dos pontos que representarão o plano de aproximação, mantendo apenas três para cada região obtida no processo de particionamento executado com as componentes principais. Por fim, o algoritmo não precisa executar um pré-processamento das amostras e o particionamento prévio do conjunto de entrada em voxels para obter a inclinação local da superfície.

3 CONCEITOS BÁSICOS

Este capítulo tem por objetivo apresentar os conceitos básicos e o estudo teórico dos assuntos principais deste trabalho. Serão revisados os conceitos de análise de componentes principais, componentes principais no contexto geométrico, nuvens de pontos, reconstrução de superfícies a partir de uma nuvem de pontos e métodos de reconstrução.

3.1 Principal Component Analysis

A área da análise de múltiplas variáveis é um método estatístico que considera duas ou mais variáveis aleatórias como uma entidade única e procura encontrar uma relação entre elas. Na técnica de Análise de Componentes Principais tal relação é definida por uma matriz de covariâncias S, que define uma medida de semelhança entre as variáveis, uma correlação entre elas. O método é baseado em uma definição da álgebra linear: seja A uma matriz quadrada $d \times d$, um autovalor de A é um número $\lambda \in R$ tal que existe um vetor $\vec{v} \in R^d$, não nulo, que satisfaça a igualdade $A\vec{v} = \lambda\vec{v}$. Neste caso dizemos que $\vec{v}$ é um autovetor de A associado a λ. Mais ainda, se A for uma matriz simétrica, não singular (invertível), ela pode ser reduzida a uma matriz diagonal L pela multiplicação por uma matriz ortonormal U tal que $U'AU = L$. Os elementos da diagonal de L são os autovalores de A e as colunas de U são os autovetores de A associados com cada elemento da matriz diagonal L (JACKSON, 1991). Sendo assim, executar PCA em um conjunto de dados X, formado por d variáveis, é diagonalizar a matriz de covariâncias $S_{(d\times d)}$. Isto é, mudá-la da base canônica para a base formada pelos autovetores $\{\vec{v_1}, \vec{v_2}, \cdots, \vec{v_d}\}$. A matriz de covariâncias $S_{(d\times d)}$ sempre será simétrica e não singular, possuindo as seguintes propriedades:

- $S_{(d\times d)}$ possui sempre d autovalores, $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_d \geq 0$. Cada autovalor $\lambda_1, \lambda_2, \cdots, \lambda_d$ está associado a um autovetor $\vec{v_1}, \vec{v_2}, \cdots, \vec{v_d}$ respectivamente.

- É possível construir uma base ortonormal com os autovetores de $S_{(d\times d)}$: $B = \{\vec{v_1}, \vec{v_2}, \cdots, \vec{v_d}\}$, isto é, tal que $\vec{v_1} \perp \vec{v_2} \perp \cdots \perp \vec{v_d}$ e $|\vec{v_1}| = |\vec{v_2}| = \cdots = |\vec{v_d}| = 1$.

- $\vec{v_1}$ aponta para a direção de maior variabilidade (variância), $\vec{v_2}$ aponta para a direção de segunda maior variabilidade e assim sucessivamente até $\vec{v_d}$, que aponta para a direção de menor variabilidade (menor variância dos dados). $\vec{v_1}$ é chamado de primeira componente principal, $\vec{v_2}$ é chamado de segunda componente principal e assim sucessivamente.

- λ_i é o valor da variância na direção $\vec{v_i}$ e $\sqrt{\lambda_i}$ é o valor do desvio padrão na mesma direção.

A matriz de covariâncias $S_{(d\times d)}$ é gerada a partir do conjunto de entrada X, conforme a equação 3.1.

$$S_{(d\times d)} = \begin{bmatrix} \sigma_{1\,1} & \sigma_{1\,2} & \cdots & \sigma_{1\,j} \\ \sigma_{2\,1} & \sigma_{2\,2} & \cdots & \sigma_{2\,j} \\ \vdots & \vdots & & \vdots \\ \sigma_{i\,1} & \sigma_{i\,2} & \cdots & \sigma_{i\,j} \end{bmatrix}, i = 1 \cdots d \; e \; j = 1 \cdots d. \tag{3.1}$$

O valor das variâncias e das covariâncias das variáveis do problema são obtidos conforme a equação 3.2. Na equação, n é o número total de amostras do conjunto de entrada X. Quando $i = j$ o valor de σ, representado por σ_i^2, é chamado de variância da variável i e de covariância entre as variáveis i e j quando $i \neq j$.

$$\sigma_{i\,j} = \frac{n \sum_{k=1}^{n} x_{i\,k} x_{j\,k} - \sum_{k=1}^{n} x_{i\,k} \sum_{k=1}^{n} x_{j\,k}}{n(n-1)} \tag{3.2}$$

Do ponto de vista geométrico, onde o conjunto de entrada X terá 3 variáveis (x, y, z) que representam um ponto no espaço (R^3), os autovetores de S podem ser usados para gerar planos de aproximação aos pontos deste conjunto. Uma das formas de definir a equação de um plano é usar um dado ponto P sobre o plano e o vetor normal a ele. Neste contexto, $\vec{v_3}$ será perpendicular (normal) ao plano α gerado por $\vec{v_1}$ e $\vec{v_2}$ e o ponto P sobre o plano será o ponto médio do conjunto de entrada, representado por $\overline{\vec{x}}$. Considerando todos os possíveis planos do R^3, o plano α será aquele que minimiza a soma dos quadrados das distâncias dos pontos a este plano, consequência do fato que $\vec{v_3}$ aponta para a direção de menor variância dos pontos.

3.2 Nuvens de Pontos

Uma nuvem de pontos é um conjunto de pontos no espaço (R^3) que representam uma superfície em três dimensões. Desta forma, os pontos deste conjunto são uma amostra da parte externa da superfície da qual pertencem. Cada ponto possui uma coordenada no espaço (x, y, z) e podem ter outros atributos associados, como a cor, o vetor normal à superfície, entre outros. Nuvens de pontos são obtidas a partir de um processo de captura em 3 dimensões, os métodos mais comuns são o LiDAR (Light Detection And Ranging), iluminação estruturada e métodos MVS (multi-view stereo) (CALAKLI; TAUBIN, 2011a). A saída do processo de captura é um arquivo com as informações do conjunto de pontos, a Figura 1 mostra um exemplo de visualização deste tipo de dados.

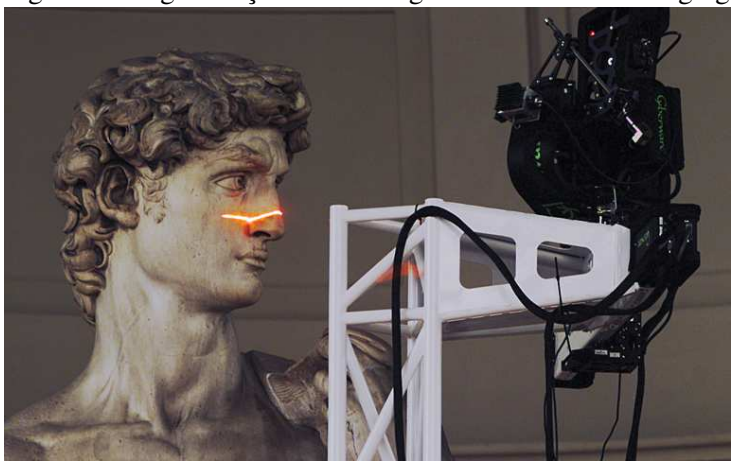
Entre os métodos pesquisados, o mais usado para a obtenção de uma nuvem de pontos é o LiDAR, nesta tecnologia um feixe de luz (laser) é disparado contra um objeto para obtenção de amostras (pontos) da superfície. Sabendo a direção do pulso enviado e o tempo de retorno ao emissor, é possível calcular a posição do ponto no espaço. Sucessivas leituras executadas em uma varredura da superfície têm como saída uma nuvem de pontos que forma um modelo em 3 dimensões do objeto (HODGETTS, 2013). O processo permite, ainda, a captura de outras propriedades do objeto, como a cor e a intensidade do laser. Aplicando técnicas de triangulação

Figura 1 – Visualização de uma nuvem de pontos

Fonte: (CALAKLI; TAUBIN, 2011a)

é possível, por exemplo, obter o vetor normal de um ponto com relação a sua superfície. Uma foto tirada do projeto desenvolvido por Levoy (1999), na Figura 2, traz um exemplo de captura em 3 dimensões com laser.

Figura 2 – Digitalização usando Light Detection And Ranging

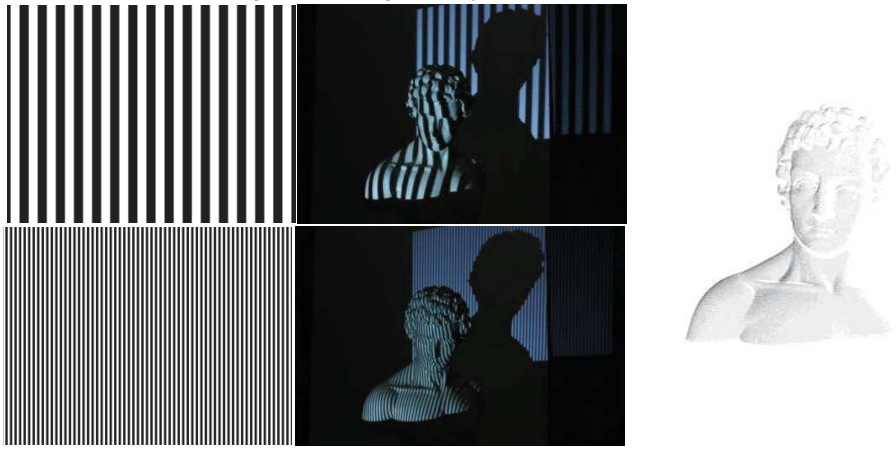


Fonte: (LEVOY, 1999)

A iluminação estruturada é um método confiável e de baixo custo para gerar superfícies tridimensionais. Na forma mais simples, o método usa um par de projetor e câmera para iluminar a superfície com um padrão conhecido em sequência e para capturar as imagens geradas. Através das imagens capturadas é possível encontrar a intersecção de diferentes projeções sobre um mesmo ponto. Esta correspondência permite a obtenção de um modelo em 3 dimensões do objeto (LANMAN; CRISPELL; TAUBIN, 2009). Uma das principais vantagens dos sistemas baseados em luz estruturada é a simplicidade e o custo, oferecendo uma ferramenta efetiva e com alta precisão. Esta precisão é obtida através da correta calibragem da câmera e do projetor

(MORENO; TAUBIN, 2012).

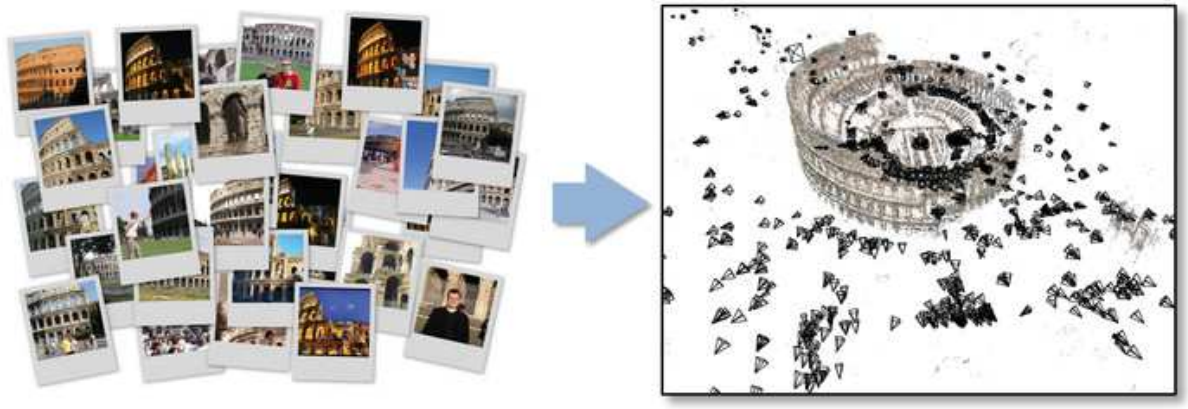
Figura 3 – Digitalização com luz estruturada



Fonte: (MORENO; TAUBIN, 2012)

Os métodos MVS trabalham com um conjunto de múltiplas imagens obtidas de câmeras em diferentes pontos de vista conhecidos para gerar um modelo em 3 dimensões de um objeto ou cena. O objetivo é encontrar quais propriedades da geometria original são preservadas em representações projetadas de uma imagem tridimensional. A principal propriedade usada nos métodos MVS para obter modelos em 3 dimensões são as linhas de perspectiva (HARTLEY; ZISSERMAN, 2003). Os algoritmos atuais nesta forma de geração de representação em 3 dimensões variam na forma de representação da cena, modelo de visibilidade, métrica de foto-consistência, modelo prévio e algoritmo de reconstrução. Cada uma destas propriedades definirá o comportamento do método. Representação da cena é o método usado para reproduzir a geometria de um objeto que pode usar voxels (representação de um pixel em um espaço tridimensional), superfície implícita, malha de polígonos e mapas de profundidade. A métrica de foto-consistência é usada para encontrar uma correlação visual entre duas ou mais imagens. Este processo geralmente é executado comparando pixel por pixel de uma imagem com outra com o objetivo de encontrar diferentes pontos de vista da mesma região capturada em diferentes fotografias. O modelo de visibilidade determina quais pontos de vista devem ser usados para obter a correlação entre diferentes imagens. Um modelo prévio pode ser usado para estabelecer uma geometria quando não é possível obter esta propriedade com precisão a partir das imagens capturadas. Por fim, o algoritmo de reconstrução determina a técnica usada para obter a superfície em 3 dimensões, de onde será obtida a nuvem de pontos que representa o objeto (SEITZ et al., 2006). Existem diferentes aplicações das técnica MVS e suas variações, a Figura 4 mostra um exemplo tirado do trabalho desenvolvido por Snavely, Seitz e Szeliski (2006).

Figura 4 – Aplicação da técnica Multi-View Stereo



Fonte: (SNAVELY; SEITZ; SZELISKI, 2006)

3.3 Reconstrução de Superfícies

Reconstrução de superfícies a partir de uma nuvem de pontos é o processo de obter um modelo em três dimensões das amostras obtidas desta superfície que, em geral, não trazem informações sobre sua conectividade ou sua posição. Conforme a pesquisa de Schall e Samozino (2005), existem duas abordagens principais para a reconstrução de superfícies. A primeira forma são os métodos baseados na triangulação dos pontos do conjunto de entrada. Entre estes métodos está a triangulação de Delaunay, que têm como ideia principal encontrar os pontos vizinhos em um conjunto de entrada X, em todas as possíveis direções, para todas as amostras. Para gerar a superfície reconstruída os métodos calculam um subconjunto da triangulação, existem diferentes algoritmos nesta forma de abordagem, os principais são Crust (AMENTA; BERN, 1998) e Cocone (AMENTA et al., 2000). A segunda forma são os chamados métodos volumétricos, que aproximam os dados através de uma função. Nestes métodos, os algoritmos são baseados na obtenção de uma superfície implícita, com o cálculo de uma função $y = f(x)$, onde a curva de nível zero $Z(f) = \{x : f(x) = 0\}$ aproxima a superfície formada pelos pontos da amostra. O objetivo é encontrar uma equação que aproxime a superfície S, definida por $S = \{x : f(x) = 0\}$, a partir de um conjunto finito de pontos $X = \{x_i\} \in R^3$, onde x_i é um ponto no espaço.

O problema da reconstrução é motivado por um grande número de aplicações e gerou uma variedade de algoritmos para sua solução (BERGER et al., 2013). Os algoritmos têm como ponto principal a forma esperada dos dados de entrada e a forma de gerar a saída da superfície reconstruída. Os dados de entrada podem ser, basicamente, imagens com mapa de profundidade e nuvens de pontos com ou sem a informação do vetor normal. A forma de saída do algoritmo possui dois componentes principais, a representação da superfície e a dependência dos dados de entrada. A representação da superfície pode ser com uma superfície paramétrica, uma superfície implícita ou por uma malha triangularizada. A dependência dos dados de entrada define se os pontos serão interpolados para todo conjunto, ou um subconjunto, das amostras como vértices

ou se os pontos serão aproximados por uma função (BERGER et al., 2013).

Ainda segundo a pesquisa de Berger et al. (2013), restringir o processo de reconstrução para usar como vértices de uma malha de polígonos somente os pontos do conjunto de entrada, geralmente, terá resultados limitados quando os dados da nuvem não forem uniformes, estiverem incompletos ou com ruído. O processo de simplificação fará a remoção de pontos do conjunto de amostras original, gerando um novo conjunto de dados com estas imperfeições. Algoritmos que geram uma superfície aproximada são mais flexíveis para estas situações. Nestes casos, a saída será a triangularização de uma superfície gerada a partir de uma função implícita que melhor aproxima os pontos do conjunto de entrada. Muitos destes algoritmos obtém um campo vetorial que é normal a superfície aproximada, calculando a menor distância usando um plano estimado para cada ponto de entrada. Nesta situação, a presença da informação do vetor normal de cada ponto no conjunto de dados de entrada torna-se crítico para o processo de reconstrução (conjuntos de pontos que contém a informação do vetor normal são chamados de nuvens de pontos orientados).

Seguindo esta linha de algoritmos existem diferentes trabalhos listados por Schall e Samozino (2005) e Berger et al. (2013), entre eles estão: Poisson Surface Reconstruction (KAZHDAN; BOLITHO; HOPPE, 2006a), Reconstruction of Solid Models from Oriented Point Sets (Fourier) (KAZHDAN, 2005), Streaming Surface Reconstruction Using Wavelets (MANSON; PETROVA; SCHAEFER, 2008), Multi-level Partition of Unity Implicits (OHTAKE et al., 2003), 3D Scattered Data Interpolation and Approximation with Multilevel Compactly Supported RBFs (OHTAKE; BELYAEV; SEIDEL, 2005), Provably Good Moving Least Squares (KOLLURI, 2005) e Algebraic Point Set Surfaces (GUENNEBAUD; GROSS, 2007). A esta lista ainda podemos adicionar o trabalho de Calakli e Taubin (2011a), Smooth Signed Distance Surface Reconstruction.

Concluindo, os experimentos foram executados com algoritmos de reconstrução baseados no cálculo de uma função implícita, trabalhando com nuvens de pontos orientados. Nesta família de algoritmos estão os métodos Smooth Signed Distance Surface Reconstruction (SSD) e Poisson Surface Reconstruction que foram usados neste trabalho.

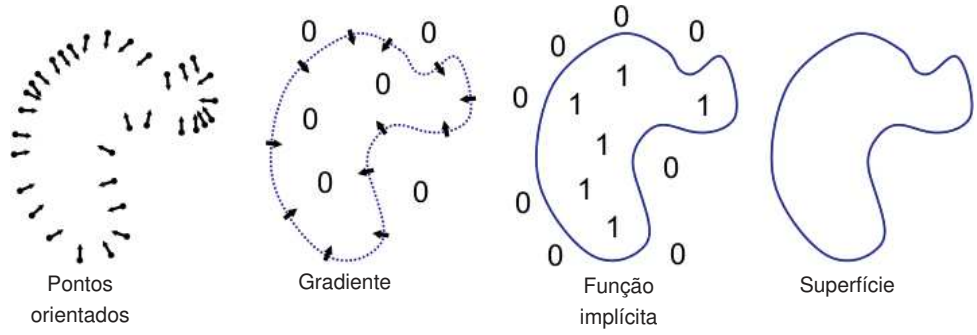
3.3.1 Poisson Surface Reconstruction

O método Poisson Surface Reconstruction tornou-se uma das principais formulações para reconstrução de superfícies nas pesquisas recentes. O algoritmo foi desenvolvido por Kazhdan, Bolitho e Hoppe (2006a) e existem diferentes implementações do método que exploram o processamento paralelo (BOLITHO et al., 2007), (BOLITHO et al., 2009) e a execução em GPU (ZHOU et al., 2011).

O algoritmo de Poisson para reconstrução de superfícies obtém uma função implícita, com valor igual a zero nos pontos do conjunto de entrada, e valor do gradiente da função igual ao valor do vetor normal associado com cada ponto (CALAKLI; TAUBIN, 2011a). O método usa

uma função implícita f com valor 1 nos pontos no interior do modelo e valor 0 nos pontos externos e obtém a superfície reconstruída calculando a curva de nível zero da função. Segundo a formulação proposta pelos autores (KAZHDAN; BOLITHO; HOPPE, 2006a), existe uma relação entre o gradiente da função implícita ∇f e a integral do campo vetorial normal à superfície amostrada. O valor do gradiente será igual ao valor do vetor normal nos pontos sobre a superfície e igual a zero nas demais regiões. A Figura 5 exemplifica o modelo proposto. Neste contexto, o problema passa a ser a obtenção da função f cujo gradiente melhor aproxima o campo vetorial $\vec{V}$ definido nas amostras da nuvem de pontos de entrada, $min_f \|\nabla f - \vec{V}\|$ (KAZHDAN; BOLITHO; HOPPE, 2006a). Com a aplicação do operador de divergência este torna-se um problema de Poisson, calcular a função f cujo Laplaciano seja igual ao divergente do campo vetorial $\vec{V}$, $\Delta f \equiv \nabla^2 f = \nabla \cdot \vec{V}$.

Figura 5 – Algoritmo de Poisson - ilustração do modelo em 2D



Fonte: (KAZHDAN; BOLITHO; HOPPE, 2006a)

Segundo os autores, a abordagem aplicando Poisson traz algumas vantagens para a solução do problema de reconstrução. Entre elas estão a obtenção de uma solução global que considere todos os dados, a geração de superfícies suavizadas e a robustez para trabalhar em amostras com ruído (desalinhadas ou não uniformes). O método também consegue adaptar-se nas regiões com maior ou menor densidade dos pontos, através do cálculo do gradiente da função implícita. Por este motivo, diferente de outros métodos, o processo de reconstrução não gera elementos mal formados nas regiões mais distantes da concentração dos pontos do conjunto de entrada.

O objetivo do método, então, é a reconstrução da superfície ∂S, de um modelo desconhecido S, a partir de um conjunto de amostras de pontos $X = \{(x_1, \vec{n}_1), \cdots, (x_N, \vec{n}_N)\}$, onde x_i é um ponto na superfície e $\vec{n}_i$ é o vetor normal do ponto i com direção orientada para dentro do objeto. A formulação busca este objetivo através do cálculo de uma função indicadora f que aproxima os pontos do conjunto de entrada. Um dos pontos principais para obtenção da função f é o relacionamento entre seu gradiente ∇f e a integral do campo vetorial $\vec{V}$ normal à superfície S. Para formalização deste relacionamento, os autores apresentam a seguinte premissa: Dada uma superfície S com limite ∂S, f_S será sua função implícita, $\vec{N}_{\partial S}(x)$ o vetor normal no ponto $x \in \partial S$, $\tilde{F}(y)$ um filtro de suavização e $\tilde{F}_x(y) = \tilde{F}(y - x)$ sua translação para o ponto x. O gradiente da função será igual ao campo vetorial obtido pela suavização do campo normal à

superfície, conforme a equação abaixo:

$$\nabla(f_S * \tilde{F})(y_0) = \int_{\partial S} \tilde{F}(y_0)\vec{N}_{\partial S}(x)\mathrm{d}x$$

Usando o conjunto X para dividir ∂S em partes $P_X \subset \partial S$, a partir da definição acima, o algoritmo obtém a aproximação da integral para cada parte P_X (com as amostras do conjunto de entrada), conforme a equação que segue:

$$\nabla(f_S * \tilde{F})(y) = \sum_{(x,\vec{n})\in X} \int_{P_X} \tilde{F}_x(y)\vec{N}_{\partial S}(x)\mathrm{d}x \approx \sum_{(x,\vec{n})\in X} |P_X|\tilde{F}_x(y)\vec{n} \equiv \vec{V}(y)$$

O filtro de suavização $\tilde{F}$ deve satisfazer duas condições. Deve ser suficientemente restrito para não gerar um efeito de suavização em excesso e deve ser suficientemente amplo para que a integral sobre P_X tenha boa aproximação pelos valores dos pontos dimensionados pela área da partição. De acordo com o modelo, uma boa escolha é uma gaussiana cuja variância seja da ordem de resolução da amostragem. Por fim, a formulação define a função como um problema de Poisson a partir do campo vetorial $\vec{V}$ para resolver a função $\tilde{f}$ tal que $\nabla\tilde{f} = \vec{V}$. Para encontrar a solução é necessário obter a melhor aproximação, usando o operador de divergência para gerar a equação de Poisson: $\Delta\tilde{f} = \nabla \cdot \vec{V}$. No problema proposto, a solução pode ser reduzida a um sistema linear esparso bem condicionado.

O algoritmo estrutura os dados de entrada em uma octree adaptativa, definida sobre os pontos de acordo com sua localização no espaço. Para cada célula da divisão do conjunto de entrada na árvore é definida uma função para aproximação dos pontos e do campo vetorial como definido acima. Para gerar a superfície reconstruída o modelo usa uma variação própria do algoritmo Marching Cubes (LORENSEN; CLINE, 1987). O algoritmo possui diferentes parâmetros de entrada, entre todos, os mais relevantes são os seguintes: nível máximo da octree (valor padrão 8), que define a resolução do processo de reconstrução, onde profundidades maiores geram imagens com mais detalhes do modelo original, número mínimo de amostras (pontos) por célula da árvore (valor padrão 1), nível máximo completo (valor padrão 5), define a partir de qual nível a árvore passará para o modo adaptativo de acordo com a densidade dos pontos, nível de amostragem do voxel (valor padrão nível da árvore), define o tamanho da grade onde a função implícita será amostrada, nível de resolução do gradiente (valor padrão 0), define o nível até onde o método do gradiente conjugado será usado para resolver o sistema linear e escala (valor padrão $1,1$) define a relação entre o volume usado para reconstrução e o volume da amostra (KAZHDAN; BOLITHO; HOPPE, 2006b).

Os autores também comparam o método com outros algoritmos para reconstrução, com bons resultados visuais. Comparam o tempo de execução e o consumo de memória, mas não há uma comparação numérica do resultado das superfícies reconstruídas para avaliar a distância para o modelo original. No trabalho é destacado a adaptação do método para amostras com problemas de alinhamento e uniformidade dos dados, assim como a escalabilidade do método

para processos de reconstrução com maior resolução.

3.3.2 Smooth Signed Distance Surface Reconstruction

O método Smooth Signed Distance Surface Reconstruction é uma nova formulação para resolver o problema da reconstrução de superfícies a partir de uma nuvem de pontos orientados, através de uma função implícita desenvolvida por Calakli e Taubin (2011a). Esta abordagem é muito próxima daquela usada no método Poisson Surface Reconstruction, os dois métodos trabalham com a discretização da formulação contínua para reduzir a solução para um sistema de equações lineares esparsas. Neste método, diferente do método de Poisson, a função implícita é definida como uma aproximação suavizada da função de distância para a superfície estimada.

A formulação proposta processa a reconstrução de uma superfície S, definida por uma função implícita $S = \{x : f(x) = 0\}$, que aproxima um conjunto finito de pontos orientados $X = \{(x_1, \vec{n}_1), \cdots, (x_N, \vec{n}_N)\}$, onde x_i é um ponto na superfície e $\vec{n}_i$ é o vetor normal do ponto i com direção orientada para fora do objeto. A formulação considera $f(x) < 0$ dentro do objeto e $f(x) > 0$ fora do objeto. O gradiente de uma função com derivadas contínuas de primeira ordem gera um campo vetorial normal à superfície de nível desta função. O método usa esta propriedade para comparar o gradiente $\nabla f(x)$ da função $f(x)$ com o vetor normal dos pontos da nuvem e estimar a inclinação da superfície reconstruída naquele ponto. Como os pontos x_i são amostras da superfície S e os vetores $\vec{n}_i$ são amostras das normais da superfície nestes pontos, é possível definir uma forma de interpolação onde a função implícita deve satisfazer $f(x_i) = 0$ e $\nabla f(x_i) = \vec{n}_i$ para todos os pontos $i = 1, \cdots, N$ do conjunto de entrada. O método define, ainda, que estas duas condições sejam satisfeitas no sentido dos mínimos quadrados e o problema passa a ser considerado como a minimização de uma equação, definida abaixo.

$$\varepsilon_X(f) = \lambda_0 \varepsilon_{X_0}(f) + \lambda_1 \varepsilon_{X_1}(f) + \lambda_2 \varepsilon_R(f)$$

Onde

$$\varepsilon_{X_0}(f) = \frac{1}{N} \sum_{i=1}^{N} f(x_i)^2 \qquad \varepsilon_{X_1}(f) = \frac{1}{N} \sum_{i=1}^{N} \|\nabla f(x_i) - n_i\|^2$$

O terceiro elemento é um termo de regularização da equação, definido por $\varepsilon_R(f)$, calculado pela matriz Hessiana da função $f(x)$. Este termo determina como a função deve se comportar em regiões mais distantes dos pontos do conjunto de entrada, na região mais esparsa da imagem. Neste caso, o termo faz o gradiente da função tender a uma constante, evitando a geração de componentes mal formados. Os dois primeiros termos, $\varepsilon_{X_0}(f)$ e $\varepsilon_{X_1}(f)$ forçam a aproximação da função com a distância Euclidiana para a superfície estimada na região de maior densidade dos pontos. Na equação, λ_0, λ_1 e λ_2 são constantes positivas usadas para aumentar o peso de cada termo definidas como parâmetros de entrada do algoritmo.

O método executa, também, a divisão do conjunto de entrada em células organizadas em uma octree que dividem os pontos, baseado na sua localização, definidos por um volume V (um cubo que define a região de cada subconjunto de pontos). O modelo define uma série de discretizações da função $f(x)$, do gradiente $\nabla f(x)$ e da Hessiana $Hf(x)$ para cada célula, que reduz o problema a solução de um sistema linear. O algoritmo prevê um processo de solução iterativa, com uma solução inicial simples refinada a cada passo da iteração, usando como entrada de cada nível o resultado do nível anterior.

A parte final do modelo é a construção de uma aproximação poligonal da curva de nível zero da função implícita discretizada obtida nos passos anteriores, a reconstrução da superfície propriamente. O método usa a implementação do algoritmo Dual Marching Cubes (SCHAEFER; WARREN, 2004). O algoritmo possui diferentes parâmetros de entrada, além dos pesos atribuídos a cada termo da equação de minimização definidos acima (valor padrão 1): nível máximo da octree (valor padrão 8), tolerância do resultado do sistema linear iterativo (valor padrão $1,0e-8$) e número mínimo de amostras (pontos) por célula da árvore (valor padrão 1) (CALAKLI; TAUBIN, 2011b).

O método proposto é comparado com outros algoritmos atuais de reconstrução pelos autores, mostrando bons resultados nas superfícies geradas. Os resultados dos diferentes processos de reconstrução são comparados de forma numérica usando a distância de Hausdorff e a ferramenta Metro (CIGNONI; ROCCHINI; SCOPIGNO, 1998), avaliando a distância entre os modelos reconstruídos e as amostras de pontos de um modelo conhecido. Os autores também apontam bons resultados em nuvens de pontos com dados não uniformes, com ruído e desalinhados, quando comparado com outros métodos. Também destacam a característica do algoritmo adaptar-se para a densidade dos pontos na amostra e a definição de um modelo mais simples para implementação. Por fim, os autores desenvolveram uma variação do modelo para reconstrução de superfícies para nuvens de pontos com a informação do mapa de cores das amostras (CALAKLI; TAUBIN, 2012).

4 ALGORITMO PROPOSTO

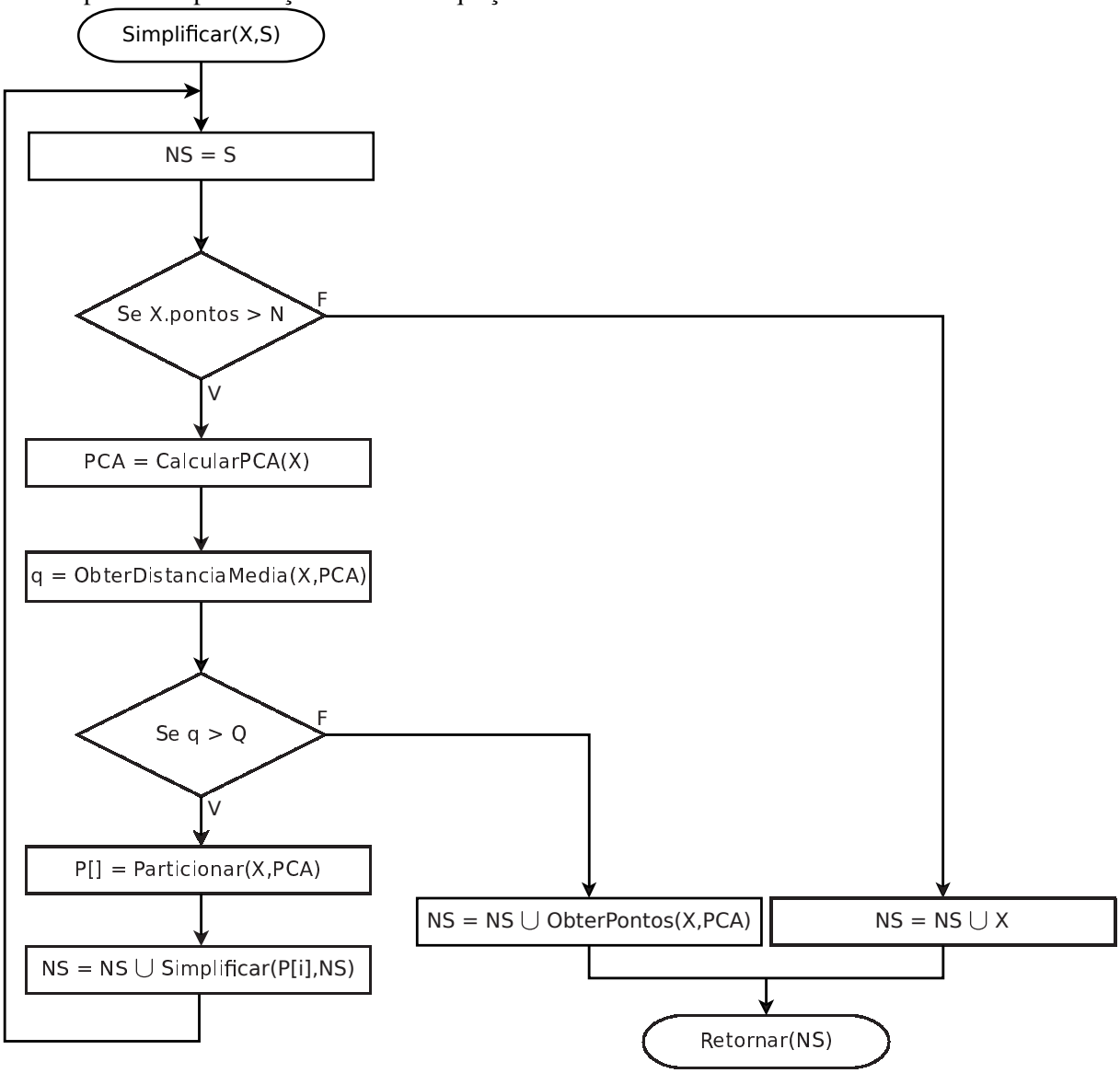
O algoritmo desenvolvido aplica a Análise de Componentes Principais para obter uma classificação estatística do conjunto de entrada, buscando uma forma da divisão com máxima independência linear. O objetivo é obter a aproximação linear dos pontos do conjunto, usando planos definidos pelas componentes principais, e dividir os pontos por esta aproximação. Para cada aproximação um parâmetro de qualidade é verificado, para avaliar se os pontos estão em um mesmo plano. Em caso negativo, o conjunto é particionado usando as componentes principais em um processo recursivo. O processo executado pelo algoritmo é descrito de forma geral no fluxograma da Figura 6 e no Algoritmo 1. A execução do algoritmo pode ser dividida em 5 etapas principais:

- Obter os autovalores, autovetores e o ponto médio do conjunto de entrada no processo da Análise de Componentes principais (linha 3).
- Obter a distância média dos pontos do conjunto de entrada ao plano definido pelas PCs (linha 4).
- Particionar, se for o caso, o conjunto de entrada a partir das PCs (linha 6).
- Chamada recursiva para para executar o algoritmo em cada novo particionamento do conjunto de pontos de entrada (linhas 7 a 9).
- Obter os pontos finais de cada partição ao final do processo de divisão do conjunto (linhas 11 e 14).

Os seguintes símbolos são usados na representação do algoritmo:

- X - Conjunto de pontos de entrada.
- S - Conjunto de pontos simplificado (usado como entrada no processo recursivo para formar o conjunto final, na primeira chamada será um conjunto vazio).
- N - Parâmetro de entrada definido abaixo. Quantidade mínima de pontos em uma partição.
- PCA - Estrutura de dados contendo os autovalores, autovetores e ponto médio do conjunto de pontos de entrada.
- Q - Parâmetro de entrada definido abaixo. Qualidade da aproximação linear.
- $P[]$ - Conjunto de partições obtidas da nuvem de entrada.
- NS - Novo conjunto de pontos simplificado.

Figura 6 – Fluxograma do processo recursivo de divisão da nuvem de pontos por planos de aproximação. A propriedade $X.pontos$ retorna a quantidade de pontos do conjunto X, o processo CalcularPCA(X) obtém a matriz de covariâncias, o ponto médio, os autovalores e autovetores do conjunto X, a função ObterDistanciaMedia(X, PCA) retorna a média das distâncias dos pontos do conjunto X ao plano usando as equações 4.3 e 4.4, a função Particionar(X, PCA) executa o processo de particionamento do conjunto X como descrito acima e a função ObterPontos(X, PCA) retorna os três pontos que representarão o plano de aproximação usando as equações 4.5 e 4.6.



Fonte: Imagem produzida pelo autor

O processo inicia obtendo uma aproximação linear para todo o conjunto de entrada. Obviamente, esta primeira aproximação será ruim mas, a partir desta, o algoritmo começará o processo de divisão da nuvem de pontos. A cada nova partição obtida, todo o processo é executado novamente, como descrito acima. A divisão do conjunto de entrada é executada de forma sucessiva até que a qualidade da aproximação atinja um limite mínimo definido pelo usuário (a qualidade é definida pela média das distâncias dos pontos ao seu plano de aproximação). Neste momento, o algoritmo fará a obtenção dos pontos que serão mantidos no conjunto simplificado.

Algoritmo 1 Simplificar

Entrada: X {Nuvem de pontos}, S {Nuvem de pontos simplificada}
Saída: NS {Nuvem de pontos simplificada}

```
1:  NS ← S
2:  se X.pontos > N então                          ▷ Verificar quantidade de pontos
3:      PCA ← CalcularPCA(X)
4:      q ← ObterDistanciaMedia(X,PCA)
5:      se q > Q então                             ▷ Verificar qualidade da aproximação
6:          P[] ← Particionar(X,PCA)
7:          para i ← 0 até Particoes(P[]) − 1 faça
8:              NS ← NS ∪ Simplificar(P[i], NS)    ▷ Chamada recursiva
9:          fim para
10:     senão
11:         NS ← NS ∪ ObterPontos(X,PCA)           ▷ Obter amostra de pontos
12:     fim se
13: senão
14:     NS ← NS ∪ X                                ▷ Obter pontos da partição
15: fim se
16: retornar NS
```

Além do parâmetro de qualidade, o usuário pode definir a quantidade mínima de pontos contidos em uma partição para que o algoritmo execute a obtenção da aproximação linear. Caso o processo chegue a este limite sem atingir a qualidade mínima esperada, todos os pontos desta partição serão mantidos no conjunto simplificado.

O algoritmo tem como entrada diferentes parâmetros que alteram seu comportamento, e têm como objetivo adaptar o processo ao conjunto de entrada e às necessidades do usuário. Desta forma, o usuário pode definir os parâmetros para tratar diferentes nuvens de pontos, definir maior ou menor qualidade no conjunto final e o impacto na taxa de simplificação. Os principais parâmetros são os seguintes:

- Q - Limite de qualidade da aproximação linear dos pontos. Atualmente é a média das distâncias de cada ponto ao plano (conforme as equações 4.3 e 4.4). Valores mais baixos indicam maior qualidade de aproximação e, portanto, devem gerar nuvens de pontos simplificadas com maior nível de detalhes e mais próximas do conjunto original. Contudo, o conjunto simplificado terá uma quantidade maior de pontos, diminuindo a taxa de compactação. Quando o processo de particionamento atinge este valor de qualidade a divisão do conjunto é interrompida e os 3 pontos representativos da aproximação linear são extraídos (conforme as equações 4.5 e 4.6). Valor padrão $0,01$.

- N - Quantidade mínima de pontos em um plano. Quando o processo de particionamento atinge esta quantidade de pontos o conjunto não é mais dividido, mesmo que a qualidade da aproximação linear (Q) não seja obtida. Neste caso, possivelmente a aproximação não está adequada e extrair apenas alguns pontos do conjunto pode trazer resultados ruins. Por isso, nesta situação, o usuário pode optar por manter todos os pontos da partição no

conjunto simplificado (maior qualidade no resultado final, menor taxa de simplificação) ou extrair apenas 3 (menor qualidade no resultado final, maior taxa de simplificação). Valor padrão 5.

- D - Tipo de divisão do conjunto de pontos de entrada para obtenção das aproximações lineares. Durante o processo de obtenção dos planos os pontos podem ser divididos de duas formas: ao meio ($D = 2$), através da primeira componente principal (associado ao maior λ) ou em quatro partes ($D = 4$), através das duas primeiras componentes principais (associadas aos dois maiores λ).

- P - Tipo de obtenção dos pontos que melhor representam o plano encontrado. Atualmente existem duas formas avaliadas: os pontos com menor distância ao plano de aproximação (equação 4.5) e os pontos mais distantes do ponto médio do plano de aproximação (equação 4.6).

- R - Retornar todos os pontos da região quando atingir a quantidade mínima definida pelo parâmetro N. Desta forma, quando a aproximação linear não for adequada (segundo a métrica escolhida), o algoritmo não fará a seleção de pontos. Valor padrão verdadeiro.

- N - Atualização do vetor normal para cada ponto que for mantido na nuvem de pontos. O vetor normal usado será aquele definido pela terceira componente principal em cada região obtida ao final do processo de particionamento. Valor padrão falso.

- PTS - Quantidade de pontos que serão usados para representar uma região aproximada por um plano. Altera a definição de 3 pontos obtidos por aproximação linear. Valor padrão 3.

As seções 4.1 (função $CalcularPCA(X)$), 4.2 (função $ObterMediaDistancia(X, PCA)$), 4.3 (função $Particionar(X, PCA)$) e 4.4 (função $ObterPontos(X, PCA)$) apresentam cada etapa do processo. A seção 4.5 mostra a métrica usada para comparação dos resultados.

4.1 Aproximação Linear (PCA)

A base principal do algoritmo é a obtenção dos autovalores e autovetores do conjunto de pontos de entrada (para cada uma das chamadas recursivas do processo de particionamento). No modelo, a matriz de covariâncias S é obtida a partir dos valores das variáveis x, y, z, coordenadas de cada ponto do conjunto de entrada X no espaço, definidas no R^3, de acordo com a matriz na equação 4.1. Os valores das variâncias e covariâncias (σ) são calculados conforme a equação 3.2.

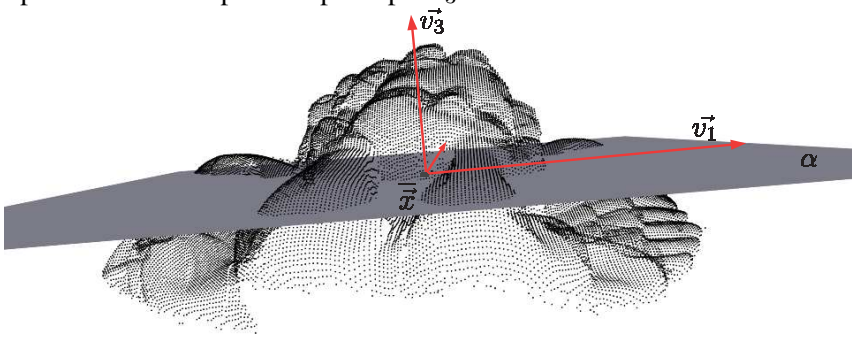
$$S_{(3x3)} = \begin{bmatrix} \sigma_{xx} & \sigma_{xy} & \sigma_{xz} \\ \sigma_{yx} & \sigma_{yy} & \sigma_{yz} \\ \sigma_{zx} & \sigma_{zy} & \sigma_{zz} \end{bmatrix} \tag{4.1}$$

A matriz de covariâncias $S_{(3x3)}$ será simétrica e invertível, possuindo as propriedades listadas na seção 3.1. Desta forma, a matriz de covariâncias obtida a partir do conjunto de entrada X possui três autovalores: $\lambda_1 \geq \lambda_2 \geq \lambda_3 \geq 0$. Associados a cada autovalor temos, respectivamente, três autovetores ortogonais: $\vec{v_1}, \vec{v_2}, \vec{v_3}$. O valor de λ representa a variância dos pontos na direção do seu respectivo autovetor. Sendo assim, tomando uma nuvem de pontos no espaço tridimensional, os autovetores $\vec{v_1}$ e $\vec{v_2}$ apontam para as direções de maior variância dos dados. Por consequência, o vetor $\vec{v_3}$ aponta para direção de menor variância. Dada esta definição, a melhor aproximação linear dos pontos do conjunto de entrada será aquela definida pelo plano gerado pelos dois primeiros autovetores ($\vec{v_1}$ e $\vec{v_2}$).

A partir desta definição, é possível obter a equação do plano que melhor aproxima um conjunto de pontos no espaço. A equação pode ser definida pelo ponto médio $\overline{\vec{x}}$ do conjunto de entrada e o vetor normal ao plano que passa por este ponto que, neste caso, será o terceiro autovetor ($\vec{v_3}$). Desta forma, os pontos que pertencem ao plano podem ser descritos como aqueles que satisfazem a equação 4.2. O autovetor $\vec{v_3}$ de $S_{(3\times3)}$ é normal ao plano α gerado por $\vec{v_1}$ e $\vec{v_2}$, este plano é aquele que minimiza a soma do quadrado das distâncias dos pontos a ele. A Figura 7 mostra um plano de aproximação em uma nuvem de pontos de entrada de acordo com a definição aplicando PCA.

$$(\vec{x} - \overline{\vec{x}}) \cdot \vec{v_3} = 0 \tag{4.2}$$

Figura 7 – Melhor aproximação dos pontos de um conjunto de entrada, definido pelas componentes principais. Esta aproximação é definida pelo plano α, que é gerado pelo ponto médio $\overline{\vec{x}}$ e pelo vetor normal indicado pela terceira componente principal $\vec{v_3}$.



Fonte: Imagem produzida pelo autor

4.2 Qualidade da Aproximação

O processo de obtenção dos planos que melhor aproximam regiões do conjunto de pontos de entrada começa obtendo um único plano α para todo o conjunto. Caso a qualidade da aproximação do plano aos pontos não esteja adequada, o conjunto é dividido em subconjuntos de pontos a partir das componentes principais. A métrica de qualidade da aproximação é definida pela média das distâncias dos pontos ao plano, de acordo com a equação 4.3. Onde $\vec{x_k}$ são os

n pontos do conjunto de entrada X considerados para a aproximação e $d(\vec{x_k}, \alpha)$ a distância de cada ponto ao plano calculada pela equação 4.4.

$$Q = \frac{1}{n} \sum_{k=1}^{n} d(\vec{x_k}, \alpha) \tag{4.3}$$

$$d(\vec{x_k}, \alpha) = |(\vec{x_k} - \vec{\overline{x}}) \cdot \vec{v_3}| \tag{4.4}$$

Na equação 4.4, $\vec{\overline{x}}$ é o ponto médio (baricentro) do conjunto de dados de entrada. São usados os pontos médios de cada aproximação, ou seja, de cada nova região obtida durante o processo de particionamento. O Algoritmo 2 apresenta o cálculo da distância média dos pontos de uma região ao seu plano de aproximação conforme as equações. No processo de aproximação esta distância é comparada com o parâmetro Q para definir a qualidade esperada pelo usuário.

Algoritmo 2 ObterDistanciaMedia

Entrada: X {Nuvem de pontos}, PCA {Estrutura com os dados da PCA}
Saída: D {Distância média dos pontos ao plano}

1: $s \leftarrow 0$
2: **para cada** $x \in X$ **faça**
3: $\quad d \leftarrow \text{ABS}((x - PCA.\overline{x}) \cdot PCA.v_3)$ ▷ Distância sem sinal
4: $\quad s \leftarrow s + d$
5: **fim para**
6: $D \leftarrow s \ / \ X.pontos$ ▷ Média das distâncias
7: **retornar** D

4.3 Particionamento do Conjunto de Entrada

Após avaliar se a qualidade da aproximação definida na seção anterior é adequada (pelo parâmetro informado pelo usuário), o algoritmo deve decidir pelo particionamento, ou não, do conjunto de entrada. Este particionamento será feito usando as componentes principais e a posição dos pontos no espaço em relação aos planos formados por elas e o ponto médio do conjunto. A divisão do conjunto de pontos em novas células pode ser feita usando dois critérios possíveis, conforme abaixo:

- Divisão de X em duas novas células de pontos (conjunto dividido ao meio através da primeira componente principal):

$$x_k \in X_1, \ se \ (\vec{x_k} - \vec{\overline{x}}) \cdot \vec{v_1} \geq 0$$

$$x_k \in X_2, \ se \ (\vec{x_k} - \vec{\overline{x}}) \cdot \vec{v_1} < 0$$

- Divisão de X em quatro novas células de pontos (conjunto dividido em 4 partes através

das primeira e segunda componentes principais):

$$x_k \in X_1,\ se\ (\vec{x_k} - \vec{\overline{x}}) \cdot \vec{v_1} \geq 0\ E\ (\vec{x_k} - \vec{\overline{x}}) \cdot \vec{v_2} \geq 0$$

$$x_k \in X_2,\ se\ (\vec{x_k} - \vec{\overline{x}}) \cdot \vec{v_1} \geq 0\ E\ (\vec{x_k} - \vec{\overline{x}}) \cdot \vec{v_2} < 0$$

$$x_k \in X_3,\ se\ (\vec{x_k} - \vec{\overline{x}}) \cdot \vec{v_1} < 0\ E\ (\vec{x_k} - \vec{\overline{x}}) \cdot \vec{v_2} < 0$$

$$x_k \in X_4,\ se\ (\vec{x_k} - \vec{\overline{x}}) \cdot \vec{v_1} < 0\ E\ (\vec{x_k} - \vec{\overline{x}}) \cdot \vec{v_2} \geq 0$$

A lógica de particionamento do conjunto conforme os critérios acima está representada nos algoritmos 3 e 4, a forma usada na divisão do conjunto de pontos é definida pelo parâmetro P de entrada do processo. Após a divisão do conjunto de entrada X em novos conjuntos de pontos, o processo de obtenção da matriz de covariâncias $S_{(3\times3)}$, a obtenção dos autovalores λ, autovetores $\vec{v_\lambda}$ e a verificação da qualidade da aproximação dos pontos pelos novos planos é executada para cada novo conjunto X_i obtido. O processo é repetido de forma recursiva até que a qualidade da aproximação esteja adequada conforme o parâmetro Q ou a quantidade de pontos do conjunto resultante do processo de divisão, para cada célula, chegue ao limite definido no parâmetro N.

Algoritmo 3 Particionar (Divisão em 2)

Entrada: X {Nuvem de pontos}, PCA {Estrutura com os dados da PCA}
Saída: $P[]$ {Partições da nuvem de pontos}

```
1: DEFINIRTAMANHO(P[],2)                    ▷ Definir lista com duas partições
2: para cada x ∈ X faça
3:     d ← (x − PCA.x̄) · PCA.v1             ▷ Distância do ponto ao plano
4:     se d ≥ 0 então                       ▷ Posição do ponto em relação ao plano
5:         P[0].inserir(x)                  ▷ Definir partição
6:     senão
7:         P[1].inserir(x)                  ▷ Definir partição
8:     fim se
9: fim para
10: retornar P[]
```

A Figura 8 exemplifica o processo de aproximação em um conjunto de pontos por segmentos de reta em um espaço bidimensional, de acordo com o algoritmo proposto. No exemplo bidimensional existem apenas duas variáveis (x, y), então, serão gerados apenas dois autovalores e dois autovetores a partir da matriz de covariâncias. As retas vermelhas mostram a direção da primeira componente principal no conjunto de dados, as retas segmentadas azuis mostram a direção da segunda componente principal e na interseção das retas está o ponto médio do conjunto. A primeira componente é a reta de melhor aproximação e a segunda componente é usada para o particionamento do conjunto de dados de entrada. O mesmo princípio é usado no espaço tridimensional. A Figura 9 mostra o primeiro passo da divisão de uma nuvem de pontos de entrada em duas células (usando a primeira componente principal como vetor normal do

Algoritmo 4 Particionar (Divisão em 4)

Entrada: X {Nuvem de pontos}, PCA {Estrutura com os dados da PCA}
Saída: P {Partições da nuvem de pontos}

1: DEFINIRTAMANHO($P[]$,4) ▷ Definir lista com quatro partições
2: **para cada** $x \in X$ **faça**
3: $d1 \leftarrow (x - PCA.\overline{x}) \cdot PCA.v_1$ ▷ Distância do ponto ao 1º plano
4: $d2 \leftarrow (x - PCA.\overline{x}) \cdot PCA.v_2$ ▷ Distância do ponto ao 2º plano
5: **se** $(d1 \geq 0)$ E $(d2 \geq 0)$ **então** ▷ Posição do ponto em relação aos 2 planos
6: $P[0].inserir(x)$ ▷ Definir partição
7: **fim se**
8: **se** $(d1 \geq 0)$ E $(d2 < 0)$ **então** ▷ Posição do ponto em relação aos 2 planos
9: $P[1].inserir(x)$ ▷ Definir partição
10: **fim se**
11: **se** $(d1 < 0)$ E $(d2 < 0)$ **então** ▷ Posição do ponto em relação aos 2 planos
12: $P[2].inserir(x)$ ▷ Definir partição
13: **fim se**
14: **se** $(d1 < 0)$ E $(d2 \geq 0)$ **então** ▷ Posição do ponto em relação aos 2 planos
15: $P[3].inserir(x)$ ▷ Definir partição
16: **fim se**
17: **fim para**
18: **retornar** $P[]$

plano usado para particionamento) e quatro células (usando as primeira e segunda componentes principais como vetor normal dos planos usados para particionamento do conjunto de dados).

4.4 Obtenção dos Pontos

Quando o processo de divisão do conjunto de pontos é encerrado e o algoritmo obtém o plano que melhor aproxima uma região da nuvem, são extraídos 3 pontos que serão usados para representar todos os pontos aproximados por este plano. A obtenção destes pontos pode ser feita de duas formas, conforme um parâmetro de entrada do processo: os 3 pontos mais próximos de cada plano de aproximação (equação 4.5) ou os 3 pontos mais distantes do ponto médio do conjunto aproximado por cada plano (equation 4.6).

$$Min(|(\vec{x_i} - \vec{\overline{x}}) \cdot \vec{v_3}|) \tag{4.5}$$

$$Max(\sqrt{(\vec{x_{i,x}} - \vec{\overline{x_x}})^2 + (\vec{x_{i,y}} - \vec{\overline{x_y}})^2 + (\vec{x_{i,z}} - \vec{\overline{x_z}})^2}) \tag{4.6}$$

Os algoritmos 5 e 6 mostram a forma de obtenção dos pontos dentro do conjunto de entrada. Nos algoritmos é usado o parâmetro PTS, definido pelo usuário, que determina a quantidade de pontos que deve ser retornada por partição.

A forma de seleção dos pontos é bastante simples, adotando como único critério sua posição em relação ao plano de aproximação de cada região. Este critério certamente não é o mais

Figura 8 – Exemplo em 2D da aproximação de um conjunto de pontos por segmentos de reta aplicando o algoritmo proposto. A imagem (a) mostra o conjunto de pontos de entrada, as imagens (b) a (h) mostram o processo recursivo de divisão do conjunto de entrada em segmentos de reta e a imagem (i) mostra o conjunto de pontos resultantes do processo de simplificação. Note que o processo de divisão é adaptativo, onde existe uma curvatura maior, mais divisões e mais retas são usadas para a aproximação e um número maior de pontos é mantido no conjunto final.

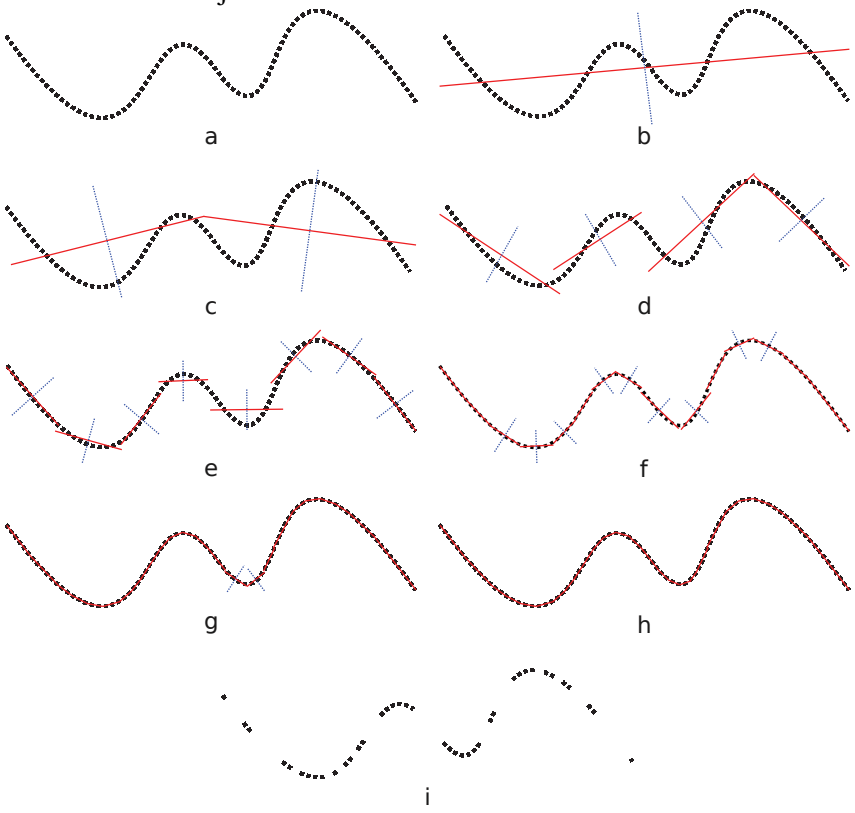
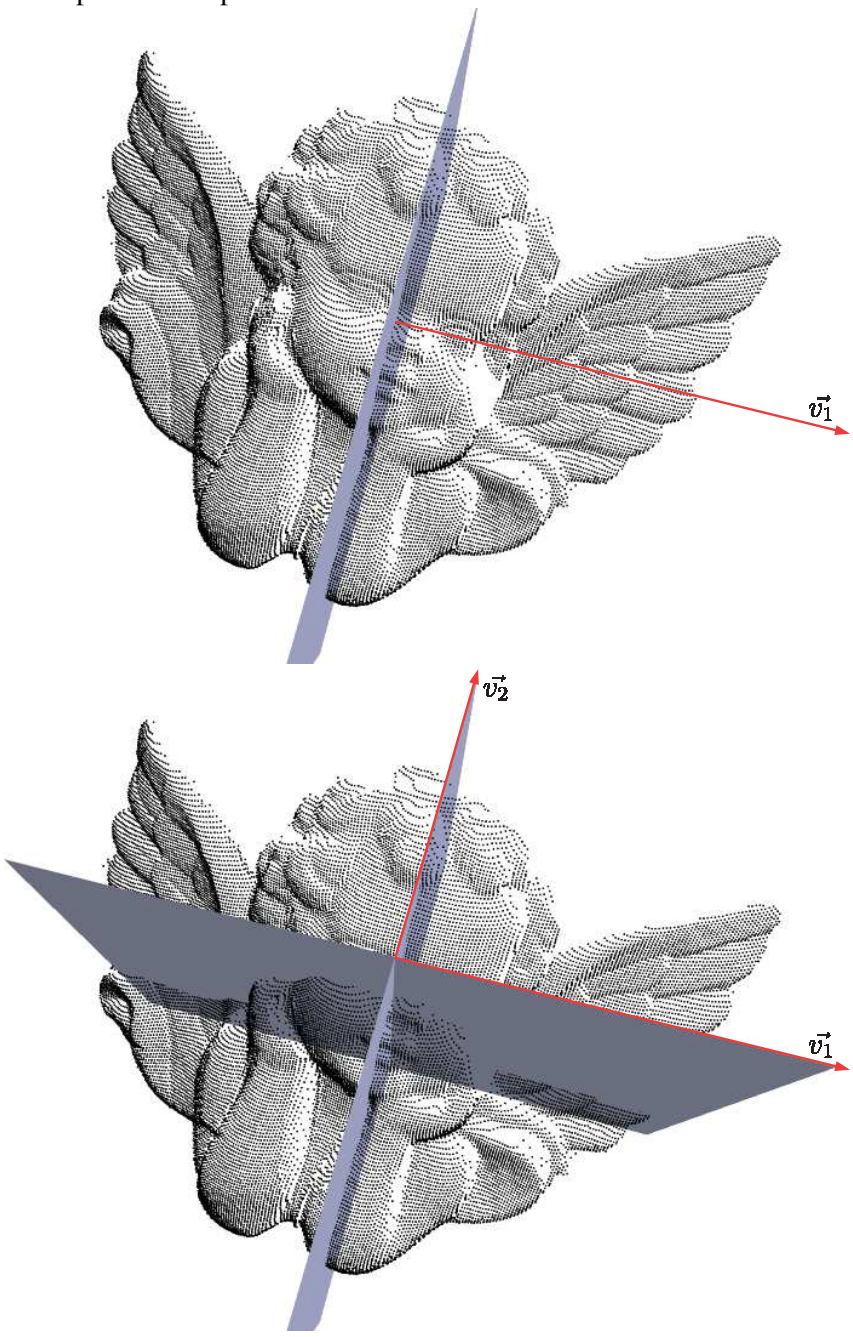


Figura 9 – Divisão do conjunto de pontos por planos. A imagem mostra o primeiro passo da divisão do conjunto em duas partes (figura superior) e em quatro partes (figura inferior). Na divisão em duas partes a primeira componente principal é o vetor normal ($\vec{v_1}$) do plano usado para particionamento e na divisão em quatro partes as primeira e segunda componentes principais são os vetores normais ($\vec{v_1}$ e $\vec{v_2}$) dos planos usados no processo de particionamento.



Fonte: Imagem produzida pelo autor

que permite a comparação entre duas malhas de polígonos e a medição numérica de sua diferença. A motivação principal do desenvolvimento da ferramenta foi a necessidade de avaliar a diferença entre uma superfície e sua versão após a execução de um processo de simplificação. Originalmente a ferramenta foi desenvolvida para comparar os resultados de algoritmos usados para simplificação de malhas de polígonos. A Figura 10 traz exemplos de uma malha de

Algoritmo 5 ObterPontos (Equação 4.5)

Entrada: X {Nuvem de pontos}, PCA {Estrutura com os dados da PCA}
Saída: S {Nuvem com os pontos selecionados}

para cada $x \in X$ **faça**
  $d \leftarrow \text{ABS}((x - PCA.\overline{x}) \cdot PCA.v_3)$ ▷ Distância do ponto ao plano sem sinal
  se $S.pontos < PTS$ **então** ▷ Definir primeiro conjunto de resultados
    $S.inserir(x)$
    $D.inserir(d)$
  senão ▷ Manter no conjunto aqueles com menor distância
    $max \leftarrow 0$
    para i $\leftarrow 1$ **até** $PTS - 1$ **faça**
      se $D[i] > D[max]$ **então**
        $max \leftarrow i$
      fim se
    fim para
    se $d < D[max]$ **então**
      $S[max] \leftarrow x$
      $D[max] \leftarrow d$
    fim se
  fim se
fim para
retornar S

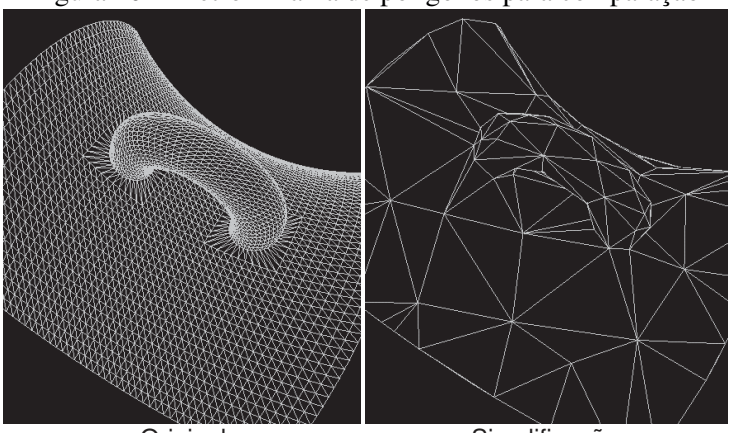
Algoritmo 6 ObterPontos (Equação 4.6)

Entrada: X {Nuvem de pontos}, PCA {Estrutura com os dados da PCA}
Saída: S {Nuvem com os pontos selecionados}

para cada $x \in X$ **faça**
  $d \leftarrow \text{SQRT}((x_x - PCA.\overline{x}_x)^2 + (x_y - PCA.\overline{x}_y)^2 + (x_z - PCA.\overline{x}_z)^2)$ ▷ Distância do ponto ao ponto médio
  se $S.pontos < PTS$ **então** ▷ Definir primeiro conjunto de resultados
    $S.inserir(x)$
    $D.inserir(d)$
  senão ▷ Manter no conjunto aqueles com maior distância
    $min \leftarrow 0$
    para i $\leftarrow 1$ **até** $PTS - 1$ **faça**
      se $D[i] < D[min]$ **então**
        $min \leftarrow i$
      fim se
    fim para
    se $d > D[min]$ **então**
      $S[min] \leftarrow x$
      $D[min] \leftarrow d$
    fim se
  fim se
fim para
retornar S

polígonos e sua simplificação.

Figura 10 – Metro - malha de polígonos para comparação



Fonte: (CIGNONI; ROCCHINI; SCOPIGNO, 1998)

Na ferramenta, a aproximação do erro entre duas malhas é definido como a distância entre as seções correspondentes de cada malha. A medida da distância é definida como:

$$e(p, S) = \min_{p' \in S} d(p, p'),$$

Onde p é um dado ponto qualquer, S uma superfície e d a distância Euclidiana entre dois pontos no espaço. A partir da definição acima, a distância, não simétrica, entre duas superfícies S_1 e S_2 é definida como:

$$E(S_1, S_2) = \max_{p \in S_1} e(p, S_2)$$

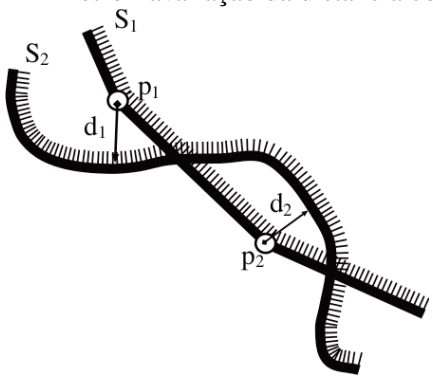
A definição da distância entre duas superfícies não é simétrica, então, $E(S_1, S_2) \neq E(S_2, S_1)$. O método define a métrica como sendo a distância de Hausdorff pela obtenção do valor máximo entre $E(S_1, S_2)$ e $E(S_2, S_1)$. O algoritmo da ferramenta também obtém a distância média entre duas superfícies, definida como a integral da distância em razão da área da superfície:

$$E_m(S_1, S_2) = \frac{1}{|S_1|} \int_{S_1} e(p, S_2) \mathrm{d}s$$

Quando as superfícies S_1 e S_2 possuem orientação, ou seja, quando um ponto p sobre a superfície possui a informação do vetor normal à ela, é possível obter a posição relativa de uma malha com relação a outra. Desta forma, o valor da distância passa a ter sinal, que indicará quando um ponto está localizado na parte interna ou externa da superfície. A Figura 11 mostra um exemplo da avaliação da distância entre duas superfícies (S_1 como base da comparação), neste caso a distância é positiva para p_1 e negativa para p_2.

A ferramenta recebe como entrada duas superfícies, a primeira é usada como base (ou pivô),

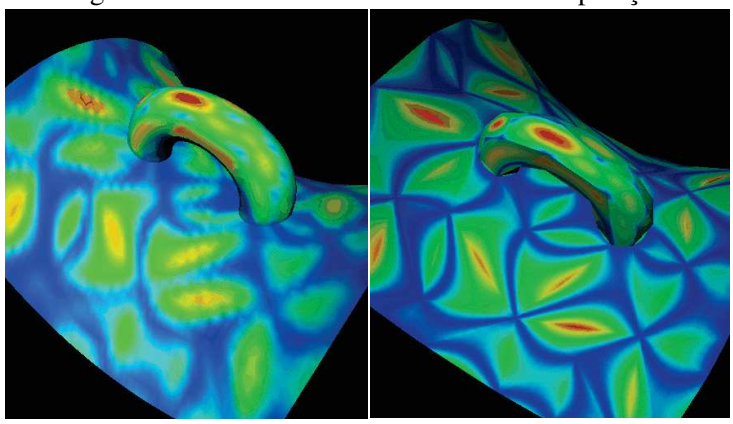
Figura 11 – Metro - avaliação da distância com sinal



Fonte: (CIGNONI; ROCCHINI; SCOPIGNO, 1998)

e obtém a aproximação do erro através de um processo de amostragem e o cálculo das distâncias de cada ponto até a superfície. O processo de amostragem é feito obtendo cada polígono da malha de entrada de acordo com uma resolução definida como parâmetro de entrada pelo usuário. O processo também permite o uso de uma aproximação onde um número aleatório de pontos é selecionado dentro de cada polígono (proporcional a sua área) para comparação das distâncias (abordagem de Montecarlo). Existem outros parâmetros que definem o comportamento do algoritmo de comparação entre as malhas, que pode considerar os vértices, arestas e faces dos polígonos. Como saída, a ferramenta mostra os resultados numéricos e uma comparação visual, com um mapeamento de cores para representar o erro em cada região da superfície. A Figura 12 mostra o resultado visual para as duas formas de mapeamento possíveis. A primeira forma mapeada pela distância de cada vértice e a segunda com o mapeamento do erro por face, obtido pelos pontos amostrados em cada polígono.

Figura 12 – Metro - resultado visual da comparação



Por vértices Por faces

Fonte: (CIGNONI; ROCCHINI; SCOPIGNO, 1998)

5 RESULTADOS

Um aplicativo foi desenvolvido para testes e avaliação dos resultados com a aplicação do algoritmo. O protótipo foi codificado na linguagem C++, usando a biblioteca Eigen (EIGEN LIBRARY, 2015) para álgebra linear e o conjunto de ferramentas OpenSceneGraph (OSG) (OPENSCENEGRAPH, 2015) para tratamento dos arquivos. O aplicativo possui diferentes argumentos de entrada para alterar o comportamento do algoritmo e adequar o processo às necessidades do usuário e ao conjunto de entrada. Para obtenção dos resultados foram usados diferentes conjuntos de pontos como entrada do processo e a reconstrução do modelo em 3D foi executada com os algoritmos SSD e Poisson para comparação. Os algoritmos de reconstrução foram executados com profundidade da árvore igual a 8 (gerando uma resolução máxima no grid de 2^8 x 2^8 x 2^8).

Trabalhando com os valores padrão foi obtido, no melhor caso, um novo conjunto de dados com aproximadamente 5% dos pontos de entrada. Os valores padrão foram definidos ao longo dos testes, de acordo com os resultados apresentados (quantidade de pontos e qualidade da imagem resultante do processo de reconstrução). Os testes foram realizados usando para reconstrução da superfície os algoritmos Smooth Signed Distance Surface Reconstrucion (SSD) e Poisson Surface Reconstruction, com diferentes modelos de entrada (Tabela 1). O modelo Angel foi obtido do trabalho que apresenta o método SSD para reconstrução de superfícies (CALAKLI; TAUBIN, 2011a). Os modelos Anchor, Dancing Children, Daratech, Gargoyle e Quasimoto foram obtidos do trabalho de avaliação de reconstrução de superfícies desenvolvido por Berger et al. (2013). Estes modelos foram desenvolvidos especificamente para os testes de reconstrução realizados pelos autores, buscando gerar diferentes exigências para os algoritmos de reconstrução. A amostragem dos pontos foi feita de forma a reproduzir o processo de captura por laser, reproduzindo as propriedades encontradas neste tipo de captura: amostragem não uniforme, dados com ruído e leituras desalinhadas. Por fim, o arquivo Bunny foi obtido do trabalho desenvolvido por (KAZHDAN; BOLITHO; HOPPE, 2006a) com o algoritmo de Poisson para reconstrução de superfícies. Todos os modelos estão no formato PLY (Stanford Polygon File Format), são nuvens de pontos orientados, cada arquivo possui as coordenadas dos pontos (x, y, z) e a informação do vetor normal de cada ponto (nx, ny, nz). Cada modelo apresenta uma forma complexa, com o objetivo de representar diferentes características e topologias (regiões planas, regiões curvas, cantos, diferentes níveis de detalhes, etc).

Os primeiros testes foram executados com o modelo Angel, com a reconstrução executada pelo algoritmo SSD, para validação e análise preliminar dos resultados. As Tabela 2 apresenta os resultados dos testes que tiveram melhor qualidade da imagem gerada ao final do processo de reconstrução (neste caso a qualidade foi definida apenas pela análise visual do resultado) e com quantidade de pontos final menor ou igual a 50% do total dos pontos de entrada. Para os resultados apresentados foi aplicado variação nos valores dos parâmetros Q ($0,01$ e $0,015$), D (2 e 4) e P (tipo de obtenção dos pontos finais). O parâmetro n não sofreu alteração (valor

Tabela 1 – Modelos de entrada para simplificação da nuvem

Modelo	**Pontos**	**Tamanho**
Anchor	263.286	17MB
Angel	24.566	1,2MB
Dancing Children	468.022	29MB
Daratech	245.836	16MB
Gargoyle	481.358	31,3MB
Quasimoto	350.000	22,5MB
Bunny	362.271	21MB

Fonte: Elaborada pelo autor

contante $n = 5$). Os testes foram executados em um computador com processador Intel Core i5-3337U, com 4 x 1,80 Ghz, 6 Gb de memória e sistema operacional GNU/Linux Ubuntu 14.04, 64 bits.

Tabela 2 – Resultados do processo de simplificação e reconstrução da superfície (Angel)

	Original	**q=0.01 D=2**	**q=0.01 D=4**	**q=0.015 D=2**	**q=0.015 D=4**
Pontos	24.566	9.471	11.963	6.685	8.979
Taxa	-	∼61%	∼51%	∼72%	∼63%
Tempo	-	∼0,043s	∼0.029s	∼0.038s	∼0.027s
SSD	∼6,729s	∼4,954s	∼5,388s	∼4,603s	∼5,084s
Tempo Total	∼6,729s	∼4,997s	∼5,417s	∼4,641s	∼5,111s

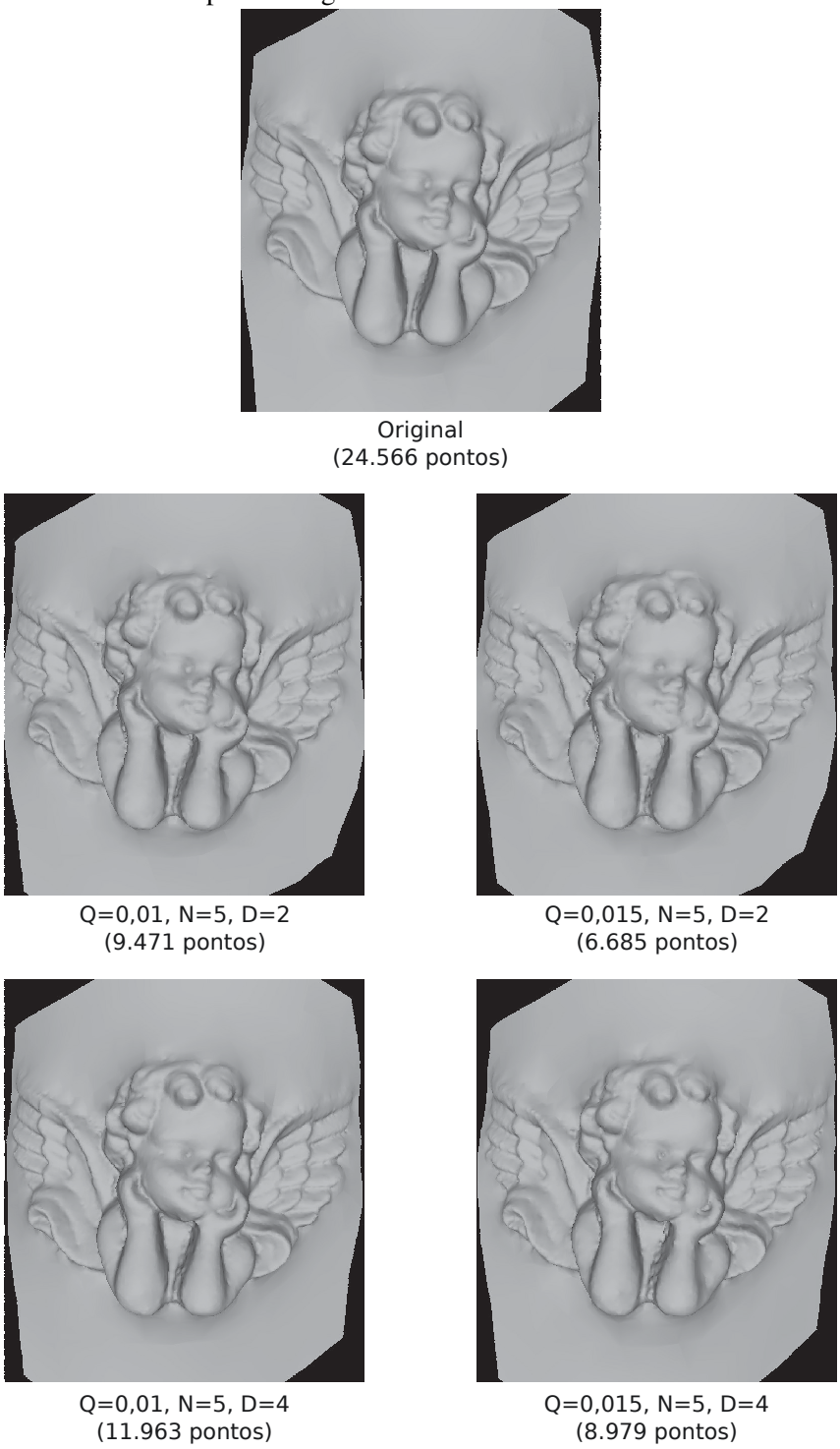
Fonte: Elaborada pelo autor

As Figuras 13 e 14 mostram as imagens resultantes dos processos de reconstrução. Na Figura 13 os resultados foram obtidos usando os pontos com menor distância ao plano de aproximação e na Figura 14 os resultados foram obtidos usando os pontos mais distantes do ponto médio do plano de aproximação (variação do parâmetro P).

Percebe-se nas imagens obtidas da reconstrução a maior qualidade visual daquelas onde a nuvem de pontos foi simplificada usando a divisão do conjunto de entrada em quatro partes, embora o arquivo resultante tenha maior número de pontos quando comparado com aqueles gerados com a divisão em duas partes. Este resultado é consequência da melhor distribuição dos planos no conjunto de pontos, gerando melhor aproximação (menor distância média) dos pontos aos planos. Como era esperado, a medida que o parâmetro Q é alterado, gerando um grau maior ou menor de exigência para a qualidade da aproximação, a quantidade final de pontos na nuvem simplificada também será alterada. Quanto maior o grau de exigência, mais pontos serão mantidos no conjunto de dados final e a imagem resultante do processo de reconstrução terá maior qualidade. Obviamente o contrário também acontece. O principal desafio está justamente em conseguir uma alta taxa de simplificação, com pontos que representem bem a superfície e, por consequência, gerem uma imagem de boa qualidade ao final do processo de reconstrução.

Após os testes iniciais para avaliação do algoritmo, o processo foi aplicado nas demais nuvens de pontos listadas na Tabela 1 usando os dois diferentes algoritmos para reconstrução. Para

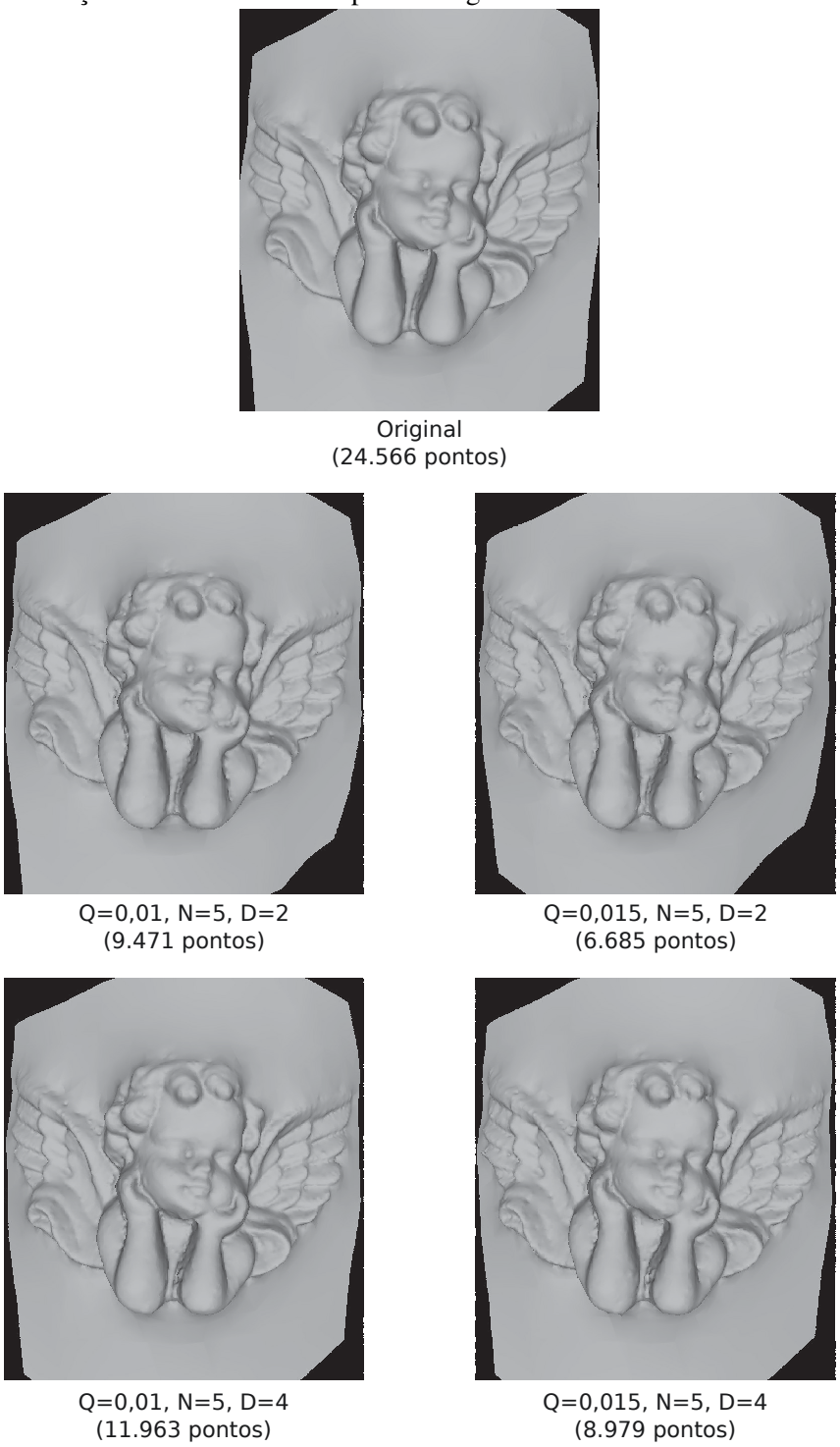
Figura 13 – Resultados do processo de reconstrução (Angel) usando diferentes valores para os parâmetros Q e D. Pontos com menor distância ao plano de aproximação. Comparação com o resultado da reconstrução usando a nuvem de pontos original.



Fonte: Imagem produzida pelo autor

avaliação dos resultados foi usada a ferramenta Metro (CIGNONI; ROCCHINI; SCOPIGNO, 1998) para calcular a distância de Hausdorff entre as imagens reconstruídas após a simplificação da nuvem e as imagens reconstruídas com os conjuntos de pontos originais. A distância de

Figura 14 – Resultados do processo de reconstrução (Angel) usando diferentes valores para os parâmetros Q e D. Pontos com a maior distância para o ponto médio do plano de aproximação. Comparação com o resultado da reconstrução usando a nuvem de pontos original.



Fonte: Imagem produzida pelo autor

Hausdorff define uma métrica formal para avaliar a diferença entre as superfícies reconstruídas, valores mais próximos de 0 indicam malhas de polígonos mais semelhantes.

A Tabela 3 apresenta os resultados para o modelo Quasimoto. O modelo possui 350.000

pontos e $22,5$MB de tamanho, o processo de reconstrução com o algoritmo SSD foi executado em ∼23,7s e a reconstrução com o algoritmo de Poisson foi executado em ∼19,3s. Os processos de reconstrução com os dois algoritmos foram executados com os parâmetros configurados com os valores padrão. O processo de simplificação foi executado com os parâmetros $N = 5$, divisão do conjunto de dados em 4 subconjuntos usando as duas primeiras componentes principais ($D = 4$), obtendo os 3 pontos mais próximos do plano de aproximação do conjunto, variando o parâmetro Q conforme a tabela. Os testes foram executados em um computador com processador Intel Core i5-3337U, com 4 x 1,80 Ghz, 6 Gb de memória e sistema operacional GNU/Linux Ubuntu 14.04, 64 bits.

Tabela 3 – Resultados do processo de simplificação e reconstrução da superfície (Quasimoto)

Q	**Original**	**0,01**	**0,02**	**0,04**
Pontos	350.000	64.396	33.032	16.730
Tamanho	22,5MB	3,6MB	1,9MB	0,95MB
Taxa	-	∼81%	∼90%	∼95%
Tempo	-	∼0,3s	∼0,3s	∼0,2s
SSD	∼23,7s	∼11,3s	∼9,2s	∼7,5s
Hausdorff	-	∼0,641	∼0,502	∼0,375
Poisson	∼19,3s	∼13,1s	∼10,1s	∼6,9s
Hausdorff	-	∼0,446	∼0,433	∼0,282

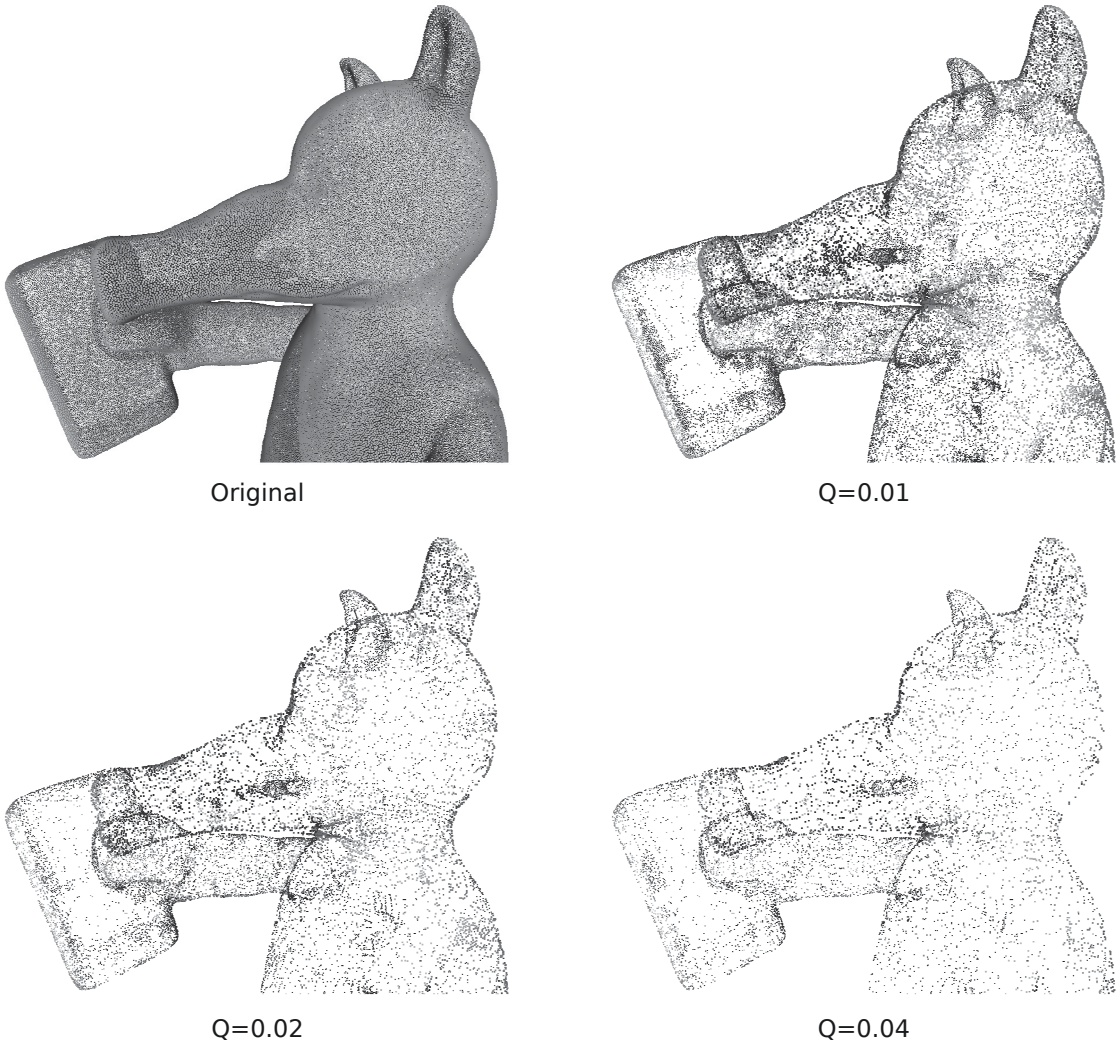
Fonte: Elaborada pelo autor

A Figura 15 mostra a nuvem de pontos original e as simplificações com $Q = 0,01$, $Q = 0,02$ e $Q = 0,04$. Na imagem é possível perceber a distribuição não uniforme dos pontos após o processo de simplificação, resultado da forma de divisão dos pontos usando as componentes principais. Nas regiões onde existe maior número de detalhes na superfície e os pontos não podem ser aproximados de forma consistente por um plano, mais divisões são geradas no processo. Desta forma, um maior número de planos serão gerados, resultando em uma quantidade maior de pontos nestas áreas da superfície.

As Figuras 16 e 17 mostram os resultados da reconstrução das superfícies para cada caso (superfície reconstruída com a nuvem original e com as nuvens de pontos simplificadas). Visualmente é possível perceber que não há perda da qualidade comparando a reconstrução do conjunto original com aqueles resultantes do processo de simplificação, tanto para o SSD, quanto para Poisson. Este resultado é percebido mesmo quando $Q = 0,04$ (nível de exigência de qualidade mais baixo), resultando em um conjunto com ∼5% da quantidade de pontos original. A similaridade das imagens também é comprovada pela métrica de Hausdorff, que gerou um bom valor de aproximação para os dois métodos de reconstrução.

O algoritmo foi testado com outros dois modelos de formas similares, Anchor e Daratech. Estes modelos trazem formas geométricas mais definidas, grandes regiões planas, sulcos, túneis, cantos e cantos arredondados. As Tabelas 4 e 5 mostram os resultados obtidos. O modelo Anchor possui 263.286 pontos e $17,0$MB de tamanho, o processo de reconstrução com o algoritmo SSD foi executado em ∼30,1s e a reconstrução com o algoritmo de Poisson foi executado

Figura 15 – Nuvem de pontos do modelo Quasimoto



Fonte: Imagem produzida pelo autor

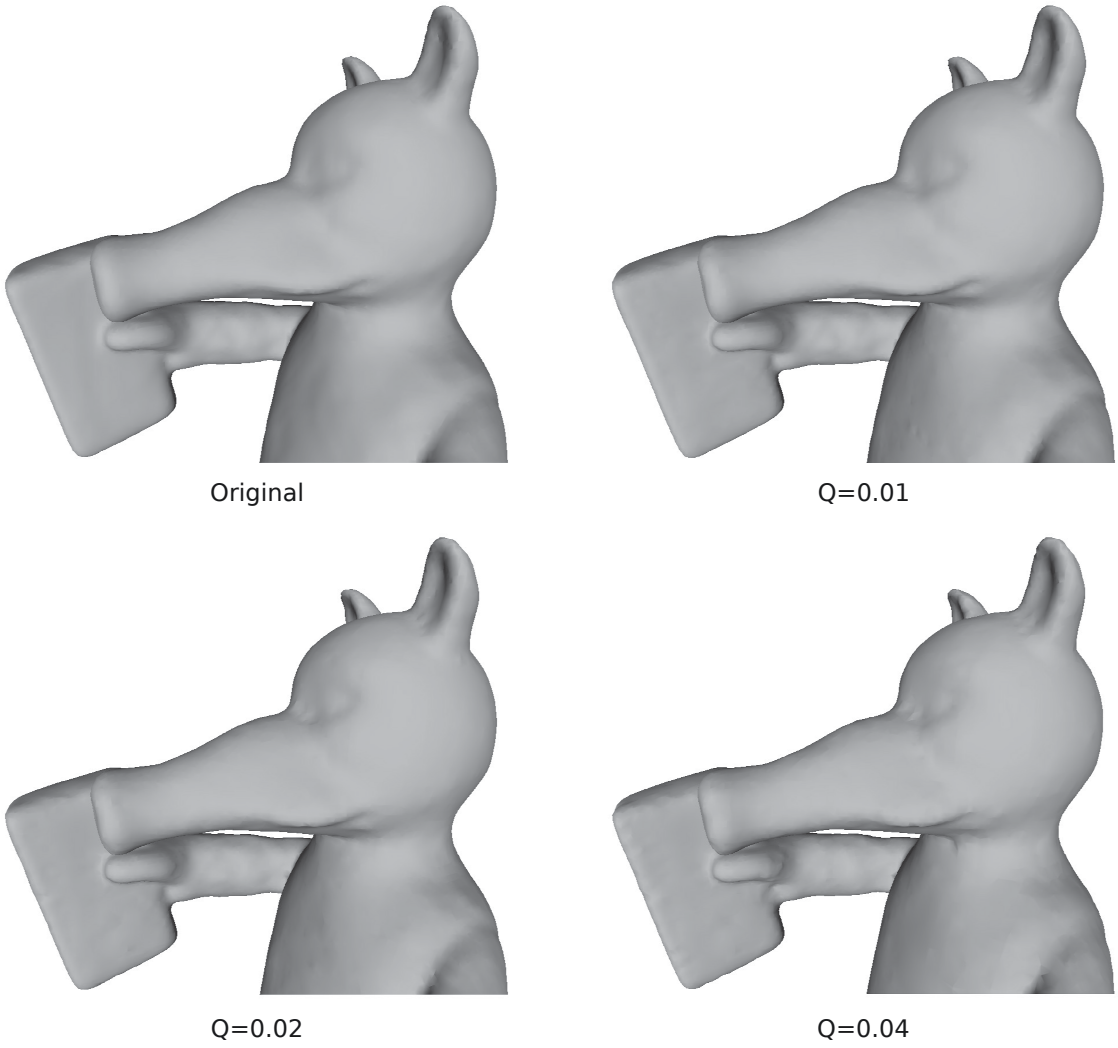
em ∼29,4s. O modelo Daratech possui 245.836 pontos e $16,0$MB de tamanho, o processo de reconstrução com o algoritmo SSD foi executado em ∼25,4s e a reconstrução com o algoritmo de Poisson foi executado em ∼17,3s.

Tabela 4 – Resultados do processo de simplificação e reconstrução da superfície (Anchor)

Q	**Original**	**0,01**	**0,04**
Pontos	263.286	38.969	23.628
Tamanho	17MB	2,2MB	1,4MB
Taxa	-	∼85%	∼91%
Tempo	-	∼0,2s	∼0,2s
SSD	∼30,1s	∼14,7s	∼12,7s
Hausdorff	-	∼1,077	∼1,179
Poisson	∼29,4s	∼13,8s	∼10,2s
Hausdorff	-	∼2,499	∼2,623

Fonte: Elaborada pelo autor

Figura 16 – Resultados do processo de reconstrução SSD (Quasimoto)



Fonte: Imagem produzida pelo autor

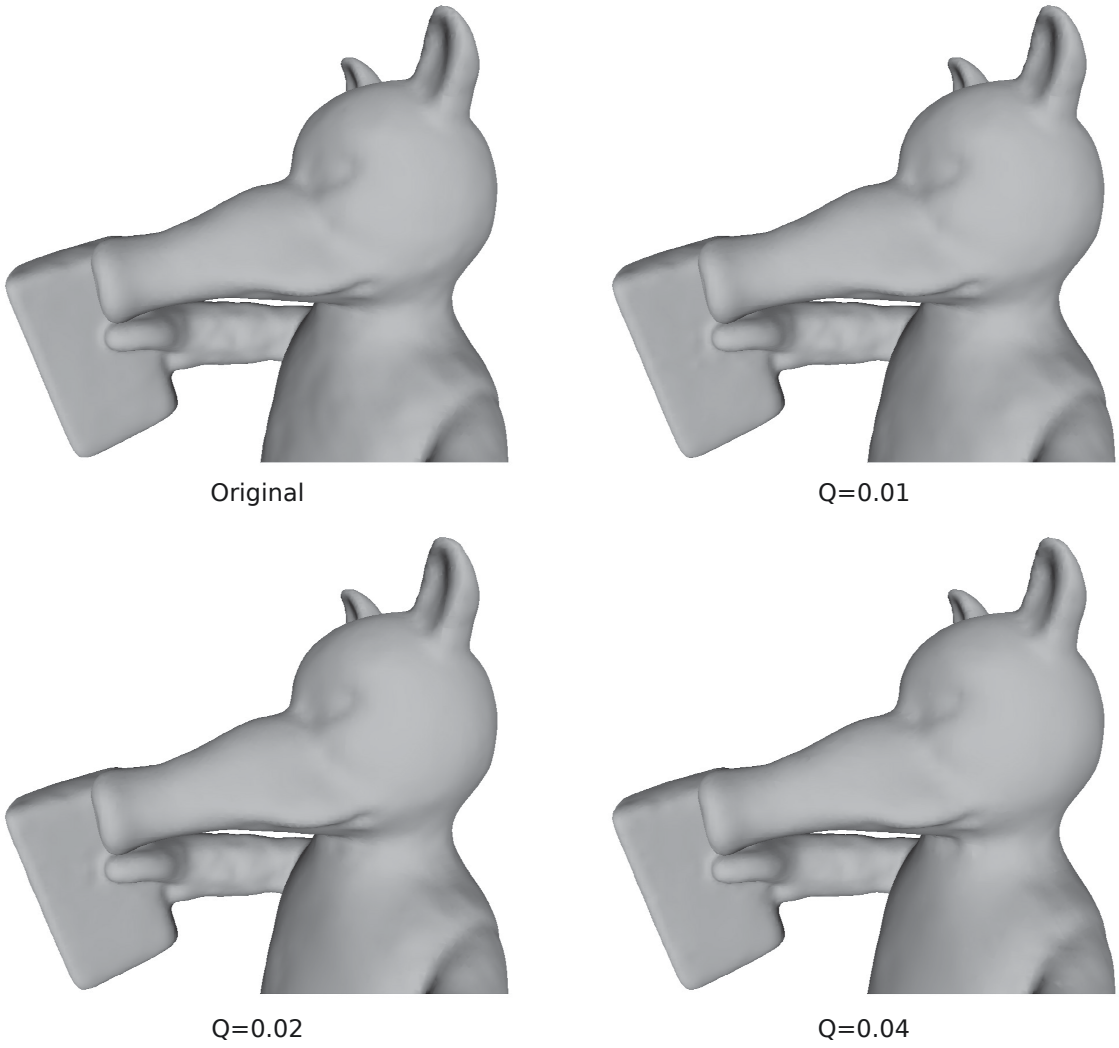
Tabela 5 – Resultados do processo de simplificação e reconstrução da superfície (Daratech)

Q	**Original**	**0,01**	**0,04**
Pontos	245.836	73.351	36.358
Tamanho	16MB	4,2MB	2,1MB
Taxa	-	∼70%	∼85%
Tempo	-	∼0,3s	∼0,2s
SSD	∼25,4s	∼12,8s	∼10,0s
Hausdorff	-	∼0,802	∼0,709
Poisson	∼17,3s	∼12,2s	∼10,5s
Hausdorff	-	∼0,900	∼0,814

Fonte: Elaborada pelo autor

O processo de simplificação para os dois modelos foi executado com os parâmetros $N = 5$, $D = 4$, obtendo os 3 pontos mais próximos do plano de aproximação do conjunto, variando o

Figura 17 – Resultados do processo de reconstrução Poisson (Quasimoto)



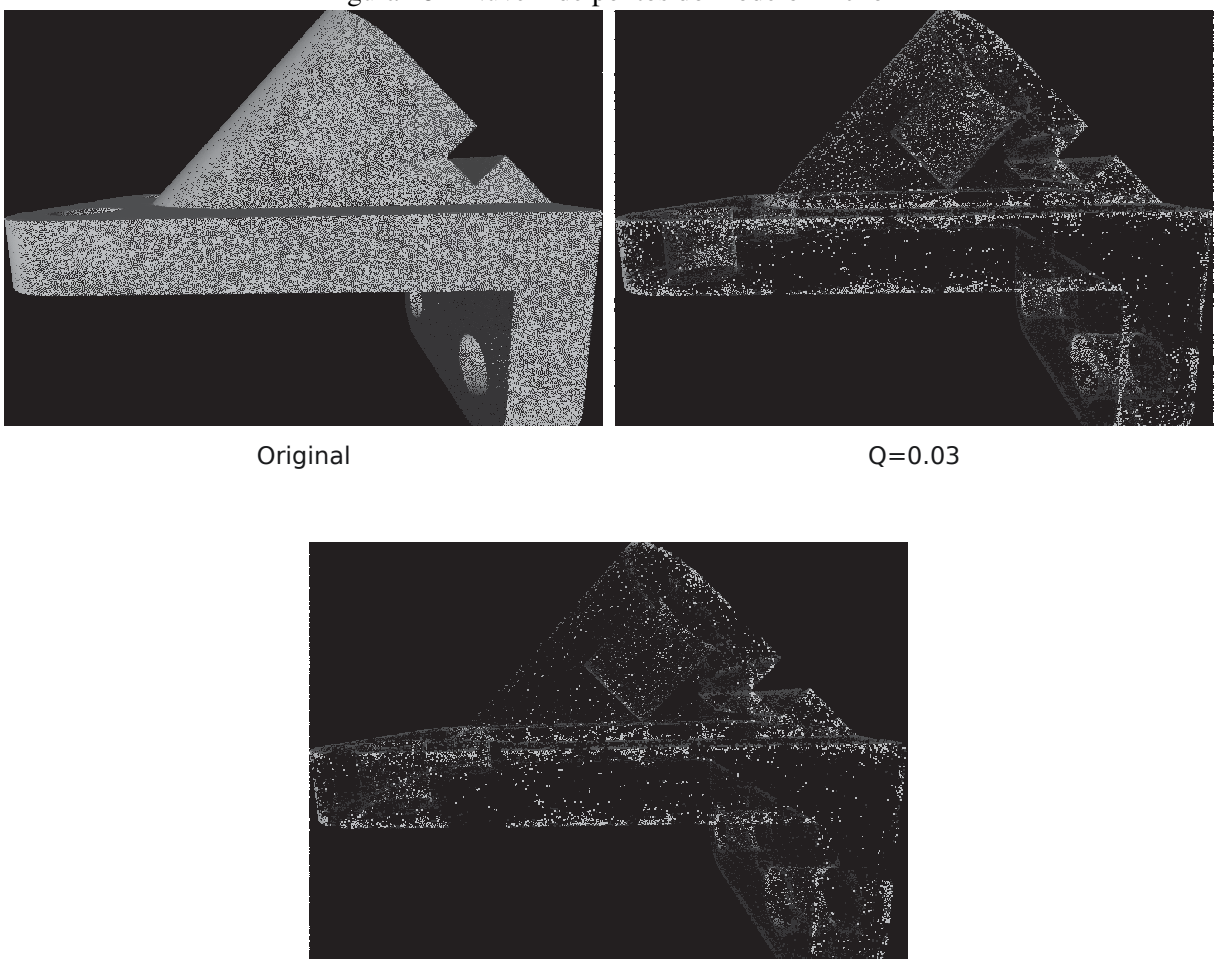
Fonte: Imagem produzida pelo autor

parâmetro Q com os valores $0,01$ e $0,04$, conforme as tabelas. Os testes foram executados em um computador com processador Intel Core i5-3337U, com 4 x 1,80 Ghz, 6 Gb de memória e sistema operacional GNU/Linux Ubuntu 14.04, 64 bits. As Figuras 18 e 19 mostram os conjuntos de pontos resultantes do processo de simplificação para os modelos Anchor e Daratech respectivamente.

Os resultados do processo de reconstrução para os dois modelos, com os algoritmos SSD e Poisson, podem ser vistos nas Figuras 20, 21, 22 e 23. Nas figuras é possível perceber a baixa qualidade das imagens especialmente nas regiões mais planas, nas bordas das superfícies e nas áreas com superfícies planas paralelas. Este limite de qualidade também pode ser verificado pela distância de Hausdorff obtida entre as imagens originais e as imagens geradas a partir das nuvens simplificadas. Desta forma, estes dois modelos trouxeram um desafio maior para o desempenho do algoritmo de simplificação.

Nas Figuras 20 e 21 é possível perceber, para os algoritmos SSD e Poisson, o problema

Figura 18 – Nuvem de pontos do modelo Anchor



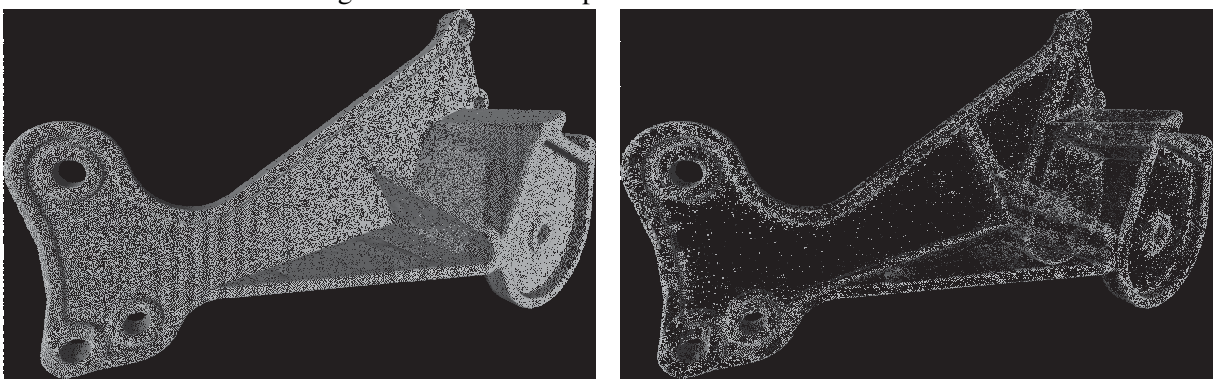
Fonte: Imagem produzida pelo autor

nas bordas das superfícies e a distorção nas regiões planas. Este é o resultado da remoção dos pontos nestas regiões e a manutenção de poucos pontos nas áreas planas, verificado nas nuvens de pontos da Figura 18, após a execução do processo de simplificação.

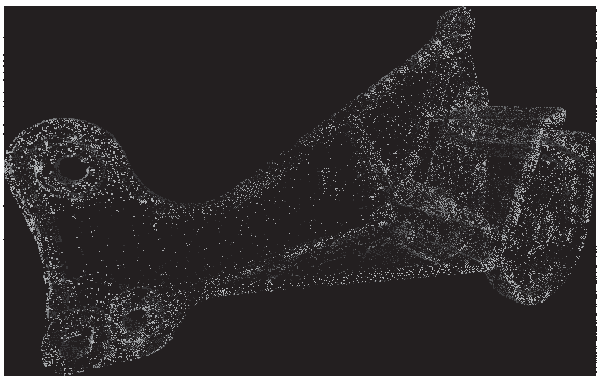
Novamente, nas Figuras 22 e 23, é possível perceber problemas no processo de reconstrução para os dois algoritmos. Inclusive existem "buracos"na superfície reconstruída, resultado das nuvens de pontos simplificadas que geraram conjuntos não uniformes e desalinhados. Neste modelo, o principal problema parece ser nas superfícies paralelas existentes na nuvem de pontos. Nestas regiões o algoritmo aproxima, em algumas partes, por um só plano pontos que estão em superfícies diferentes. Desta forma, o algoritmo de reconstrução não tem elementos suficientes para reconstruir as superfícies de forma adequada, e trata os pontos como se fossem pertencentes a uma só face do objeto.

Estes problemas podem ser atribuídos à forma simples de escolha dos pontos que representam cada aproximação local, pela falta de uniformidade dos pontos e pelo ruído gerado no processo de simplificação. Por isso, é necessário explorar algumas variações do algoritmo apresentado, como novas formas de obtenção dos pontos do conjunto final simplificado, correções

Figura 19 – Nuvem de pontos do modelo Daratech



Original Q=0.01



Q=0.04

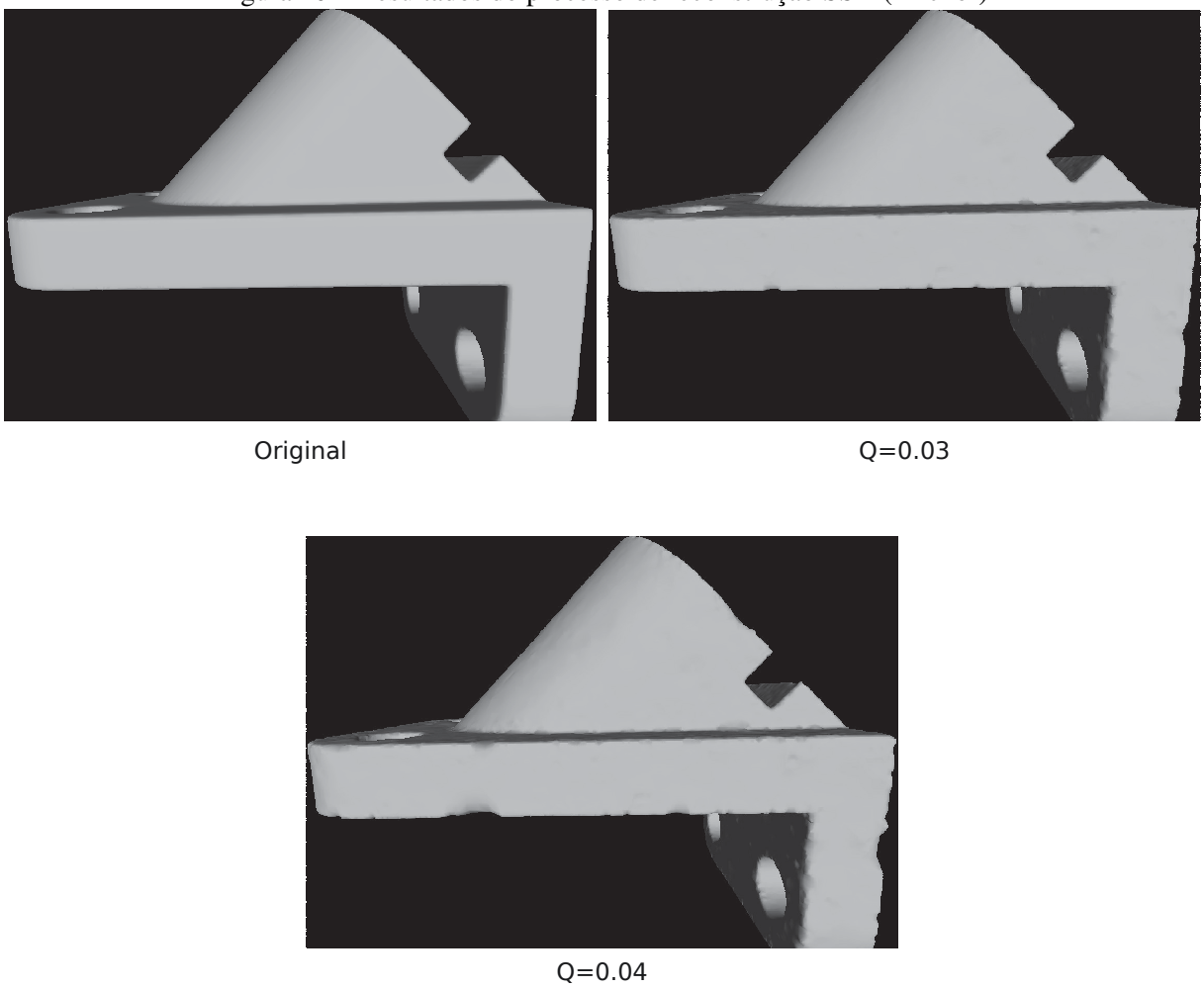
Fonte: Imagem produzida pelo autor

nas propriedades dos pontos mantidos na nuvem, em especial a atualização do vetor normal, critérios diferentes para pontos nas bordas da superfície e manter maior densidade de pontos em grandes áreas regulares da superfície amostrada.

A Tabela 6 apresenta os resultados para o modelo Gargoyle. Este modelo apresenta diferentes níveis de detalhes na superfície, exigindo um maior número de pontos em diferentes regiões para representação da superfície. Esta característica aumenta a dificuldade do processo de simplificação e limita o resultado do algoritmo para a remoção de pontos. Em comparação, o modelo Quasimoto é um exemplo de figura com partes articuladas, como os braços, pernas e cabeça, contudo, possui uma superfície mais uniforme.

O modelo Gargoyle possui 481.358 pontos e $31,3$MB de tamanho, o processo de reconstrução com o algoritmo SSD foi executado em $\sim$18,9s e a reconstrução com o algoritmo de Poisson foi executado em $\sim$14,6s. O processo de simplificação foi executado com os parâmetros $N = 5$, $D = 4$, obtendo os 3 pontos mais próximos do plano de aproximação do conjunto, variando o parâmetro Q conforme a tabela. Os testes foram executados em um computador com processador Intel Core i5-3337U, com 4 x 1,80 Ghz, 6 Gb de memória e sistema operacional Microsft Windows 8.1, 64 bits.

Figura 20 – Resultados do processo de reconstrução SSD (Anchor)



Fonte: Imagem produzida pelo autor

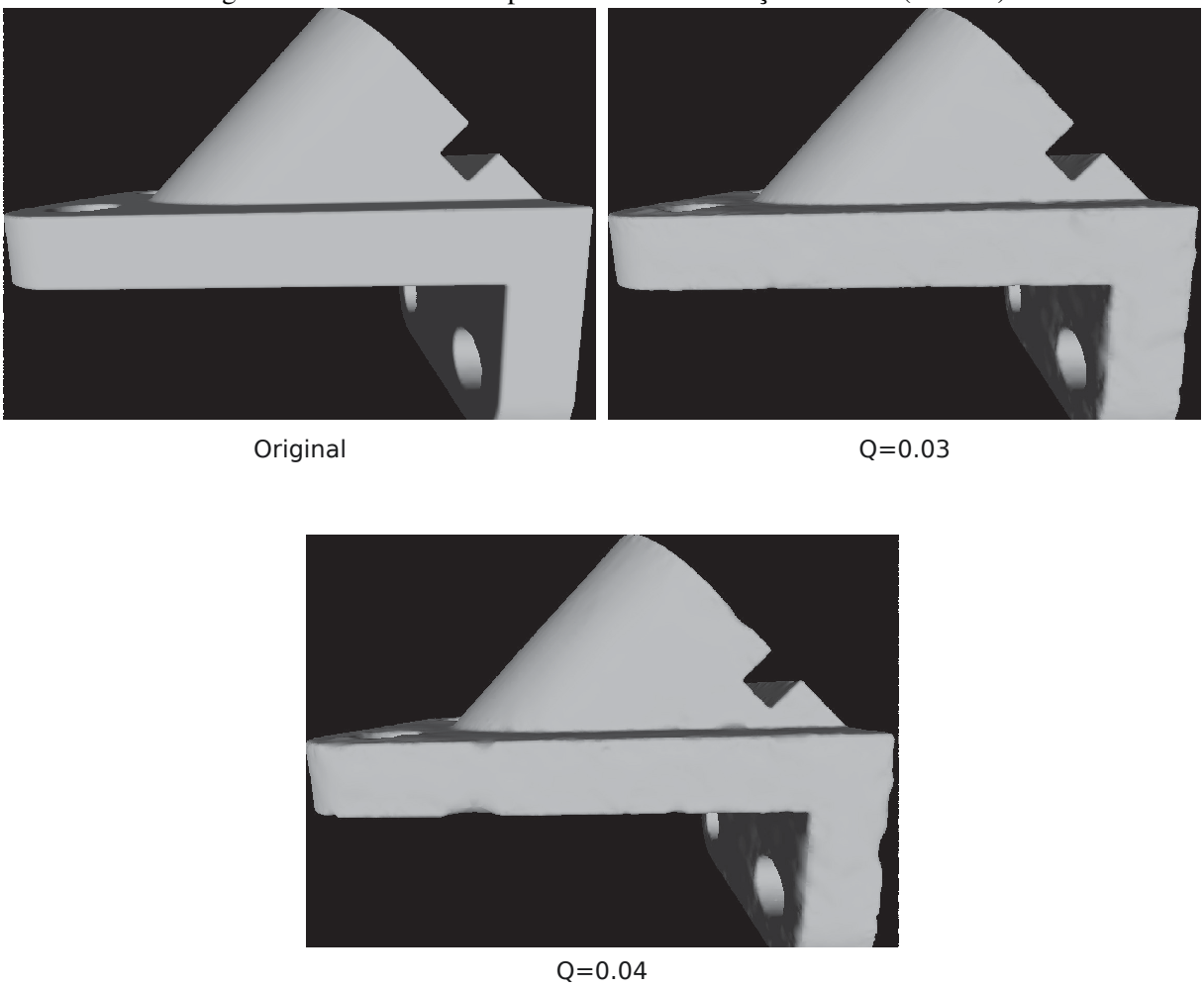
Tabela 6 – Resultados do processo de simplificação e reconstrução da superfície (Gargoyle)

Q	**Original**	**0,01**	**0,02**	**0,04**
Pontos	481.358	195.303	117.807	61.845
Tamanho	31,3MB	11,5MB	6,9MB	3,6MB
Taxa	-	∼59%	∼66%	∼82%
Tempo	-	∼0,6s	∼0,5s	∼0,4s
SSD	∼18,9s	∼8,9s	∼7,4s	∼5,5s
Hausdorff	-	∼0,584	∼0,461	∼0,380
Poisson	∼14,6s	∼9,8s	∼8,4s	∼6,5s
Hausdorff	-	∼0,228	∼0,582	∼1,367

Fonte: Elaborada pelo autor

A Figura 24 mostra a nuvem de pontos original e aquelas geradas pelo processo de simplificação. É possível perceber a menor taxa de remoção de pontos, quando comparado ao modelo Quasimoto. Uma quantidade maior de pontos foi mantida na nuvem para representar áreas com maior nível de detalhes, especialmente nas asas, na cabeça e na base da gárgula. Como mencionado acima, este modelo limita o resultado do algoritmo, com $Q = 0,01$ o processo de

Figura 21 – Resultados do processo de reconstrução Poisson (Anchor)



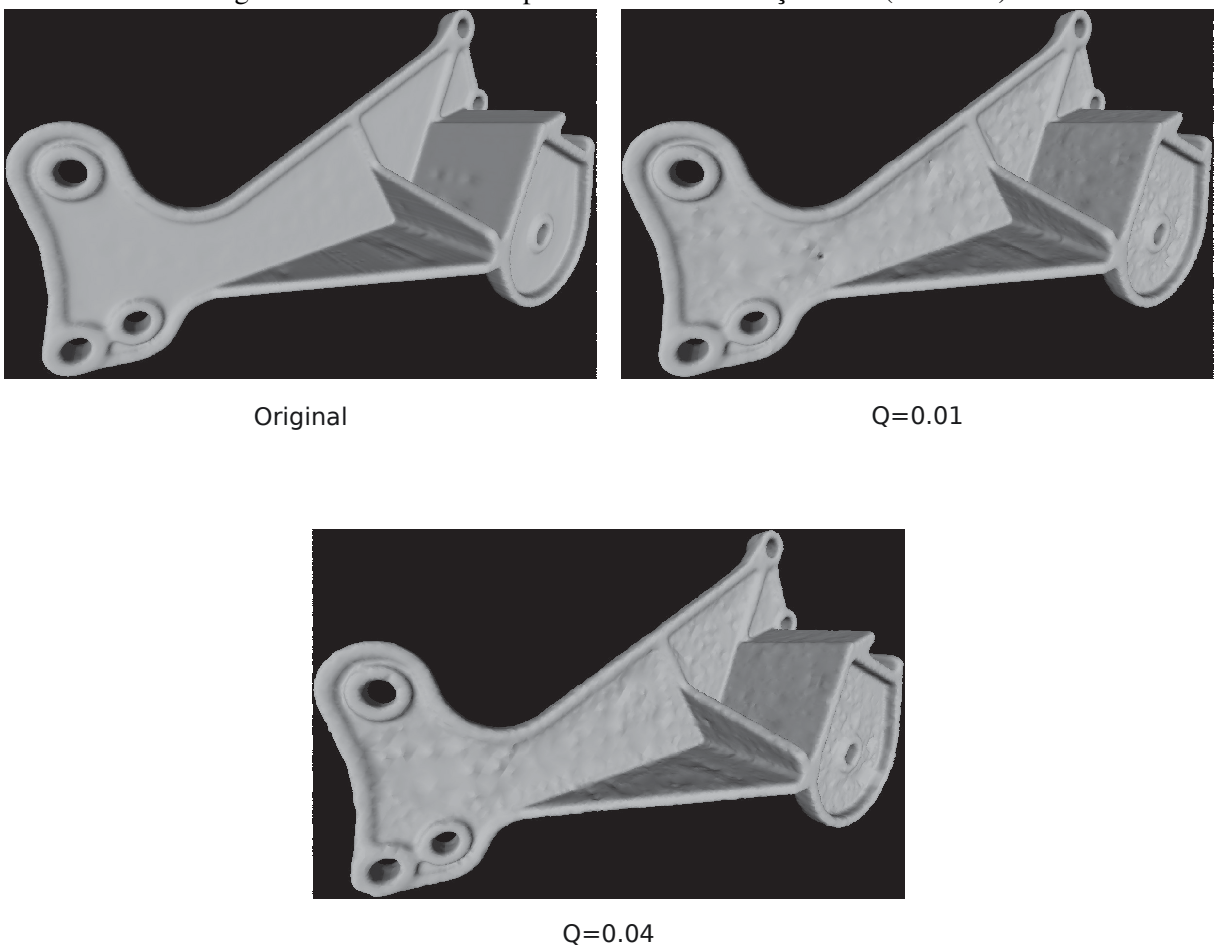
Fonte: Imagem produzida pelo autor

simplificação manteve $\sim$40% dos pontos. Com o mesmo valor de Q, no modelo Quasimoto, a quantidade de pontos ficou com $\sim$18% da nuvem original.

Com esta taxa de remoção de pontos, executando a reconstrução com o algoritmo de Poisson, não percebem-se visualmente falhas expressivas na superfície reconstruída. Contudo, é possível perceber que a qualidade da reconstrução diminui a medida em que o parâmetro de qualidade é relaxado. Visualmente a qualidade menor da imagem resultante não é percebida de forma clara, mas a comparação usando a métrica de Hausdorff mostra melhor este efeito. Usando o algoritmo SSD para reconstrução os resultados foram um pouco diferentes, trazendo como melhor valor para a métrica de Hausdorff a nuvem simplificada com parâmetro $Q = 0,04$. Ou seja, com o algoritmo SSD, mesmo diminuindo a exigência do parâmetro de qualidade, a superfície reconstruída não aumentou a distância daquela gerada pela nuvem original. As Figuras 25 e 26 mostram os resultados da reconstrução para os algoritmos SSD e Poisson respectivamente.

Comparando os resultados entre os modelos Quasimoto e Gargoyle é possível perceber, como mencionado anteriormente, a diferença entre as taxas de simplificação. Para exemplificar,

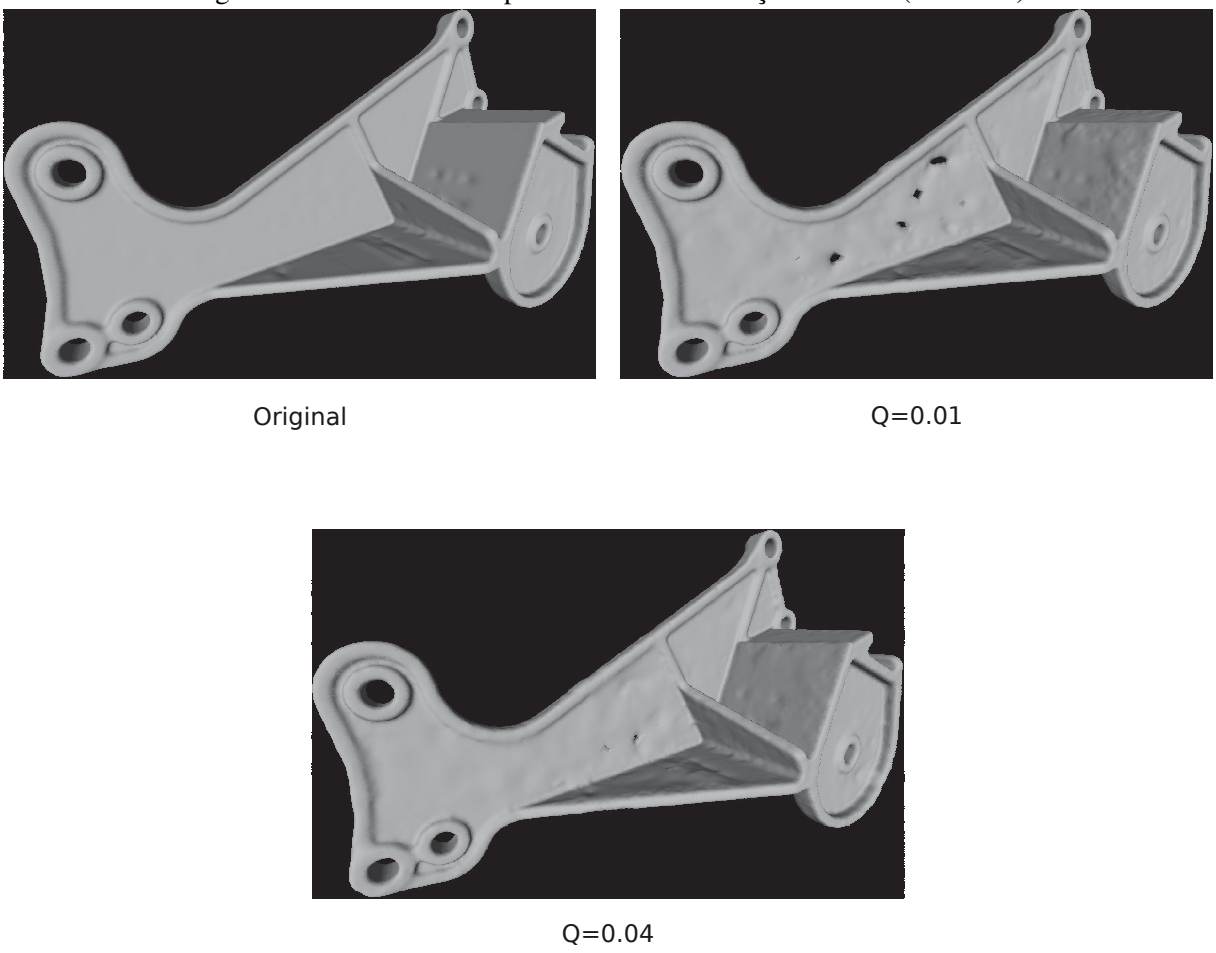
Figura 22 – Resultados do processo de reconstrução SSD (Daratech)



Fonte: Imagem produzida pelo autor

no primeiro modelo, com $Q = 0,04$, foram mantidos $\sim$5% dos pontos originais. Enquanto no segundo modelo foram mantidos $\sim$13% dos pontos. Esta diferença ocorre para qualquer valor de Q. Este comportamento é explicado pela base do algoritmo, que é a detecção de planos no conjunto de pontos. Após a detecção de um plano, que atenda ao parâmetro de qualidade, apenas 3 pontos são mantidos para representar esta região da imagem. Assim, a taxa de simplificação está ligada diretamente à quantidade de planos detectados na nuvem de pontos. Quanto maior for variação da distância dos pontos de um subconjunto ao plano de aproximação, maior será o número de planos necessários para encontrar a melhor aproximação. Ou seja, quanto maior for a inclinação encontrada na superfície, um maior número de pontos será necessário para representá-la. Os modelos Quasimoto e Gargoyle têm diferenças de topologia. No primeiro caso, a superfície possui menos variações e detalhes, sendo possível aproximar áreas maiores, que contenham maior número de pontos, com um único plano. Regiões com mais detalhes existem, mas com menos ocorrências. O modelo Gargoyle possui variação maior na superfície, mais regiões com detalhes e variação na inclinação da superfície. Neste caso, não existem grandes regiões que possam ser representadas por um único plano. Por consequência, mais planos são gerados e a taxa de simplificação é menor. A Figura 27 mostra os planos detectados

Figura 23 – Resultados do processo de reconstrução Poisson (Daratech)



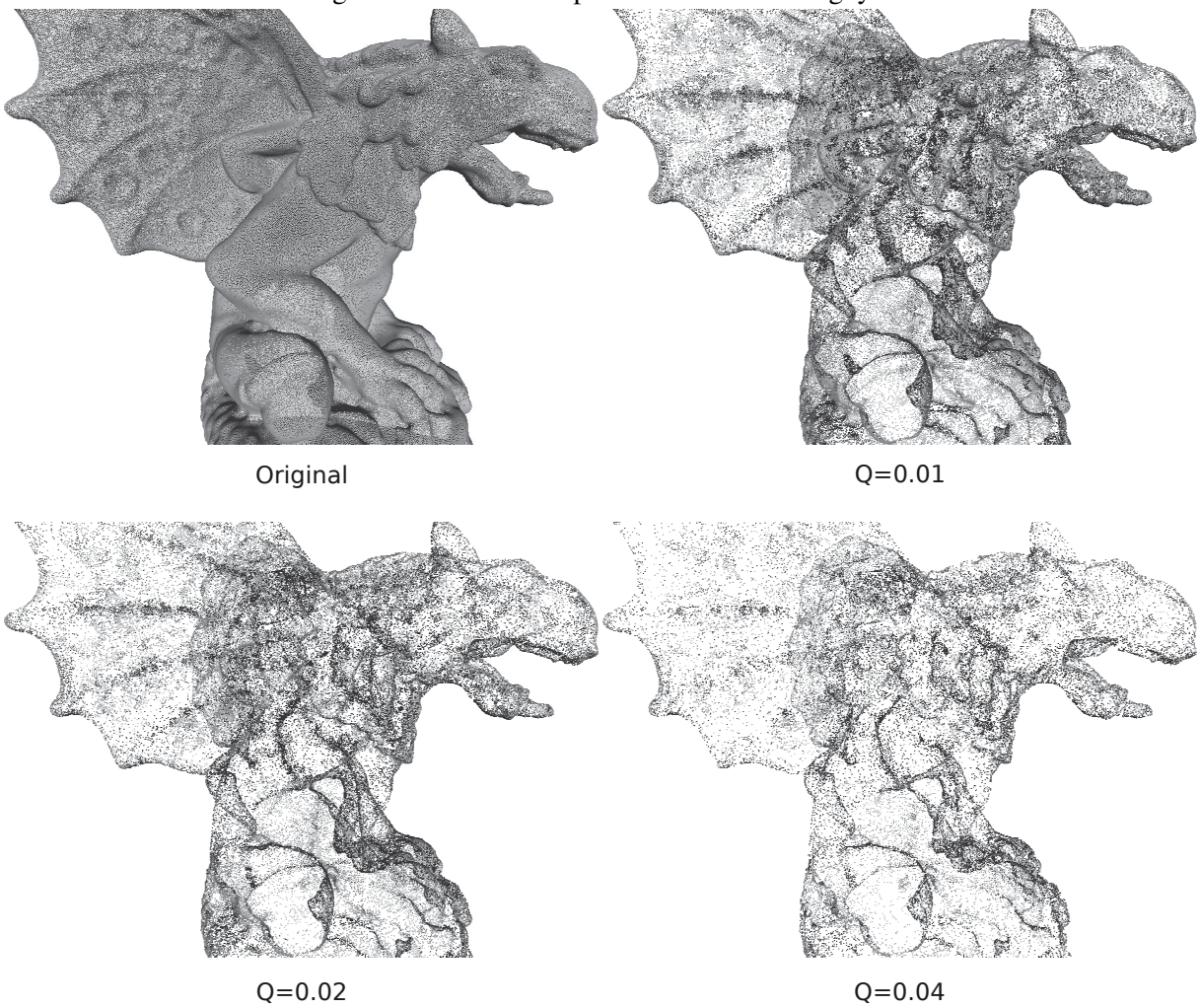
Fonte: Imagem produzida pelo autor

nos dois modelos, com o processo de simplificação executado com os parâmetros $Q = 0,04$, $N = 5$ e $D = 4$.

Para o modelo Dancing Children o algoritmo de simplificação proposto teve desempenho similar ao modelo Quasimoto. Ambos tem características semelhantes, como partes articuladas (cabeça, braços e pernas) e regiões mais uniformes que favorecem a detecção de um número menor de planos. Contudo, o modelo Dancing Children adiciona alguns elementos mais complexos como o anel de base da imagem, a conexão entre as figuras e os vincos das roupas. Este modelo possui 468.022 pontos e $29,0$MB de tamanho, o processo de reconstrução com o algoritmo SSD foi executado em ∼43,8s e a reconstrução com o algoritmo de Poisson foi executado em ∼29,9s. O processo de simplificação foi executado com os parâmetros $N = 5$, $D = 4$, obtendo os 3 pontos mais próximos do plano de aproximação do conjunto, variando o parâmetro Q com os valores $0,03$ e $0,04$. Os testes foram executados em um computador com processador Intel Core i5-3337U, com 4 x 1,80 Ghz, 6 Gb de memória e sistema operacional GNU/Linux Ubuntu 14.04, 64 bits. A Tabela 7 mostra os resultados dos testes.

Neste caso apenas os resultados de duas variações do parâmetro Q foram usados para exemplificar o desempenho do algoritmo. A Figura 28 mostra os conjuntos de pontos original e os

Figura 24 – Nuvem de pontos do modelo Gargoyle



Fonte: Imagem produzida pelo autor

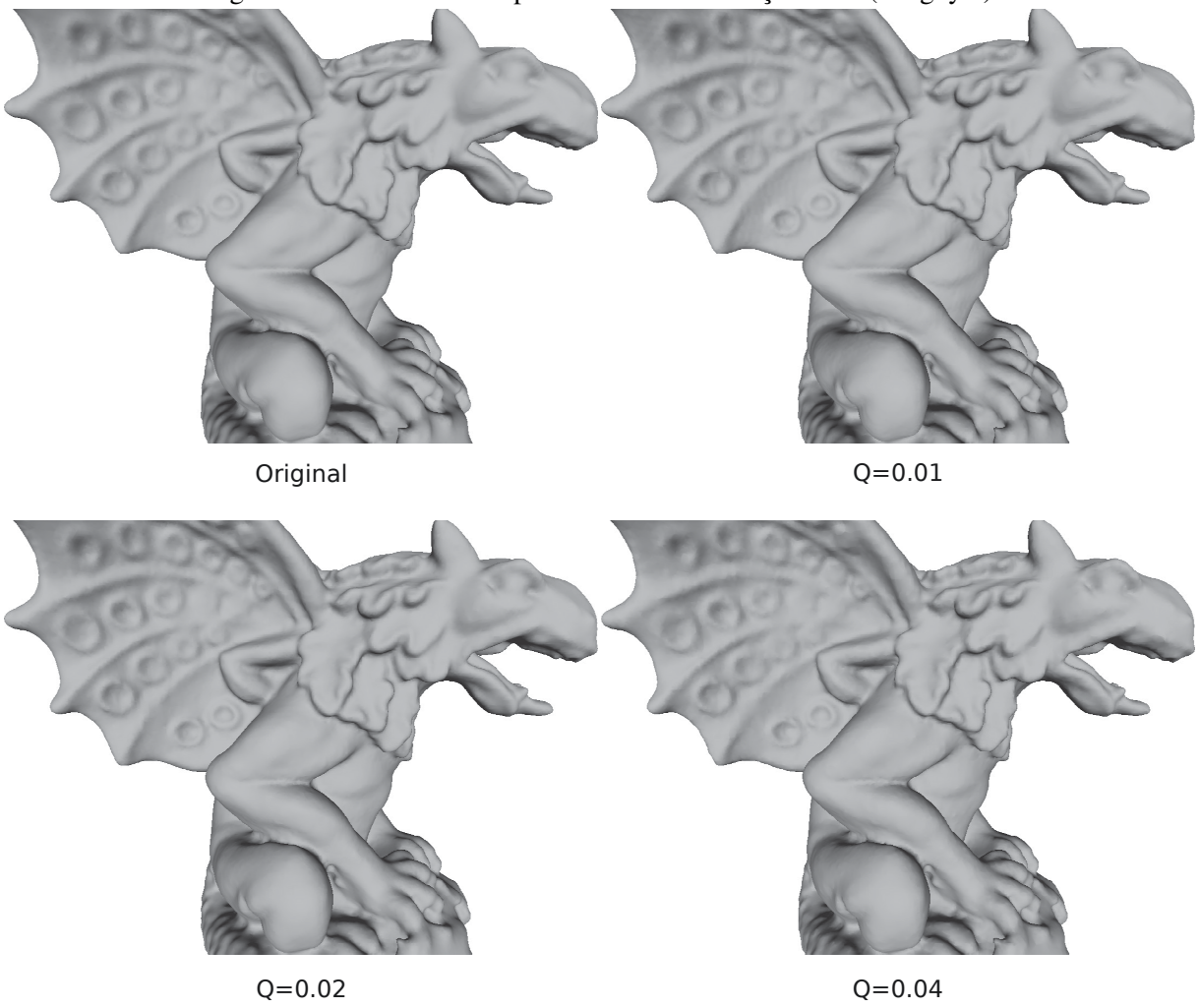
Tabela 7 – Resultados do processo de simplificação e reconstrução da superfície (Dancing Children)

Q	**Original**	**0,03**	**0,04**
Pontos	468.022	48.296	37.143
Tamanho	29MB	2,8MB	2,1MB
Taxa	-	∼89%	∼92%
Tempo	-	∼0,4s	∼0,4s
SSD	∼43,8	∼15,1s	∼13,3s
Hausdorff	-	∼0,343	∼0,554
Poisson	∼29,9	∼15,9s	∼13,5s
Hausdorff	-	∼0,394	∼0,399

Fonte: Elaborada pelo autor

resultantes de cada execução da simplificação para o modelo. Como mencionado acima, também houve boa taxa de remoção dos pontos do conjunto original, chegando a ∼8% do conjunto de entrada com $Q = 0,04$. Apesar de similar, o desempenho do algoritmo, em termos da taxa de simplificação, não foi tão efetivo quanto para o modelo Quasimoto. Comportamento justifi-

Figura 25 – Resultados do processo de reconstrução SSD (Gargoyle)



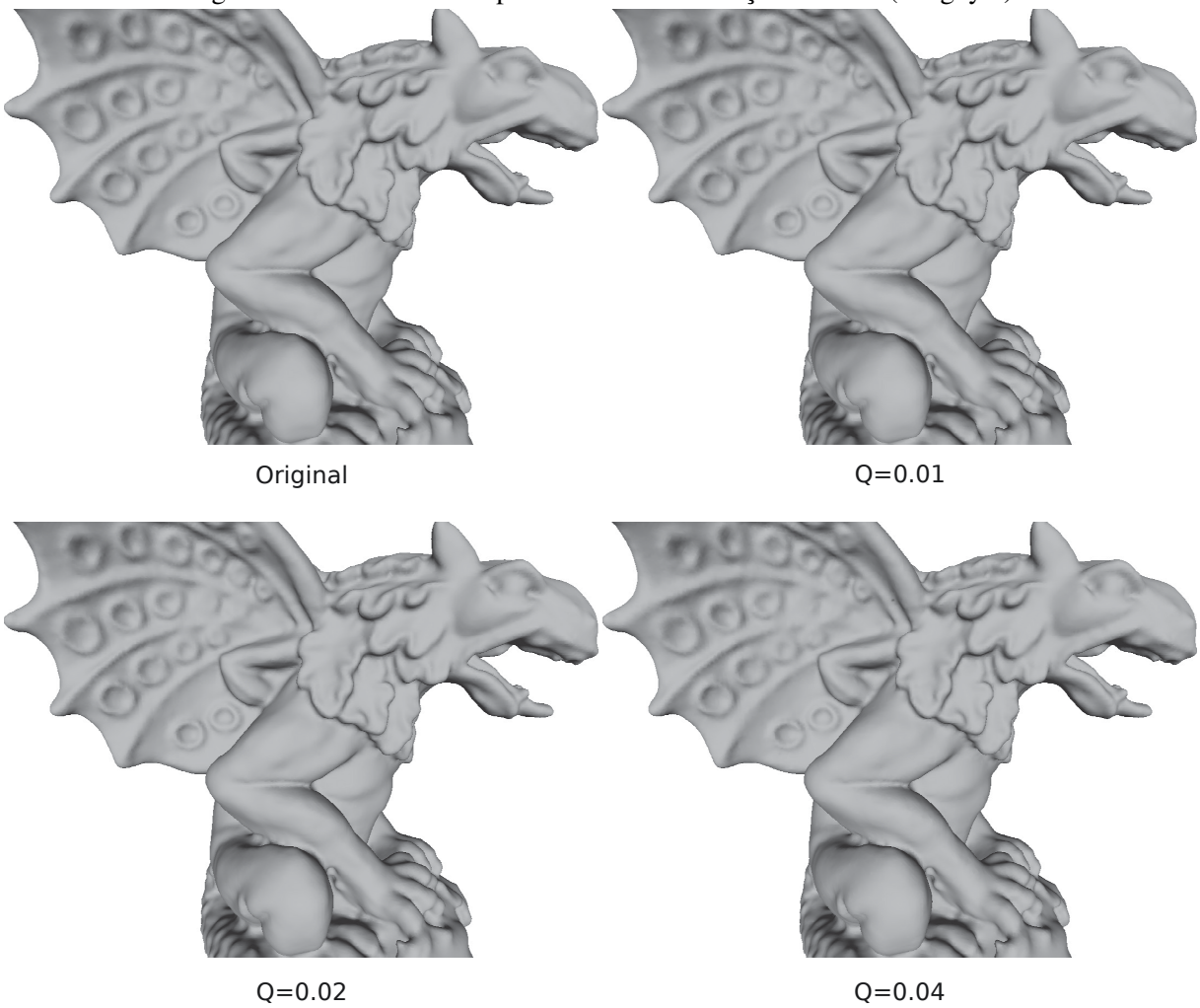
Fonte: Imagem produzida pelo autor

cado novamente pela diferença entre os modelos, sendo necessário um maior número de pontos para representar regiões não uniformes.

As Figuras 29 e 30 mostram os resultados do processo de reconstrução do modelo Dancing Children para os algoritmos SSD e Poisson respectivamente. As imagens geradas a partir das nuvens simplificadas mostram boa aproximação com aquela gerada a partir da nuvem original. Este resultado é percebido visualmente e pela métrica de Hausdorff apontada na Tabela 7. Uma distorção maior é verificada nas regiões mais planas da imagem, onde um número maior de pontos é aproximado pelo mesmo plano e, por consequência, um maior número de pontos é removido. Este efeito pode ser visto, principalmente, no anel da base da imagem e no chapéu da primeira criança no lado esquerdo da imagem. Na medida com que o parâmetro Q é relaxado esta distorção tende a ocorrer mais (obviamente pela remoção de um número maior de pontos e pela baixa qualidade de aproximação dos planos aos pontos que formam a superfície).

Por fim, foram executados testes com o modelo Bunny (Stanford Bunny) para avaliação do algoritmo. Aqui apenas um resultado está sendo mostrado para exemplificar a variação

Figura 26 – Resultados do processo de reconstrução Poisson (Gargoyle)

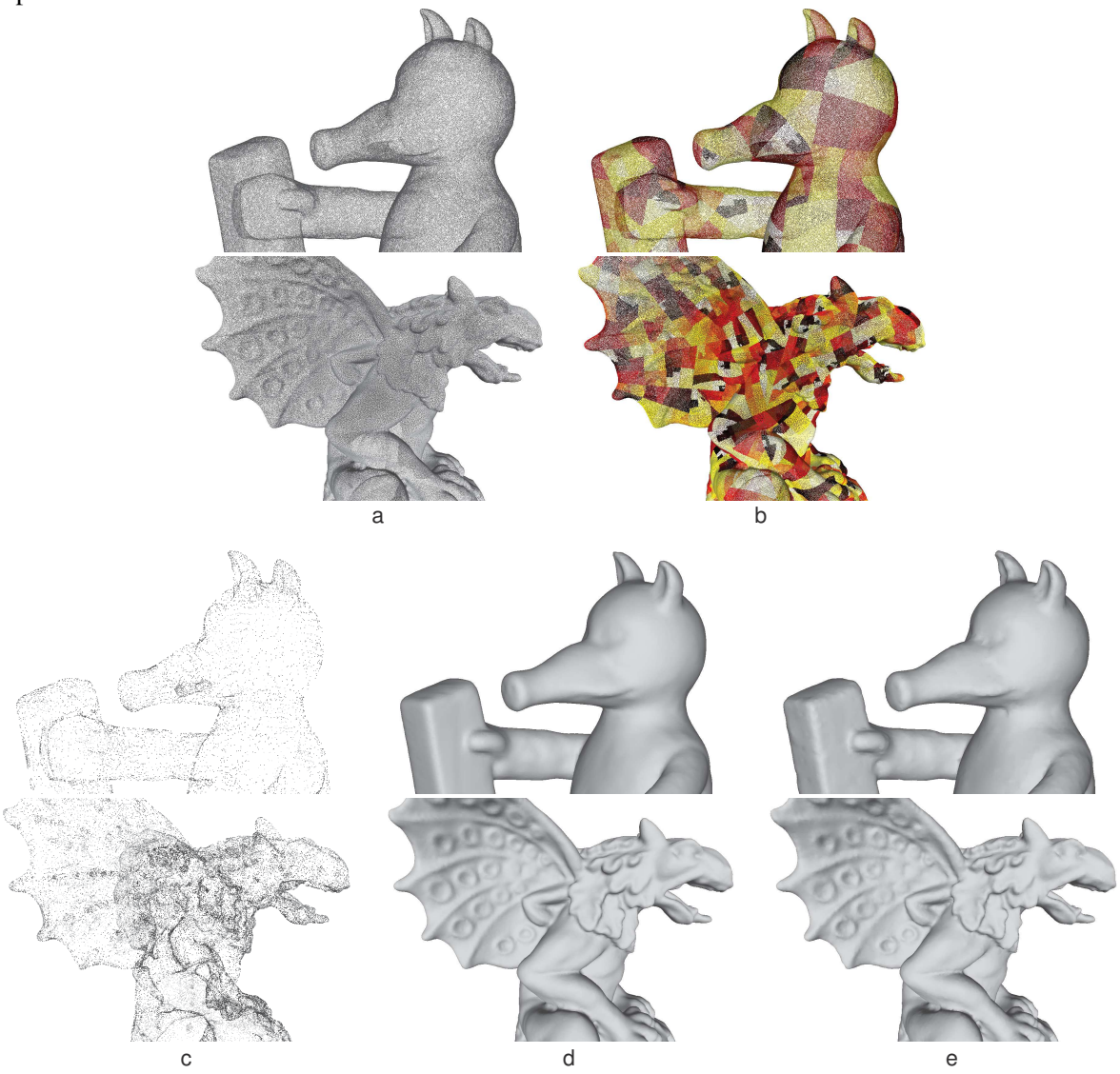


Fonte: Imagem produzida pelo autor

dos parâmetros Q e N, e como a reconstrução da nuvem simplificada pode ficar próxima da superfície reconstruída com a nuvem original. Neste caso, a simplificação foi executada com $Q = 0,00001$ (grande exigência de qualidade da aproximação dos planos aos pontos), $N = 20$ (no mínimo 20 pontos em cada aproximação) e divisão do conjunto de dados em duas partes usando a primeira componente principal ($D = 2$). O modelo possui 362.261 pontos e 21MB de tamanho. O resultado foi uma nuvem com 77.399 pontos, em um arquivo com $4,2$MB (uma taxa de simplificação de $\sim$78%). A Figura 31 mostra as superfícies reconstruídas com os algoritmos SSD e Poisson para a nuvem original e a resultante do processo de simplificação. A distância de Hausdorff para o algoritmo SSD foi $0,001590$ e para o algoritmo de Poisson foi $0,001451$. Estes valores mostram a similaridade entre as imagens, que também é percebida no resultado visual.

Encerrando os resultados obtidos, é importante destacar a quantidade de parâmetros diferentes disponíveis no protótipo para validação do modelos. É possível variar a qualidade da aproximação (Q), a quantidade mínima de pontos em um plano (N), tipo de divisão do con-

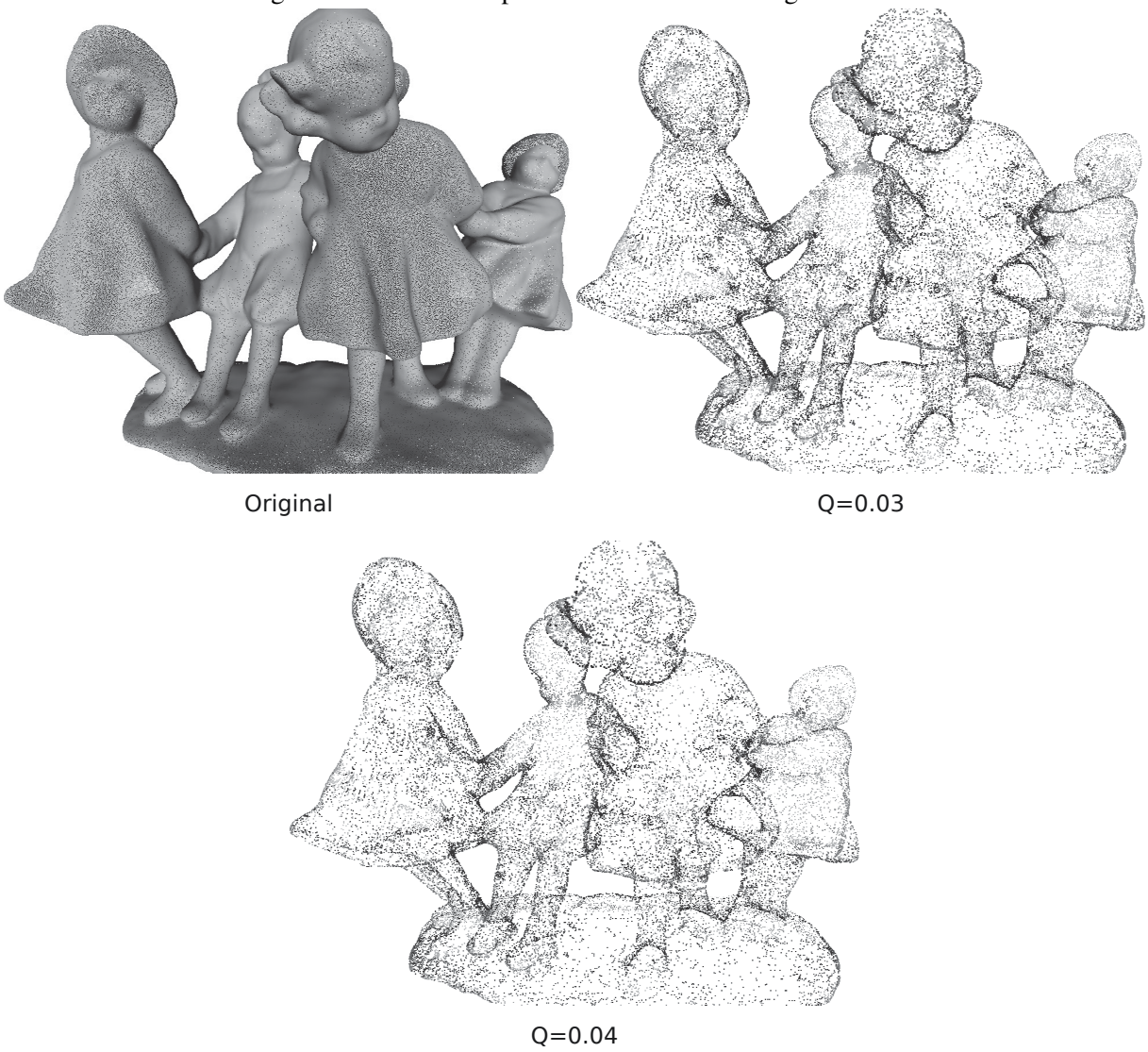
Figura 27 – Simplificação das nuvens de pontos dos modelos Quasimoto e Gargoyle pelo algoritmo desenvolvido. Modelo Quasimoto (no alto) com redução de 350.000 pontos para 16.730 pontos. Modelo Gargoyle (embaixo) com redução de 481.358 pontos para 61.485 pontos. Da esquerda para a direita: (a) nuvens de pontos originais, (b) divisão das regiões por aproximação linear, (c) nuvens de pontos simplificadas, (d) superfícies reconstruídas a partir das nuvens de pontos originais e (e) superfícies reconstruídas a partir das nuvens filtradas.



Fonte: Imagem produzida pelo autor

junto de entrada usando as componentes principais (2 ou 4 conjuntos) e o tipo de obtenção dos 3 pontos que representarão o plano (mais próximos do plano ou mais distantes do ponto médio). O protótipo implementado na linguagem de programação permite, também, a alteração da quantidade de pontos mantidos por região aproximada, retornar todos os pontos de uma região com aproximação ruim (que não atingiu o valor do parâmetro de qualidade da aproximação) e atualização do vetor normal (atualmente é aquele definido pela terceira componente principal em cada região obtida ao final do processo de particionamento). Desta forma, é possível fazer um grande número de combinações e testes dos resultados. Aqui foram usados os parâmetros

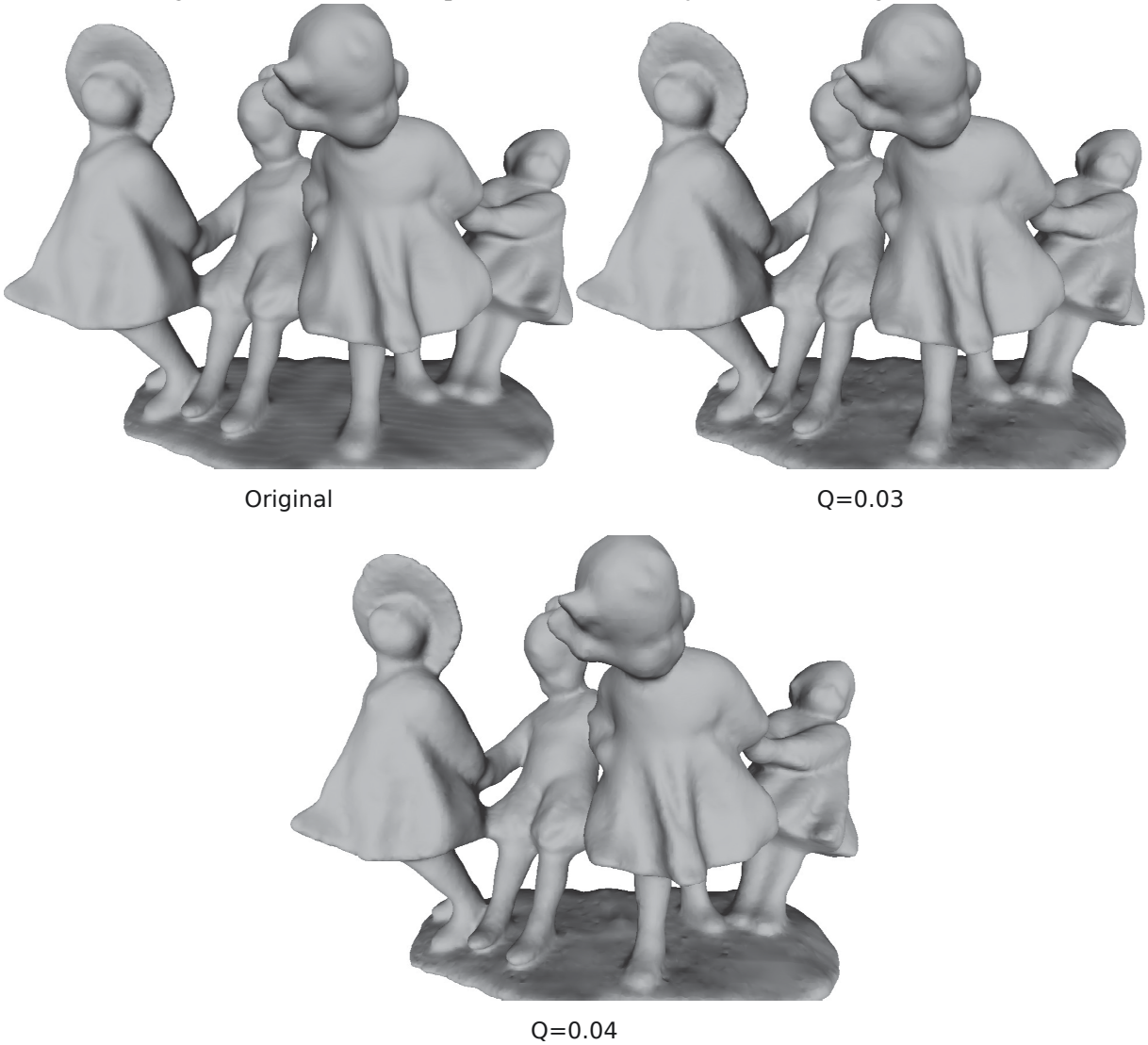
Figura 28 – Nuvem de pontos do modelo Dancing Children



Fonte: Imagem produzida pelo autor

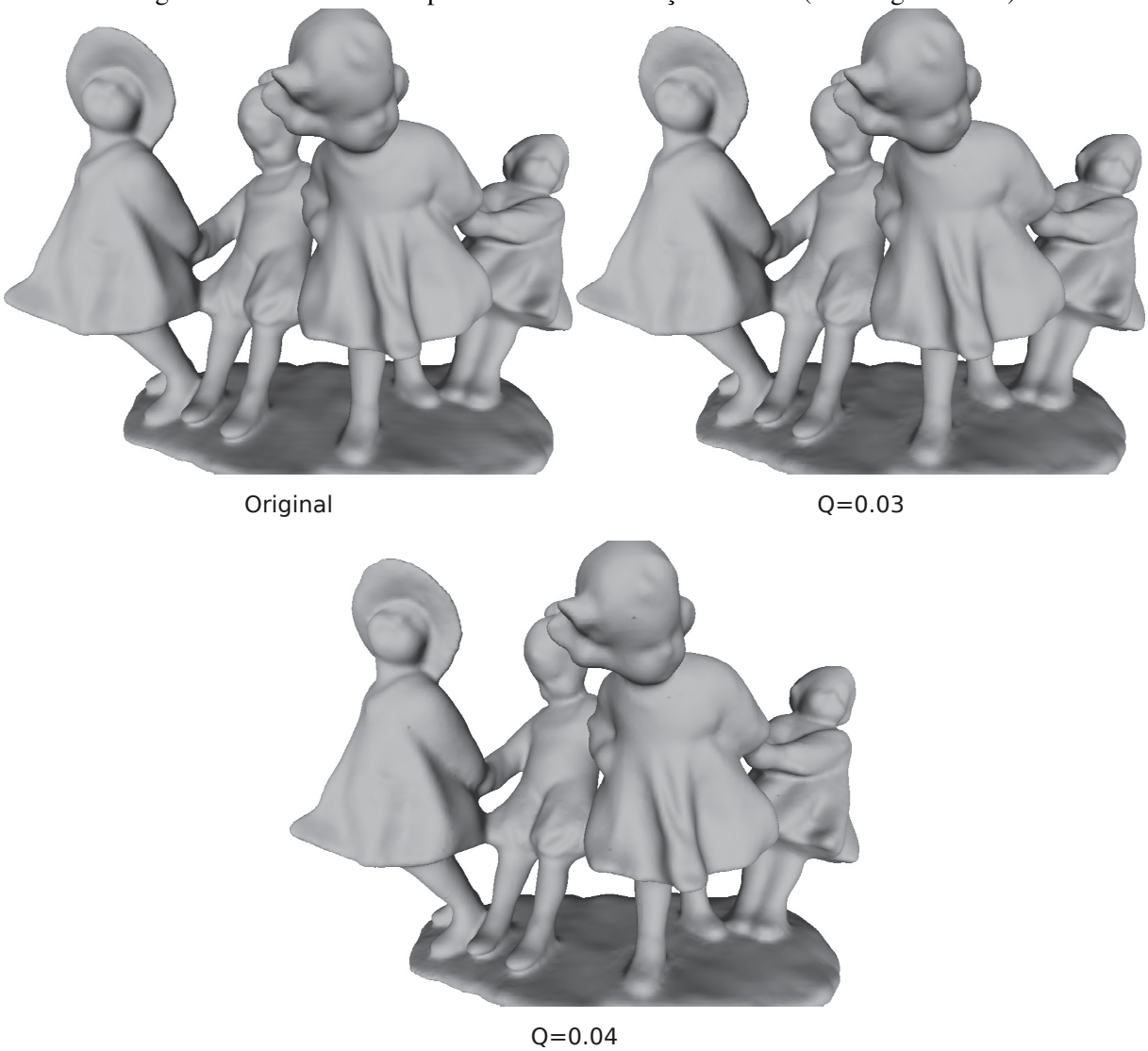
que tiveram, na análise, melhores resultados. A variação principal foi no limite da qualidade (Q), parâmetro que tem maior influência no resultado das imagens e na taxa de simplificação.

Figura 29 – Resultados do processo de reconstrução SSD (Dancing Children)



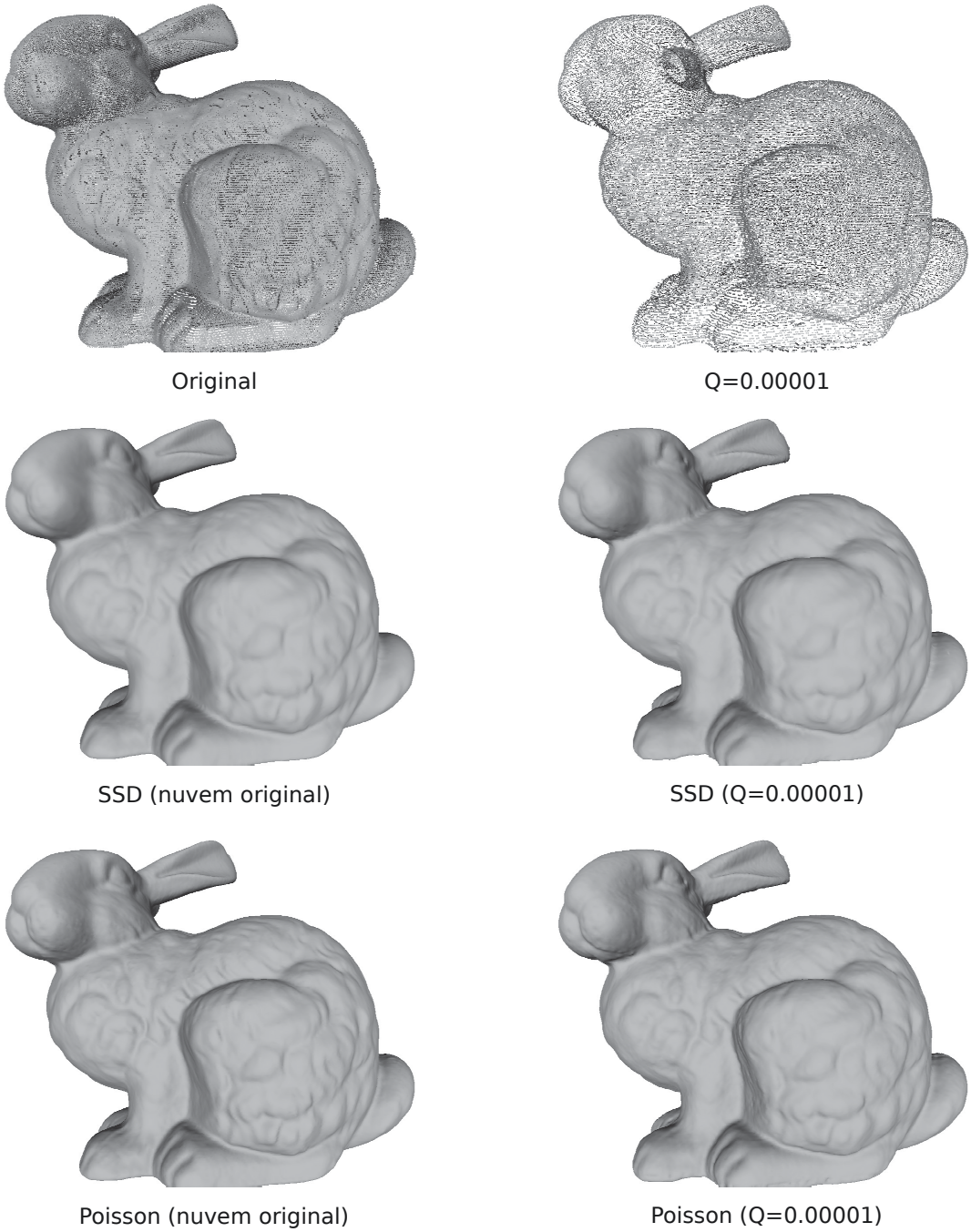
Fonte: Imagem produzida pelo autor

Figura 30 – Resultados do processo de reconstrução Poisson (Dancing Children)



Fonte: Imagem produzida pelo autor

Figura 31 – Resultados do processo de reconstrução SSD e Poisson (Bunny)



Fonte: Imagem produzida pelo autor

6 CONCLUSÃO

Este trabalho apresentou um novo algoritmo para simplificação de nuvens de pontos, baseado na aplicação da Análise de Componentes Principais no conjunto de pontos de entrada. O algoritmo aplica as componentes principais para definir as partições sucessivas do conjunto de entrada, para obter a aproximação linear (por planos) em cada partição e para avaliar a qualidade de cada aproximação. A abordagem deste trabalho difere de outros na forma da divisão dos pontos, usando sua posição no espaço em relação às componentes principais. No trabalho, o algoritmo desenvolvido foi detalhado, mostrando o processo de divisão, a medida de qualidade da aproximação e a forma de seleção dos pontos. Para avaliação do processo foram usadas diferentes nuvens de pontos, com diferentes características, e a aplicação das nuvens simplificadas foi na reconstrução das superfícies trabalhando com dois algoritmos diferentes. Nos resultados foram obtidas taxas de simplificação de até 95%, mantendo a qualidade da reconstrução da superfície final. Os testes também mostram a redução do tempo total de processamento da reconstrução das superfícies (considerando o tempo de simplificação somado ao tempo de reconstrução). O processo é executado diretamente na nuvem de pontos sem necessidade de uma malha de polígonos e o algoritmo foi adaptativo ao conjunto de entrada, mantendo mais pontos em regiões com maior nível de detalhes.

Entre as qualidades deste trabalho está a simplicidade do modelo e do algoritmo proposto, sendo de fácil implementação. Também, não há necessidade do particionamento do conjunto de entrada em voxels ou clusters, ou a definição de uma vizinhança prévia para verificação de regiões planares. Além disso, não há necessidade de organizar os pontos em uma estrutura de dados como grades de voxels regulares, Octrees ou KD-Trees. Nesta abordagem, PCA é usada para definir a vizinhança como aqueles pontos que pertencem a mesma região planar. O algoritmo executa um processo recursivo de aproximações lineares usando os planos gerados a partir das componentes principais, particionando o conjunto de entrada onde é necessário. Desta forma, o algoritmo consegue executar uma reamostragem adaptativa dos pontos da nuvem de entrada, considerando a curvatura local da superfície, usando a variação espacial das terceiras componentes em cada passo do processo recursivo.

A abordagem usando PCA nos dá uma classificação estatística dos pontos na nuvem de entrada, dividindo o espaço em direções com máxima independência linear. Assim, as componentes principais fornecem uma forma ótima de aproximação geométrica, através de uma abordagem linear, usando planos para aproximar os pontos no espaço tridimensional. O processo de aproximação sucessiva, de forma recursiva, segue a distribuição estatística dos pontos em cada região particionada. Por causa destas propriedades o algoritmo proposto pode classificar de forma mais natural uma nuvem de pontos em regiões bem aproximadas por planos. Esta é a principal característica que diferencia este trabalho dos demais existentes nesta área. Ao final do processo de particionamento, um critério de seleção deve ser aplicado para escolher os pontos mais representativos de cada região. Estes pontos formarão o conjunto de dados

simplificado.

A quantidade e a forma de seleção dos pontos amostrados em cada região dependerá da necessidade de aplicação do usuário. O usuário, também, precisará decidir entre uma qualidade maior da reconstrução do modelo tridimensional e a taxa de simplificação da nuvem de pontos. Neste trabalho a seleção destes pontos é bastante simples, com a obtenção de apenas 3 como amostra representativa de cada região, usando como critério sua posição em relação ao plano de aproximação. Esta forma de seleção proporciona uma alta taxa de simplificação, mas não garante uma amostragem mais representativa das características da superfície. Contudo, isto demonstra a qualidade do algoritmo de divisão planar das regiões da superfície amostrada pela nuvem de pontos, atingindo bons resultados mesmo com a seleção de pontos por região não apurada.

Os piores resultados obtidos na reconstrução das nuvens simplificadas ocorreram nos casos onde existem longas regiões planas, bordas longas e vivas, e regiões planas paralelas nas nuvens originais. Nestes casos foram observadas irregularidades nas bordas e buracos nas áreas planas da superfície. Estes problemas podem ser atribuídos à forma simples de escolha dos pontos que representam cada aproximação local, sendo esta uma questão que deve ser mais explorada.

Por fim, existem diferentes aspectos que podem ser melhorados neste trabalho. A eficiência do algoritmo está baseada na divisão recursiva do conjunto de dados usando os autovetores. A obtenção dos pontos que serão mantidas no conjunto de dados filtrado é ainda muito simples. Para trabalhos futuros, a principal evolução do algoritmo deve ser a forma de seleção dos pontos finais. Uma das ideias é obter os pontos a partir do convex hull de suas projeções no plano de aproximação, buscando representar melhor regiões limítrofes entre duas aproximações. Outro problema a ser resolvido é a seleção de pontos em regiões com duas superfícies paralelas próximas, uma alternativa seria avaliar a orientação do vetor normal nestas regiões. Também merece atenção a avaliação de outras métricas para a qualidade da aproximação dos pontos ao plano. O valor de λ_3 pode ser usado como limite de qualidade da aproximação linear ou pode ser usado o conceito de proporção entre os autovalores para determinar a qualidade da distribuição dos planos no espaço (FERNANDEZ, 2005).

REFERÊNCIAS

ALEXA, M. et al. Point set surfaces. In: CONFERENCE ON VISUALIZATION '01, 2001, Washington, DC, USA. **Proceedings...** IEEE Computer Society, 2001. p. 21–28. (VIS '01).

AMENTA, N.; BERN, M. Surface reconstruction by voronoi filtering. In: FOURTEENTH ANNUAL SYMPOSIUM ON COMPUTATIONAL GEOMETRY, 1998, New York, NY, USA. **Proceedings...** ACM, 1998. p. 39–48. (SCG '98).

AMENTA, N. et al. A simple algorithm for homeomorphic surface reconstruction. In: SIXTEENTH ANNUAL SYMPOSIUM ON COMPUTATIONAL GEOMETRY, 2000, New York, NY, USA. **Proceedings...** ACM, 2000. p. 213–222. (SCG '00).

BERGER, M. et al. A benchmark for surface reconstruction. **ACM Trans. Graph.**, New York, NY, USA, v. 32, n. 2, p. 20:1–20:17, Apr 2013.

BOLITHO, M. et al. Multilevel streaming for out-of-core surface reconstruction. In: FIFTH EUROGRAPHICS SYMPOSIUM ON GEOMETRY PROCESSING, 2007, Aire-la-Ville, Switzerland, Switzerland. **Proceedings...** Eurographics Association, 2007. p. 69–78. (SGP '07).

BOLITHO, M. et al. Parallel poisson surface reconstruction. In: INTERNATIONAL SYMPOSIUM ON ADVANCES IN VISUAL COMPUTING: PART I, 5., 2009, Berlin, Heidelberg. **Proceedings...** Springer-Verlag, 2009. p. 678–689. (ISVC '09).

CALAKLI, F.; TAUBIN, G. SSD: Smooth signed distance surface reconstruction. **Computer Graphics Forum**, [S.l.], v. 30, n. 7, p. 1993–2002, 2011.

CALAKLI, F.; TAUBIN, G. **Smooth signed distance**. http://mesh.brown.edu/ssd/.

CALAKLI, F.; TAUBIN, G. SSD-C: Smooth signed distance colored surface reconstruction. In: EXPANDING THE FRONTIERS OF VISUAL ANALYTICS AND VISUALIZATION, 2012. **Proceedings...** Springer London, 2012. p. 323–338.

CIGNONI, P.; ROCCHINI, C.; SCOPIGNO, R. Metro: measuring error on simplified surfaces. **Computer Graphics Forum**, [S.l.], v. 17, n. 2, p. 167–174, 1998.

EIGEN Library. http://eigen.tuxfamily.org/.

FERNANDEZ, O. Obtaining a best fitting plane through 3d georeferenced data. **Journal of Structural Geology**, [S.l.], v. 27, n. 5, p. 855 – 858, 2005.

GUENNEBAUD, G.; GROSS, M. Algebraic point set surfaces. **ACM Trans. Graph.**, New York, NY, USA, v. 26, n. 3, Jul 2007.

HARTLEY, R.; ZISSERMAN, A. **Multiple view geometry in computer vision**. 2. ed. New York, NY, USA: Cambridge University Press, 2003.

HODGETTS, D. Laser scanning and digital outcrop geology in the petroleum industry: a review. **Marine and Petroleum Geology**, [S.l.], v. 46, n. 0, p. 335 – 354, 2013.

JACKSON, J. E. **A user's guide to principal components**. New York: John Wiley & Sons, 1991.

KAZHDAN, M. Reconstruction of solid models from oriented point sets. In: THIRD EUROGRAPHICS SYMPOSIUM ON GEOMETRY PROCESSING, 2005, Aire-la-Ville, Switzerland, Switzerland. **Proceedings...** Eurographics Association, 2005. (SGP '05).

KAZHDAN, M.; BOLITHO, M.; HOPPE, H. Poisson surface reconstruction. In: FOURTH EUROGRAPHICS SYMPOSIUM ON GEOMETRY PROCESSING, 2006, Aire-la-Ville, Switzerland, Switzerland. **Proceedings...** Eurographics Association, 2006. p. 61–70. (SGP '06).

KAZHDAN, M.; BOLITHO, M.; HOPPE, H. **Poisson surface reconstruction**. http://research.microsoft.com/en-us/um/people/hoppe/proj/poissonrecon/.

KOLLURI, R. Provably good moving least squares. In: ACM-SIAM SYMPOSIUM ON DISCRETE ALGORITHMS, 2005. **Proceedings...** [S.l.: s.n.], 2005. p. 1008–1018.

LANMAN, D.; CRISPELL, D.; TAUBIN, G. Surround structured lighting: 3-D scanning with orthographic illumination. **Computer Vision and Image Understanding**, [S.l.], v. 113, n. 11, p. 1107 – 1117, 2009. Special issue on New Advances in 3D Imaging and Modeling.

LEE, P. F.; HUANG, C.-P. The dso feature based point cloud simplification. In: COMPUTER GRAPHICS, IMAGING AND VISUALIZATION (CGIV), 2011 EIGHTH INTERNATIONAL CONFERENCE ON, 2011. **Anais...** [S.l.: s.n.], 2011. v. 3, p. 1–6.

LEVOY, M. The digital michelangelo project. In: D DIGITAL IMAGING AND MODELING, 1999. PROCEEDINGS. SECOND INTERNATIONAL CONFERENCE ON, 3., 1999. **Proceedings...** [S.l.: s.n.], 1999. p. 2–11.

LI, H.; XU, P.; SHEN, Y. A self-adaption fast point cloud simplification algorithm based on normal eigenvalues. In: IMAGE AND SIGNAL PROCESSING (CISP), 2014 7TH INTERNATIONAL CONGRESS ON, 2014. **Anais...** [S.l.: s.n.], 2014. p. 852–856.

LI, L. et al. Point cloud simplification based on an affinity propagation clustering algorithm. In: ARTIFICIAL INTELLIGENCE AND COMPUTATIONAL INTELLIGENCE, 2009. AICI '09. INTERNATIONAL CONFERENCE ON, 2009. **Anais...** [S.l.: s.n.], 2009. v. 3, p. 163–167.

LORENSEN, W. E.; CLINE, H. E. Marching cubes: a high resolution 3d surface construction algorithm. **SIGGRAPH Comput. Graph.**, New York, NY, USA, v. 21, n. 4, p. 163–169, Aug 1987.

MANSON, J.; PETROVA, G.; SCHAEFER, S. Streaming surface reconstruction using wavelets. **Computer Graphics Forum (Proceedings of the Symposium on Geometry Processing)**, [S.l.], v. 27, n. 5, p. 1411–1420, 2008.

MIAO, Y.; PAJAROLA, R.; FENG, J. Curvature-aware adaptive re-sampling for point-sampled geometry. **Computer-Aided Design**, [S.l.], v. 41, n. 6, p. 395–403, 2009.

MORENO, D.; TAUBIN, G. Simple, accurate, and robust projector-camera calibration. In: D IMAGING, MODELING, PROCESSING, VISUALIZATION AND TRANSMISSION (3DIMPVT), 2012 SECOND INTERNATIONAL CONFERENCE ON, 3., 2012. **Proceedings...** [S.l.: s.n.], 2012. p. 464–471.

NGUYEN, V.-S.; BAC, A.; DANIEL, M. Simplification of 3d point clouds sampled from elevation surfaces. In: INTERNATIONAL CONFERENCE ON COMPUTER GRAPHICS, VISUALIZATION AND COMPUTER VISION, 21., 2013, Plzen, Czech Republic. **Proceedings...** [S.l.: s.n.], 2013. n. 2, p. 60–69. (WSCG2013, v. 21).

OHTAKE, Y.; BELYAEV, A.; SEIDEL, H.-P. 3d scattered data interpolation and approximation with multilevel compactly supported RBFs. **Graphical Models**, [S.l.], v. 67, n. 3, p. 150 – 165, 2005. Special Issue on {SMI} 2003.

OHTAKE, Y. et al. Multi-level partition of unity implicits. **ACM Trans. Graph.**, New York, NY, USA, v. 22, n. 3, p. 463–470, Jul 2003.

OPENSCENEGRAPH. http://www.openscenegraph.org/.

PAULY, M.; GROSS, M.; KOBBELT, L. P. Efficient simplification of point-sampled surfaces. In: CONFERENCE ON VISUALIZATION '02, 2002, Washington, DC, USA. **Proceedings...** IEEE Computer Society, 2002. p. 163–170. (VIS '02).

SCHAEFER, S.; WARREN, J. Dual marching cubes: primal contouring of dual grids. In: COMPUTER GRAPHICS AND APPLICATIONS, 2004. PG 2004. PROCEEDINGS. 12TH PACIFIC CONFERENCE ON, 2004. **Proceedings...** [S.l.: s.n.], 2004. p. 70–76.

SCHALL, O.; SAMOZINO, M. Surface from scattered points: a brief survey of recent developments. In: INTERNATIONAL WORKSHOP TOWARDS SEMANTIC VIRTUAL ENVIRONMENTS, 1., 2005, Villars, Switzerland. **Proceedings...** MIRALab, 2005. p. 138–147.

SEITZ, S. et al. A comparison and evaluation of multi-view stereo reconstruction algorithms. In: COMPUTER VISION AND PATTERN RECOGNITION, 2006 IEEE COMPUTER SOCIETY CONFERENCE ON, 2006. **Proceedings...** [S.l.: s.n.], 2006. v. 1, p. 519–528.

SHI, B.-Q.; LIANG, J.; LIU, Q. Adaptive simplification of point cloud using k-means clustering. **Computer-Aided Design**, [S.l.], v. 43, n. 8, p. 910–922, 2011.

SNAVELY, N.; SEITZ, S. M.; SZELISKI, R. Photo tourism: exploring photo collections in 3d. **ACM Trans. Graph.**, New York, NY, USA, v. 25, n. 3, p. 835–846, Jul 2006.

YU, Z. et al. Asm: an adaptive simplification method for 3d point-based models. **Computer-Aided Design**, [S.l.], v. 42, n. 7, p. 598–612, 2010.

ZHOU, K. et al. Data-parallel octrees for surface reconstruction. **Visualization and Computer Graphics, IEEE Transactions on**, [S.l.], v. 17, n. 5, p. 669–681, May 2011.