UNIVERSIDADE ESTADUAL PAULISTA
Faculdade de Ciências - Bauru
Bacharelado em Ciência da Computação

Fernando Vinicius Lima Fernandes

SIMPLIFICAÇÃO DE MALHAS VISANDO REPRESENTAÇÕES EM MULTIRRESOLUÇÃO

UNESP

2013

Fernando Vinicius Lima Fernandes

SIMPLIFICAÇÃO DE MALHAS VISANDO REPRESENTAÇÕES EM MULTIRRESOLUÇÃO

Prof. Dr. Antonio Carlos Sementille
Prof. Dr. João Fernando Marar

Monografia apresentada junto à disciplina Projeto e Implementação de Sistemas II, do curso de Bacharelado em Ciência da Computação, Faculdade de Ciências, Unesp, campus de Bauru, como parte do Trabalho de Conclusão de Curso.

UNESP

2013

Fernando Vinicius Lima Fernandes

SIMPLIFICAÇÃO DE MALHAS VISANDO REPRESENTAÇÕES EM MULTIRRESOLUÇÃO

Monografia apresentada junto à disciplina Projeto e Implementação de Sistemas II, do curso de Bacharelado em Ciência da Computação, Faculdade de Ciências, Unesp, campus de Bauru, como parte do Trabalho de Conclusão de Curso.

BANCA EXAMINADORA

Prof. Dr. Antonio Carlos Sementille
DCo - FC - UNESP - Bauru
Orientador

Prof. Dr. Simone das Graças Domingues Prado
UNESP - Bauru

Prof. Dr. Renê Pegoraro
UNESP - Bauru

Bauru, 21 de Janeiro de 2014.

RESUMO

A representação de objetos reais em ambientes virtuais tem aplicações em diversas áreas, como cartografia, realidade misturada e engenharia reversa. A geração de tais objetos pode ser realizada a partir de ferramentas CAD (*Computer Aided Design*), ou por meio de técnicas de reconstrução de superfícies. Quanto mais simples o modelo 3D, mais fácil é processá-lo e armazená-lo. Porém, tais métodos podem gerar elementos virtuais muito detalhados, que podem trazer problemas no processamento desta malha resultante, devido ao grande número de arestas e polígonos que devem ser tratados na visualização. Desta forma, podem ser aplicados algoritmos de simplificação, de forma a eliminar polígonos da malha resultante, sem prejudicar a topologia da mesma, gerando uma malha mais leve e sem certos detalhes que podem ser irrelevantes ao usuário.

Portanto, este projeto teve como objetivos o estudo, a implementação e testes comparativos de algoritmos de simplificação de malhas geradas por um *pipeline* de reconstrução baseado em nuvens de pontos.. Este trabalho efetua a realização da etapa de simplificação, como complemento ao *pipeline* já desenvolvido por (ONO et al., 2012), que desenvolveu a reconstrução a partir de nuvem de pontos capturadas pelo Microsoft Kinect, utilizando-se do algoritmo de Poisson.

Palavras-chave: Reconstrução de superfícies, simplificação de malhas, multirresolução

ABSTRACT

The representation of real objects in virtual environments has applications in many areas, such as cartography, mixed reality and reverse engineering. The generation of these objects can be performed through two ways: manually, with CAD (Computer Aided Design) tools, or automatically, by means of surface reconstruction techniques. The simpler the 3D model, the easier it is to process and store it. However, this methods can generate very detailed virtual elements, that can result in some problems when processing the resulting mesh, because it has a lot of edges and polygons that have to be checked at visualization. Considering this context, it can be applied simplification algorithms to eliminate polygons from resulting mesh, without change its topology, generating a lighter mesh with less irrelevant details.

The project aimed the study, implementation and comparative tests of simplification algorithms applied to meshes generated through a reconstruction pipeline based on point clouds. This work proposes the realization of the simplification step, like a complement to the pipeline developed by (ONO et al., 2012), that developed reconstruction through cloud points obtained by Microsoft Kinect, and then using Poisson algorithm.

Keywords: Surface reconstruction, multiresolution, geometric modelling.

Lista de Figuras

Lista de Tabelas

Lista de Algoritmos

Sumário

Capítulo 1

INTRODUÇÃO

A Computação Gráfica sempre foi alvo de avançados estudos desde seus primórdios, especialmente devido a sua alta utilização nas diversas áreas de conhecimento. Mais especificamente, pode-se citar o estudo sobre reconstrução tridimensional de superfícies, que basicamente constitui a criação de modelos virtuais a partir de um objeto físico real. Tal estudo é muito utilizado nas produções audiovisuais e de entretenimento, que acabam utilizando destas tecnologias para poupar custos da criação de alguma figura ou cenário surreal ou fictícia em tamanho real. Tais técnicas, quando utilizadas, podem enriquecer produções cinematográficas, além de aumentar o realismo da cena em questão. Tudo isso envolve técnicas de reconstrução de superfícies, e a partir disso gerar modelos virtuais, resultando em malhas poligonais (normalmente triângulos). Tais aplicações podem ser utilizadas não só na produção audiovisual, como também na Medicina e na Engenharia. O processo de reconstrução tridimensional envolve um *pipeline* (conjunto de etapas), conforme mostrado na Figura 1.1 (SALEEM, 2004). O mesmo é normalmente formado pela captura do objeto a partir de algum dispositivo de aquisição e a subsequente criação da nuvem de pontos. A nuvem de pontos é então usada para reconstruir uma malha poligonal tridimensional, e posteriormente, pode ser feita a simplificação da mesma. (MADSEN et al., 2011).

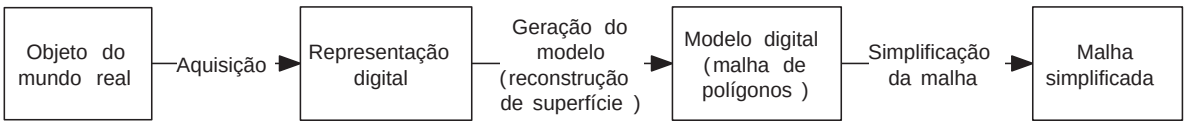
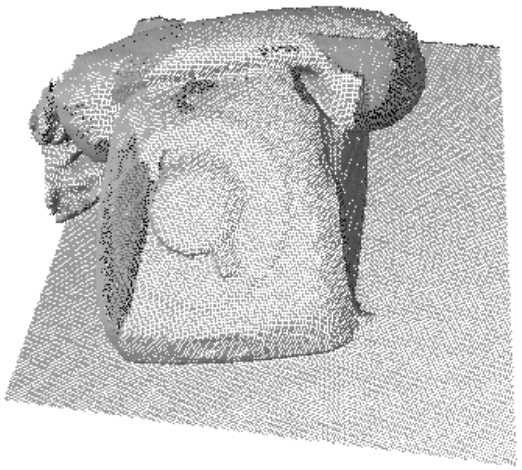
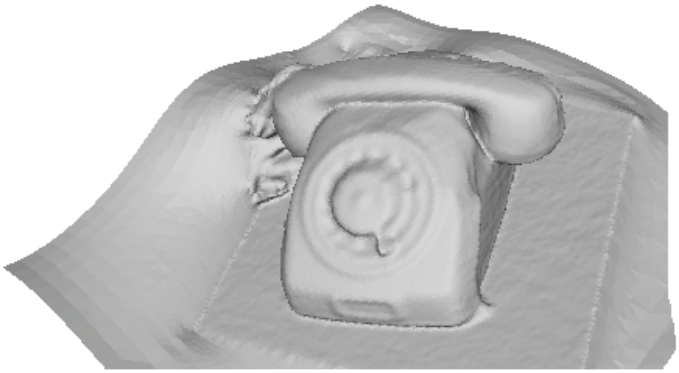


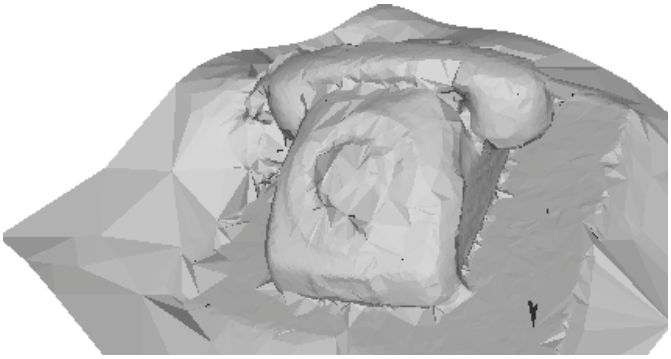
Figura 1.1: *Pipeline* de Reconstrução. Adaptado de (SALEEM, 2004).



(a) Nuvem de pontos



(b) Malha poligonal reconstruída



(c) Malha poligonal simplificada

Figura 1.2: Etapas do processo de reconstrução (ONO et al., 2012).

Malhas obtidas a partir de algum método de reconstrução podem conter muitos detalhes, resultando em um arquivo muito complexo que pode trazer problemas no processamento do mesmo. Tais detalhes podem ser irrelevantes ao espectador final, tendo em vista a possibilidade da variação da distância e a angulação do mesmo em relação ao elemento virtual. Desta forma, uma malha

detalhada é utilizada quando o objeto está próximo da câmera e aproximações mais simplificadas podem ser utilizadas conforme o objeto se distancia. A figura 1.3 ilustra um objeto em multirresolução renderizado em diferentes distâncias e angulações. É importante observar que o objeto mais distante constitui a malha com menos detalhes, caracterizando a importância da etapa de simplificação.

Figura 1.3: Modelo *Stanford Bunny* renderizado em diferentes níveis de detalhe. Da esquerda para a direita, 10.000, 20.000, 30.000, 50.000 e 2.000.000 vértices (PAULY; GROSS; KOBBELT, 2002).

Ono et al (2012) desenvolveram as primeiras etapas do *pipeline* proposto, de forma que é possível a captura da nuvem de pontos a partir do dispositivo Microsoft Kinect, e a aplicação do algoritmo de Poisson (ONO et al., 2012). A ênfase deste trabalho é a etapa final desse *pipeline* que trata da simplificação.

Considerando o contexto exposto, este trabalho focou-se no processo de simplificação de superfícies baseada em nuvens de pontos. As principais técnicas são apresentadas e comparadas, obtendo então o melhor algoritmo para complementar o *pipeline* proposto.

1.1 OBJETIVOS

1.1.1 Objetivos Gerais

Investigar os principais algoritmos de simplificação, de forma a descobrir qual a melhor solução para se complementar o *pipeline* de reconstrução de superfícies.

1.1.2 Objetivos Específicos

- Levantar os principais algoritmos de simplificação;
- Realizar suas implementações;
- Analisar as malhas simplificadas obtidas, considerando-se a quantidade de polígonos e vértices, bem como, a discrepância com relação à malha mais detalhada, visando a escolha do metódo de simplificação mais adequado.

1.2 ESTRUTUA DA MONOGRAFIA

A monografia está estruturada da seguinte forma:

- Capítulo 1: É apresentado aqui as etapas da reconstrução de superfícies, focando na etapa de simplificação de superfícies.
- Capítulo 2: Define-se aqui a simplificação de superfícies, assim como os tipos e métodos encontrados na literatura.
- Capítulo 3: Neste capítulo, são apresentadas as ferramentas utilizadas no trabalho, e como elas foram aplicadas para possibilitar a implementação dos métodos de simplificação e a subsequente verificação da eficiência de cada método.
- Capítulo 4: Os algoritmos implementados foram aplicados em diferentes malhas virtuais, gerando resultados que permitiram a verificação de cada método utilizado. Além disso, foi aplicada uma métrica, a fim de avaliar numericamente o erro de cada malha simplificada em relação a original.
- Capítulo 5: De acordo com os resultados obtidos, este capítulo mostra as conclusões obtidas ao final do projeto.

Capítulo 2

SIMPLIFICAÇÃO DE SUPERFÍCIES

2.1 Considerações Iniciais

Malhas virtuais obtidas a partir de algum método de reconstrução podem conter muitos detalhes, tornando-a muito pesada e difícil de ser processada. Isso pode trazer possíveis problemas durante o armazenamento até a inserção do objeto virtual em cena. Os algoritmos de simplificação tem como intuito reduzir o número de polígonos de uma malha virtual, mantendo a topologia da mesma, eliminando detalhes que não serão perceptíveis ao espectador, resultando em um arquivo mais leve.

2.2 Definição

O processo de simplificação de malhas é definido por Turk (1992) e Schroeder, Zarge e Lorensen (1992) como o problema de reduzir o número de faces em uma malha densa, mantendo-se a forma e a topologia da malha original, obtendo então uma malha em multirresolução, ou seja, em diferentes niveis de detalhes. A Figura 2.1 exibe um exemplo de representação em múltiplas resoluções.

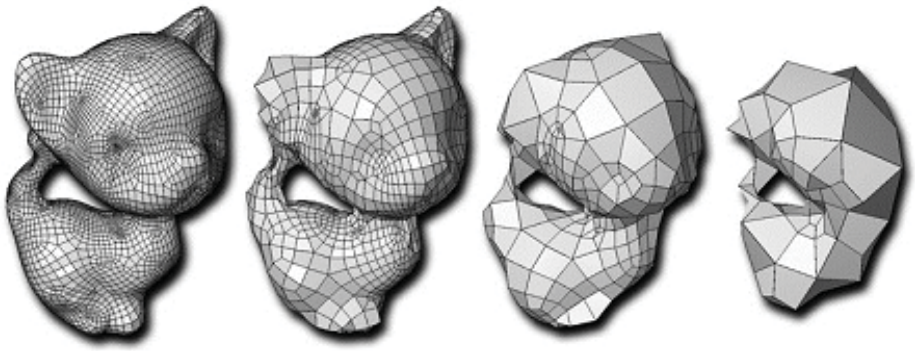
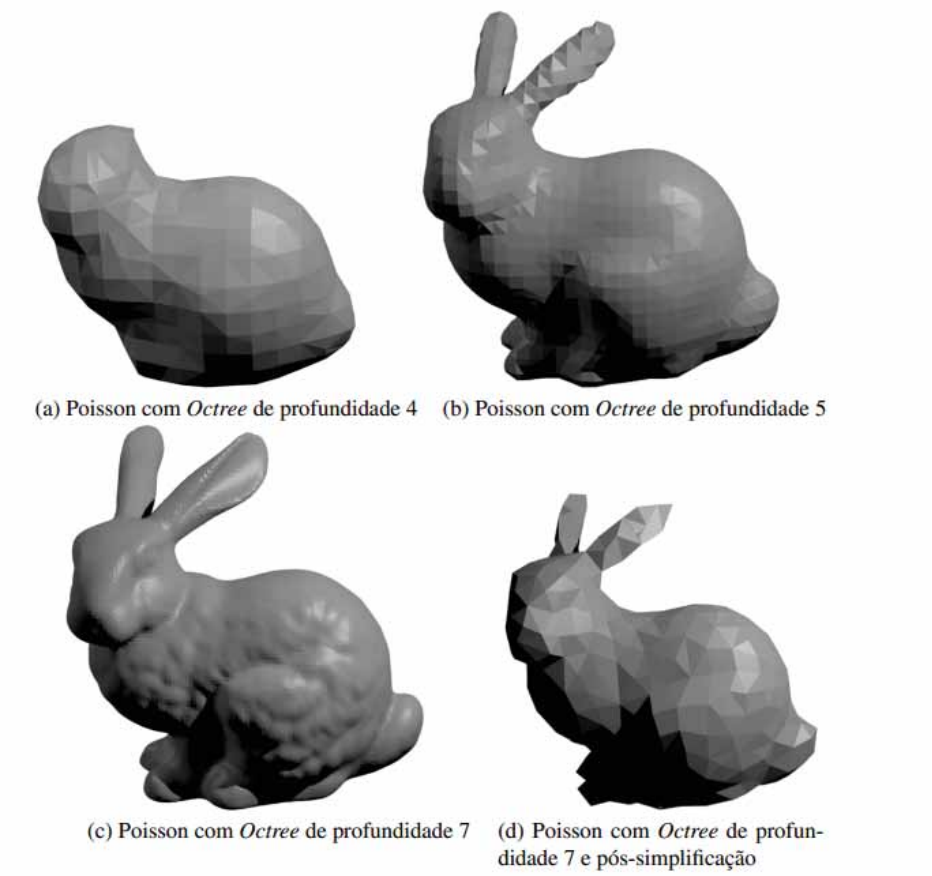


Figura 2.1: Representação de um gato em múltiplas resoluções(DANIELS et al., 2008).

Os algoritmos de reconstrução como o de Poisson traz consigo a possibilidade de controlar o nível de detalhe desejado, estabelecendo um valor para a profundidade; quanto menor o valor da profundidade, menos detalhes a malha resultante terá. Porém, as distorções causadas nas malhas devido a esses baixos valores estabelecidos são muito altas, prejudicando muito a topologia

da malha resultante, tornando inviável seu uso. A Figura 2.2 exibe um comparativo de uma malha reconstruída a partir do método de Poisson e com nível de detalhe baixo, e a mesma malha reconstruída a partir de Poisson com um nível de detalhe alto, porém aplicando um algoritmo de simplificação para controlar o nível de detalhe da mesma.



(a) Poisson com *Octree* de profundidade 4 (b) Poisson com *Octree* de profundidade 5

(c) Poisson com *Octree* de profundidade 7 (d) Poisson com *Octree* de profundidade 7 e pós-simplificação

Figura 2.2: Algoritmo de Poisson aplicado ao Coelho de Stanford em diferentes resoluções (STANFORD,2011)

É possivel observar que de (a) a (c), o coelho está reconstruído em diferentes níveis de detalhe através do controle do nível da *octree,* e em (d), foi aplicado um algoritmo de simplificação em um coelho reconstruído a partir de Poisson e um valor alto de profundidade, obtendo-se um resultado mais fiel em relação a topologia do objeto.

De acordo com Floriani et al. (1999), as principais abordagens de simplificação de superfícies podem ser divididas em métodos incrementais e métodos não incrementais.

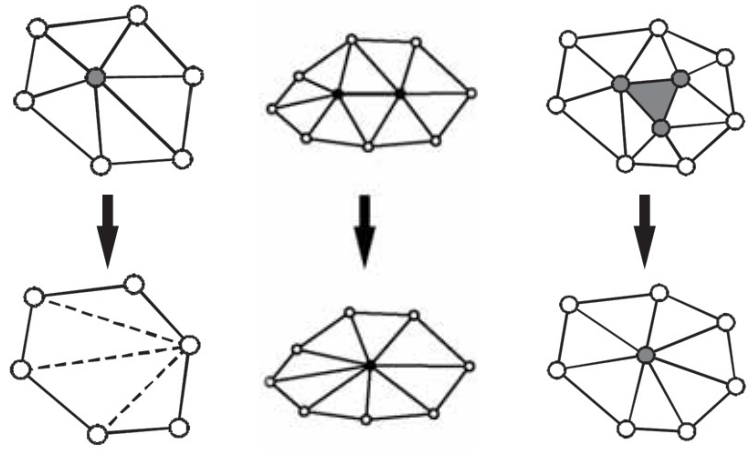
2.3 Métodos incrementais

Técnicas incrementais de simplificação de superfícies consistem em sequências de atualizações locais, em que cada atualização reduz o tamanho da malha e diminui a precisão da aproximação. Um algoritmo comum para a aplicação dessas técnicas é:

Algoritmo 2.1 Algoritmo geral para aplicações de simplificação incremental

Faça
    Seleciona elemento a ser removido / contraído
    Executa operação
    Atualiza malha
Até que: precisão/tamanho da malha esteja satisfatório

Tradicionalmente, existem três operações de atualização local: remoção de vértices (elimina duas faces), contração de arestas (elimina duas faces) e contração de triângulo (elimina quatro faces) (FLORIANI et al., 1999). Essas operações são ilustradas na figura 2.3.



(a) Remoção de vértices (b) Contração de arestas (c) Contração de triângulo

Figura 2.3: Operações de atualização local (FLORIANI et al., 1999).

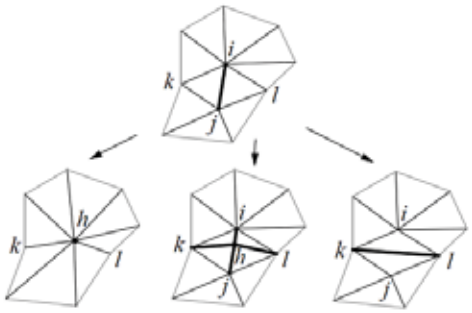
Hoppe et al. (1993) desenvolveram um método de simplificação que consiste na execução iterativa de contração de aresta (*collapse*), partição de aresta (*split*) e troca de aresta (*swap*), de modo a otimizar a função de energia, definida como $E(M) = Edist(M) + Erep(M) + Espring(M)$, onde

Edist: soma das distâncias dos pontos originais a M;

Erep: fator proporcional ao número de vértices em M;

Espring: soma dos comprimentos das arestas. (ONO et al., 2012)

A qualidade da aproximação (simplificação) é avaliada através dessa função de energia. A figura 2.4 ilustra a execução das três etapas em uma malha de exemplo. De acordo com Floriani et al. (1999), embora esse método produza resultados com alta qualidade e preserve a topologia da malha original, tem a desvantagem de ser muito demorado.



Contração Divisão Troca

Figura 2.4: Simplificação por otimização de função de energia (HOPPE et al., 1993).

Em 1996, Hoppe verificou a possibilidade de simplificar uma superfície apenas com a operação de contração de arestas. Nesse trabalho, descreveu um algoritmo que armazena a sequência de transformações inversas (divisão de aresta), possibilitando a aquisição de malhas simplificadas com qualquer número de faces, com precisão de ± 1, já que cada operação *collapse* remove duas faces da malha. Além de possibilitar multirresolução, o método também permite a execução de *geomorphing* (transição visual suave entre duas malhas em diferentes níveis de detalhe), exemplificada na figura 2.5 (HOPPE, 1996).

(a) $\alpha = 0,00$ (b) $\alpha = 0,25$ (c) $\alpha = 0,50$ (d) $\alpha = 0,75$ (e) $\alpha = 1,00$

Figura 2.5: Exemplo de *Geomorph*, de 500 faces (a) a 1.000 faces (e), de acordo com um parâmetro de transição α (HOPPE, 1996).

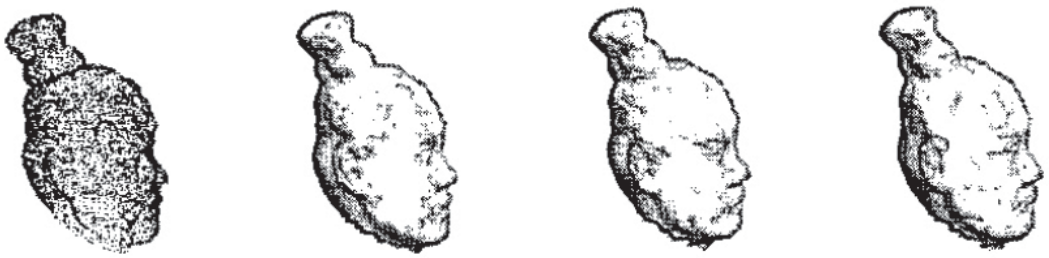
Nesse método, otimiza-se uma função de energia mais complexa, definida como
$E(M) = Edist(M) + Espring(M) + Escalar(M) + Edisc(M)$,
sendo que os dois primeiros valores são os mesmos de (HOPPE et al., 1993) e
Escalar(M): responsável por preservar os cantos da malha;
Edisc(M): responsável pela simplificação nas descontinuidades das malhas.

A abordagem de Lindstrom e Turk (1998) utiliza operações de contração de arestas mantendo-se o volume da superfície original e minimizando-se as mudanças ocorridas nos triângulos da malha e na área desses triângulos. A vantagem desse método é que, por não efetuar comparações com a malha original, é computacionalmente mais eficiente e consome menos memória em comparação com o método proposto por Hoppe (1996). Essa técnica é muito mais rápida e possibilita multirresolução, transmissão progressiva e *geomorphing (FLORIANI et al., 1999).*

2.4 Métodos não incrementais

Diversos algoritmos de simplificação utilizam técnicas não incrementais. Pode-se citar, por exemplo, métodos que utilizam junção de faces coplanares, *remeshing*, *clustering* e métodos baseados em *wavelets (FLORIANI et al., 1999)*.

O trabalho de Mingyi, Bangshu e Huajing (2004) descreve um algoritmo capaz de gerar malhas em diferentes resoluções a partir de nuvens de pontos, reduzindo o número de triângulos comparado com o algoritmo convencional Marching Cubes (LORENSEN; CLINE, 1987). A partir de um erro pré-estabelecido, é verificada se duas faces são coplanares através do cálculo da angulação entre elas; se a angulação for menor que o erro, verifica-se a coplanaridade, portanto, estas duas faces podem ser transformadas em uma. Este resultado possibilita simplificações na representação da malha, sendo que para um erro maior, obtêm-se uma malha mais simplificada. A figura 2.6 ilustra esse resultado. De acordo com Floriani et al. (1999), métodos que utilizam essa heurística são mais complexos, mas conseguem preservar a topologia original das descontinuidades da malha.



(a) Amostra (9076 pontos) (b) $\theta_{max} = 0°$ (16473 faces) (c) $\theta_{max} = 30°$ (6828 faces) (d) $\theta_{max} = 50°$ (3358 faces)

Figura 2.6: Superfície reconstruída em diferentes LODs com o método de Mingyi, Bangshu e Huajing (2004).

Lee et al (1998) apresentaram um algoritmo de simplificação que elimina vértices de acordo com uma função, e gera uma malha mais simplificada através do calculo de uma nova triangularização da malha. O algoritmo utiliza uma simplificação hierárquica para parametrizar a malha original sobre um domínio base contendo um pequeno número de triângulos, para depois melhorá-lo através de procedimentos de suavização baseado em subdivisões. O algoritmo é ilustrado na figura 2.7.

(a) Modelo original (b) Domínio base (c) Malha simplificada

Figura 2.7: Execução do método de Lee et al. (1998).

Lindstrom e Turk (1998) abordaram o método de Hoppe utilizando operações de contração de arestas mantendo-se o volume da superfície original e minimizando-se as mudanças ocorridas nos triângulos da malha e na área desses triângulos. Esta abordagem permite maior rapidez durante o processo computacional por não efetuar comparações com a malha original.

2.5 Algoritmos de Simplificação Estudados

Foram encontrados na literatura algoritmos de simplificação, cada um possuindo seu critério e suas peculiaridades. Tais métodos foram escolhidos por utilizarem critérios já utilizados pela OpenMesh, permitindo maior flexibilidade durante a implementação.

2.5.1 *Aspect Ratio*

Este método, proposto por (SCHROEDER; ZARGE; LORENSEN, 1992), avalia a distância dos vértices em relação ao plano médio da malha, como mostra a Figura 2.8 e também em relação a aresta relativa ao polígono em que ele se encontra.

Caso a distância seja menor ou igual a distância (erro) pré-estabelecido, o vértice deve ser eliminado.

Já em relação ao outro critério utilizado, usa-se a distância dos vértices até a aresta. Neste caso, o algoritmo determina a distância da linha definida pelos dois vértices que formam a aresta, como mostra a Figura 2.9. Se a distância do vértice a ser avaliado for menor que a linha em questão, o vértice pode ser eliminado.

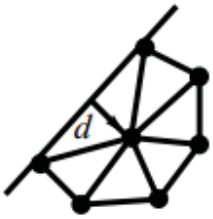


Figura 2.8: Distância do vértice a ser avaliado em relação a aresta (SCHROEDER; ZARGE; LORENSEN, 1992)

Figura 2.9: Distância do vértice em relação ao plano (SCHROEDER; ZARGE; LORENSEN, 1992)

Se um vértice é eliminado, o laço criado pela remoção do polígono correspondente deve ser triangulado, para evitar a presença de ocasionais buracos e falhas na malha simplificada. Com a triangulação finalizada, o vértice original e os respectivos triângulos são removidos da estrutura.

2.5.2 *Normal Deviation*

Hinker e Hansen(1993) propuseram um algoritmo muito utilizado até hoje em muitos *softwares* de edição e criação de malhas. O método avalia, através de uma função custo, se dois ou mais polígonos estão no mesmo plano, ou se a angulação entre eles é maior que um valor pré-estabelecido.

A redução dos polígonos é feita seguindo alguns passos. Primeiramente é feito o agrupamento de todos os polígonos que estão no mesmo plano, ou que apresentam uma angulação muito pequena, criando dessa forma uma lista segmentada para cada grupo e criando uma classificação para cada uma. Após isso, segmentos redundantes são eliminados, resultando em uma lista contendo apenas os pares de vértices correspondentes as bordas dos polígonos. Com isso, uma nova triangulação é feita, resultando no objeto simplificado, como mostrado na Figura 2.10

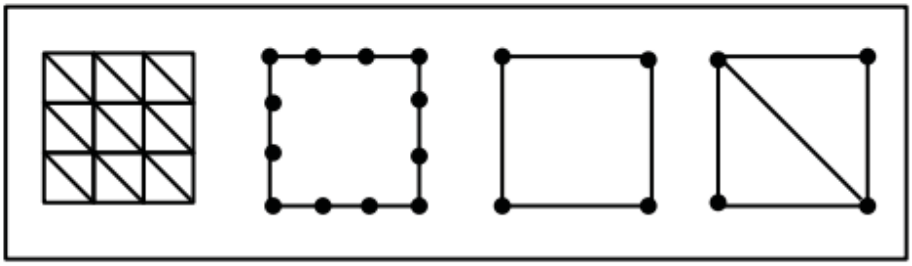


Figura 2.10: Fluxo de trabalho proposto por Hinker e Hansen (HINKER; HANSEN, 1993)

2.5.3 *Quadric*

Esta abordagem proposta por Garland e Heckbert (1997) traz uma função quádrica para definir o erro usado para a eliminação ou não de determinados vértices. Segundo os próprios autores, pode-se considerar uma diferente abordagem do trabalho de (LINDSTROM; TURK, 1998), pois após a verificação dos vértices, é feita a operação de contração de arestas referentes aos vértices analisados. (GARLAND; HECKBERT, 1997)

São necessárias traz 5 etapas para a elaboração deste método:

(a) – Computar a matriz quádrica para todos os vértices

(b) – Avaliar todos os pares válidos

(c) – Para cada par válido, verifica-se o novo vértice resultante da contração da aresta resultante do par de vértices analisados. O custo resultante dessa operação é definido como $v^{-t}(Q1+Q2)v1$, sendo v1 o novo vértice resultante da contração da aresta relativa ao par de vértices, cuja matrizes quádricas são definidas por Q1 e Q2.

(d) – Os valores encontrados na etapa c, são ordenados em ordem crescente, e colocados em uma lista.

(e) - A partir desta lista, os pares com menores custos sofrem contração, ocorrendo a eliminação dos dois vértices e a respectiva aresta. Os outros pares que contém v1 ou v2 são atualizados na lista.

2.5.4 *Edge Length*

Este algoritmo, desenvolvido por Lindstrom and Turk(1998), consiste em selecionar repetidamente uma aresta a partir de um custo mínimo, contrair esta aresta, e então calcular novamente o custo das arestas afetadas pela operação. Uma operação de contração de arestas seleciona um par de vértices {v0,v1} e substitui este par por um novo vértice {v}, ocasionando na substituição dos triângulos por arestas. As arestas e triângulos restantes relativas ao par de vértices {v0,v1} são, então, modificados para que o par de arestas seja substituído por {v}. A figura 2.11 ilustra o método

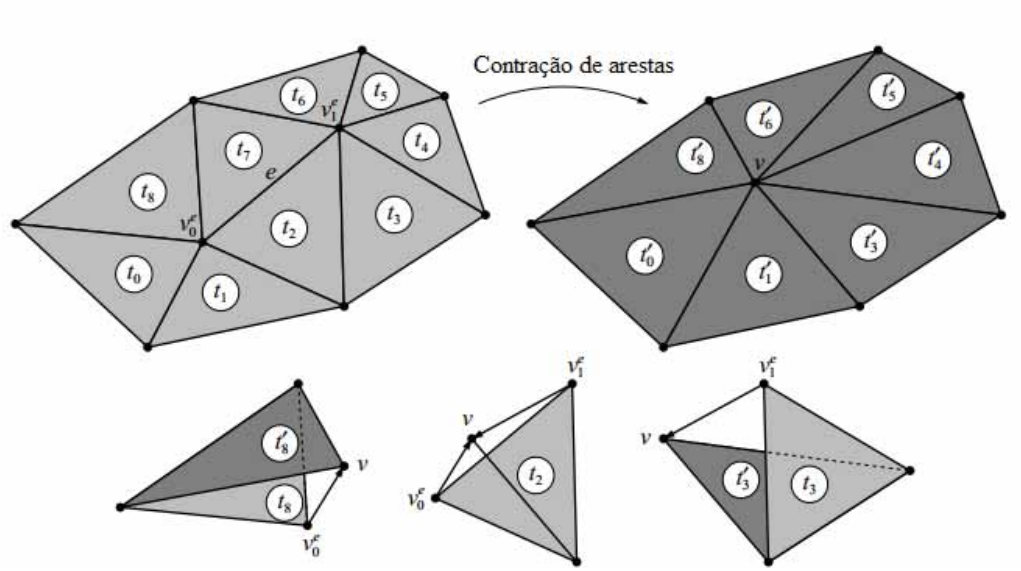


Figura 2.11: Operação de contração de arestas (LINDSTROM; TURK, 1998)

O método basicamente armazena todos os pares de vértices em uma fila ordenada, atribuindo para cada componente da fila um valor "custo". A partir deste valor custo, os primeiros pares de vértices da fila ordenada são testados, e caso respeitem o critério de eliminação, a operação de contração de aresta é feita, a partir do novo vértice {v}, como mostra a figura 19. A posição do vértice {v} pode ser encontrada de diferentes formas, como colocá-lo no ponto médio da aresta referente ao par de vértices {v0,v1}, ou efetuar um cálculo de forma que o novo vértice {v} resulte em uma distância pequena entre a malha simplificada e a malha original, o que resultaria na diminuição da distorção causada pela eliminação dos polígonos na malha (LINDSTROM; TURK, 1998).

2.6 Considerações Finais

Observou-se nesse capítulo a definição e a aplicação de algoritmos de simplificação, assim como exemplo de métodos e estudos já feitos na literatura. Os algoritmos *Aspect Ratio*, *Normal Deviation*, *Quadric* e *Edge Length* se destacaram por ser bastante citado na literatura, além de já existir primitivas na biblioteca *OpenMesh* que permitem implementar os critérios trazidos por estes métodos.

Capítulo 3

MATERIAIS E MÉTODOS

3.1 Considerações Iniciais

Através da biblioteca *OpenMesh*, foram aplicados diferentes algoritmos de simplificação encontrados na literatura como descrito no capítulo anterior, em diferentes malhas virtuais. Posteriormente, a ferramenta Metro foi utilizada para avaliar o erro médio da malha simplificada em relação a original, a fim de estabelecer uma métrica para avaliar qual método se mostrou mais eficiente. O software aberto *MeshLab* foi utilizado para a visualização das malhas resultantes.

3.2 Biblioteca *OpenMesh*

Utilizou-se a biblioteca *OpenMesh*, desenvolvida em C++, sendo então própria para ser compilada e utilizada no ambiente Microsoft Visual Studio 2010. A utilização da biblioteca torna-se importante, pois a mesma traz funcionalidades adequadas que possibilitam o tratamento das estruturas de uma malha poligonal assim como primitivas para a remoção dos vértices e polígonos. (BOTSCH et al., 2002).

A biblioteca trata a malha de forma a armazenar os pares de vértices correspondentes a todas as arestas do elemento virtual em uma lista computacional. Cada método de simplificação estabelece um critério para a remoção ou não de determinado par de vértice, ou seja, é dado um valor numérico para cada elemento da lista. Posteriormente, essa lista é ordenada, e os pares de vértices que estão no topo da lista são removidos, pois possuem o maior erro numérico. O valor limite para remoção depende do erro estabelecido, ou seja, do grau de simplificação desejado. A ordenação da lista depende do critério estabelecido por cada método de simplificação.

Além disso, dispõe-se das primitivas necessárias para implementar a avaliação da malha estabelecida por cada método, como por exemplo a distância da aresta em relação ao plano médio da malha, a angulação entre uma aresta e outra, etc.

A biblioteca utiliza como parâmetros de entrada e saída, arquivos com extensão *obj*, que podem ser facilmente obtidos de arquivos de outros formatos conhecidos através de um software de edição 3D, como o *MeshLab*. A partir disso, foram construídos algoritmos que utilizam das ferramentas trazidas pela biblioteca, possibilitando então a configuração das características de cada método, assim como o valor do erro e da proporção de simplificação que se deseja efetuar.

3.3 *Metro*

Uma comparação geral de malhas simplificadas é algo que exige um critério muito cuidadoso, tendo em vista os diferentes métodos que os algoritmos de simplificação usam para efetuar a redução das arestas e polígonos. Na verdade, muitos métodos de simplificação não retornam medidas do erro utilizado para efetuar a simplificação.

Outras abordagens permitem ao usuário estabelecer o erro máximo desejado, mas não retornam o erro real estimado após a simplificação. (CIGNONI; ROCCHINI; SCOPIGNO, 1998)

Portanto, uma ferramenta para "medir" a diferença de uma malha simplificada em relação a original é necessária e permite verificar com mais precisão qual método trabalha melhor a simplificação.

A ferramenta *Metro* é uma ferramenta que compara numericamente duas malhas, que descrevem a mesma superfícies em diferentes níveis de detalhe. A ferramenta não exige conhecimento do método de simplificação utilizado para reconstruir a malha reduzida.

A verificação é feita com base na distância numérica entre as duas malhas, definida como a distância entre seções correspondentes da malha simplificada e a original. Dado um ponto p e uma superfície S, temos a distância $e(p,S)$ definida por:

$$e(p,S) = \min d(p,p')$$

Sendo p' pertencente a S, e $d()$ sendo a distância euclidiana entre dois pontos. Desta forma, a distância entre duas superfícies S1 e S2 é definida como:

$$E(S1,S2) = \max e(p, S2)$$

Sendo p pertencente a S1. É bom salientar que esta definição não é simétrica, ou seja, a distância de S1 para S2 é diferente de S2 para S1, sendo tal detalhe devidamente tratado pela Metro ao se verificar valores negativos após o cálculo. A Figura 3.1 mostra as distâncias verificadas pela Metro.

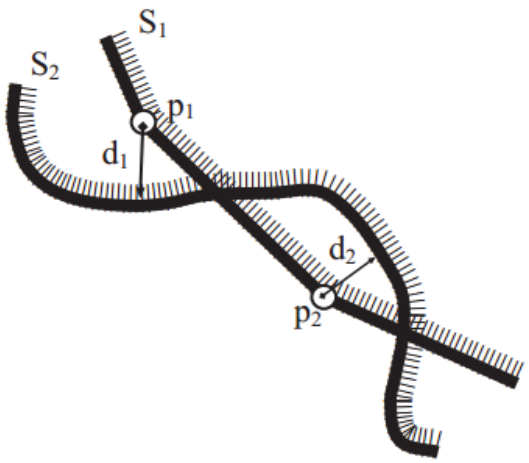


Figura 3.1: Verificação da distância entre duas superfícies (CIGNONI; ROCCHINI; SCOPIGNO, 1998)

A ferramenta aceita como parâmetros de entrada malhas no formato *PLY*, que podem ser facilmente obtidas através do software *MeshLab*, a partir das malhas em formato *obj* tratadas pela *OpenMesh*.

3.4 *MeshLab*

O software MeshLab consiste em uma aplicação para visualização de malhas, onde um objeto tridimensional pode ser armazenado e carregado em diferentes formatos além de ser facilmente manipulável. Através do mouse, é possível percorrer toda a malha simplesmente clicando e arrastando o objeto, permitindo a sua visualização. (CIGNONI et al., 2008)

Com a malha carregada, o usuário pode trabalhar com várias ferramentas disponíveis pelo sofware, como filtros de suavização e a exportação da malha para diversos formatos. O uso do software mostra-se importante, pois os algoritmos de reconstrução retornam apenas a malha em si, ou seja, apenas as arestas e os polígonos, configurando uma espécie de "esqueleto". A partir desse "esqueleto", o MeshLab é capaz de aplicar algoritmos de preenchimento, permitindo uma visualização mais coerente, como mosta a Figura 3.2. A Figura 3.3 mostra a utilização do software para percorrer o ambiente 3D onde se localiza a malha.

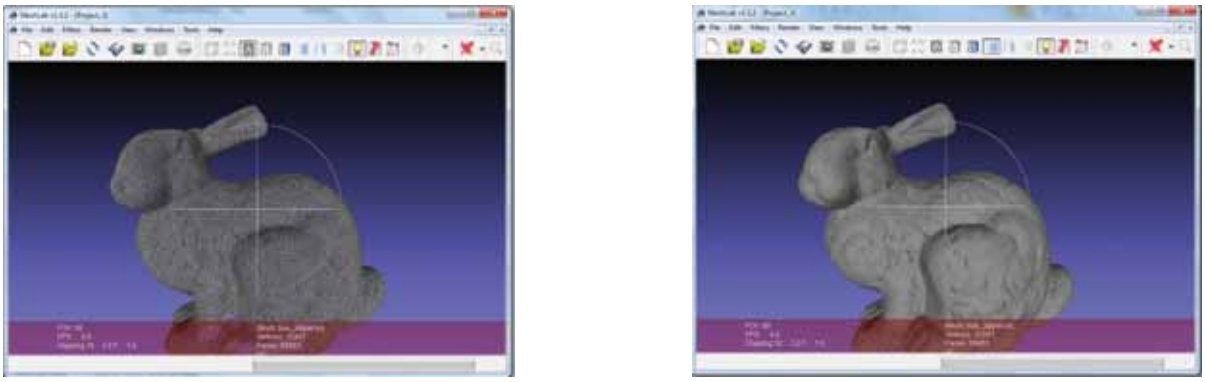


Figura 3.2: Exemplo de visualização a partir do *MeshLab*. Na primeira figura, tem-se a malha contendo apenas as arestas e os polígonos. Na segunda figura, é possivel observar o resultado final da renderização.

Figura 3.3: Exemplo de manipulação de uma malha no ambiente 3D

Além disso, o software traz a opção de importar e exportar malhas para diferentes formatos, o que é muito importante, tendo em vista o fato da *OpenMesh* trabalhar com malhas no formato *obj*, e a Metro trabalhar com malhas no formato *PLY*.

3.5 Reconstrução e simplificação da superfície

Iniciou-se agora a investigação dos métodos de simplificação, a fim de verificar qual algoritmo complementa de forma mais eficiente o *pipeline* já desenvolvido. Através dos recursos trazidos pela *OpenMesh*, foi possível criar um algoritmo geral para a implementação dos algoritmos, como segue abaixo.

Algoritmo 3.1 Algoritmo usado no desenvolvimento do software implementado

Ler malha original
Registrar a malha ao módulo de remoção de polígonos
Registrar o módulo de remoção de polígonos ao método
Estabelecer um valor numérico ao erro
Inicializar módulo remoção de polígonos
Estabelecer um valor numérico para a quantidade resultante de vértices
Inicializar remoção de polígonos
Escrever malha resultante

Através do algoritmo acima, os métodos de simplificação descritos na seção 2.5 foram implementados, resultando em um arquivo executável para cada método, após a compilação utilizando o Visual Studio 2010. A utilização do executável deve ser feita via linha de comando, e também segue um modelo geral, como segue abaixo.

nome_do_executavel.exe malha_original.obj malha_simplificada.obj

O arquivo "nome_do_executável.exe" se refere ao executável relativo a um determinado método de simplificação. Caso o executável relativo ao método *AspectRatio*, por exemplo, tenha nome

"*AspectRatio.exe*", e deseja-se aplicar o mesmo no Coelho de Stanford, cujo arquivo tem nome "*bunny.obj*" (STANFORD, 2011), tem-se o seguinte:

```
AspectRatio.exe bunny.obj out_bunny.obj
```

O arquivo "out_*bunny.obj*" será a malha simplificada resultante.

Além disso, os métodos propostos foram validados, aplicando uma métrica nas malhas simplificadas a fim de analisar qual a deformação existente em relação a malha original, e se o processo de simplificação manteve a topologia e as características do elemento virtual. Para isso, podem ser aplicadas métricas quantitativas, que calculam o erro baseado na distância calculada a partir dos polígonos de cada malha, assim como métricas qualitativas, que retornam o erro com base na topologia do próprio objeto. Neste caso, foi aplicada uma métrica quantitativa trazida pela ferramenta *Metro*, que retorna o erro com base na diferença da distância de pontos inseridos nas duas malhas, em relação aos vértices e arestas dos polígonos de cada malha. (CIGNONI; ROCCHINI; SCOPIGNO, 1998)

Ao final da etapa de simplificação, uma malha poligonal é gerada para o usuário no formato *OBJ*, podendo ser importada pela maior parte de *softwares* modeladores 3D como o *MeshLab* possibilitando sua visualização.

3.6 Ambiente experimental

Os materiais (componentes de hardware e software) utilizados são descritos nas seções seguintes.

3.6.1 Hardware

- 1 Computador pessoal: Dell Latitude D630
  - Processador: Intel(R) Core(TM) Duo CPU T7100. 1.80GHz, com placa de vídeo integrada
  - Memória RAM: 2 GB;
  - Disco rígido: SATA 120 GB;

3.6.2 Software

- Sistema operacional: Windows 7 Ultimate SP1;
- Ambiente de desenvolvimento: Microsoft Visual Studio 2010;
- Sistema de configuração de projetos: CMake 2.8.9;
- Biblioteca OpenMesh 2.3.1
- Metro
- MeshLab

3.7 Considerações Finais

Neste capítulo foram trazidas as ferramentas utilizadas para a implementação dos algoritmos de simplificação propostos, assim como para o estabelecimento de uma métrica para avaliar a deformação causada na malha virtual em cada método. A ferramentas OpenMesh traz várias primitivas que permitem a correta manipulação da malha, em especial a simplificação da mesma, permitindo o controle do nível de simplificação, retornando uma malha simplificada no formato *obj.* A *Metro* consite em avaliar o erro médio de uma malha simplificada em relação a original, permitindo uma verificação mais consistente, além da visual, de qual método mostra-se mais eficiente.

Capítulo 4

EXPERIMENTOS

4.1 Considerações Iniciais

Os algoritmos de simplificação propostos foram testados usando-se nuvens de pontos de repositórios de domínio público (Repositório 3D de Stanford (STANFORD, 2011)). Foram utilizados o modelo "Stanford *Bunny*" (STANFORD, 2011), e também ao modelo *Armadillo* (STANFORD, 2011). Além disso, os algoritmos também foram aplicados em nuvens provenientes do Kinect a partir do trabalho de (ONO et al., 2012). As malhas utilizadas no experimento podem ser vistas nas Figuras 4.1 e 4.6.

4.2 Malhas de Stanford

Para efetuar os experimentos, inicialmente utilizou-se 2 malhas, ja citadas e também utilizadas durante as etapas inicias do projeto: coelho de Stanford , contendo 35947 vértices; Armadillo de Stanford, contendo 132640 vértices (STANFORD, 2011).

Figura 4.1: Malhas de Stanford utilizadas nos experimentos. Respectivamente, tem-se o *Armadillo* e o Coelho de Stanford

A utilização destes modelos é interessante pelo fato dos 2 possuírem diferentes níveis de detalhe, possibilitando a análise da eficácia dos algoritmos em diferentes situações.

Por uma questão de padronização, o número de vértices de todas as malhas foi reduzido para 10000. Foram obtidas, então, as malhas resultantes como mostrado nas figuras 4.2 à 4.5, para cada método proposto:

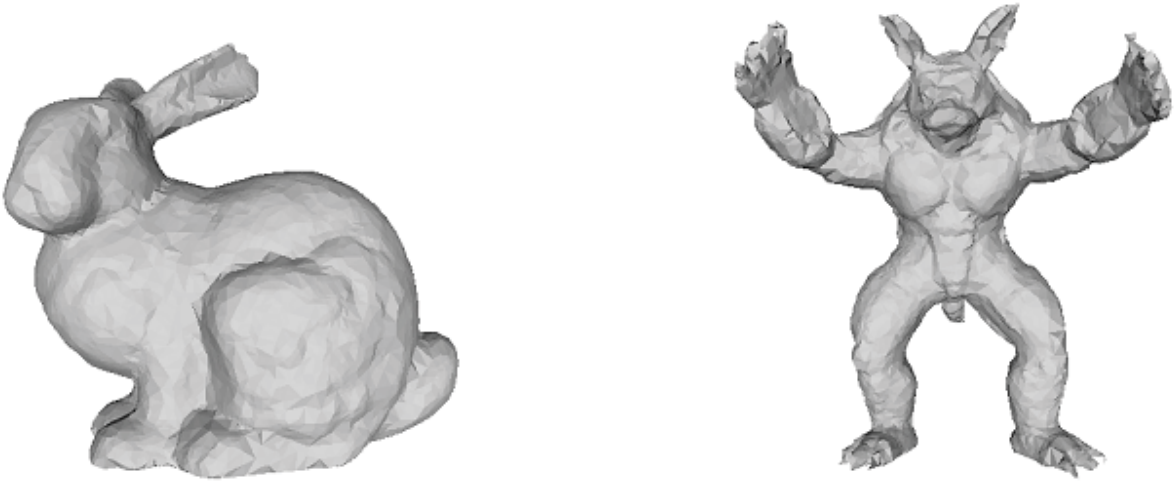


Figura 4.2: Simplificação através do método *Aspect Ratio* (10000 vértices)

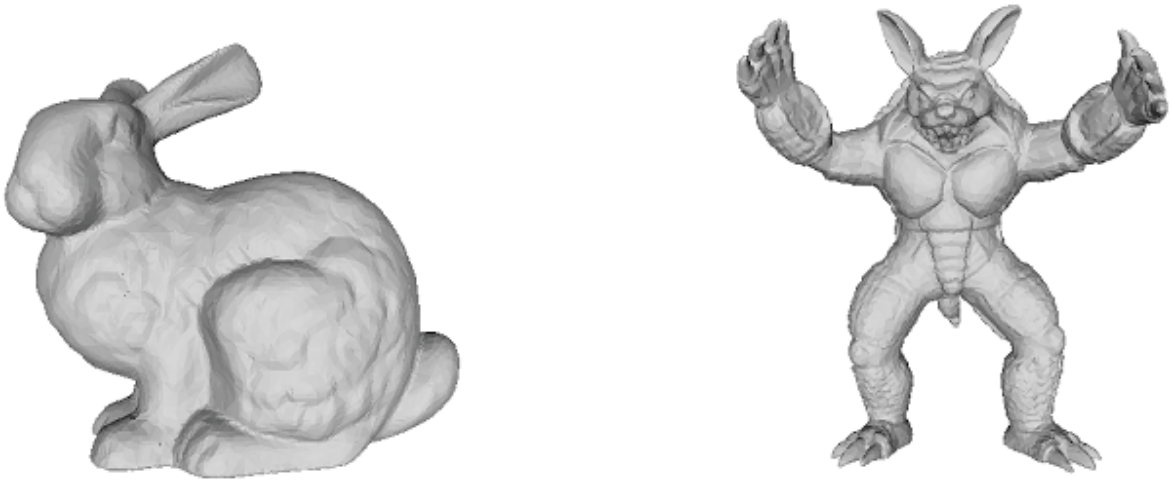


Figura 4.3: Simplificação através do método *Quadric* (10000 vértices)

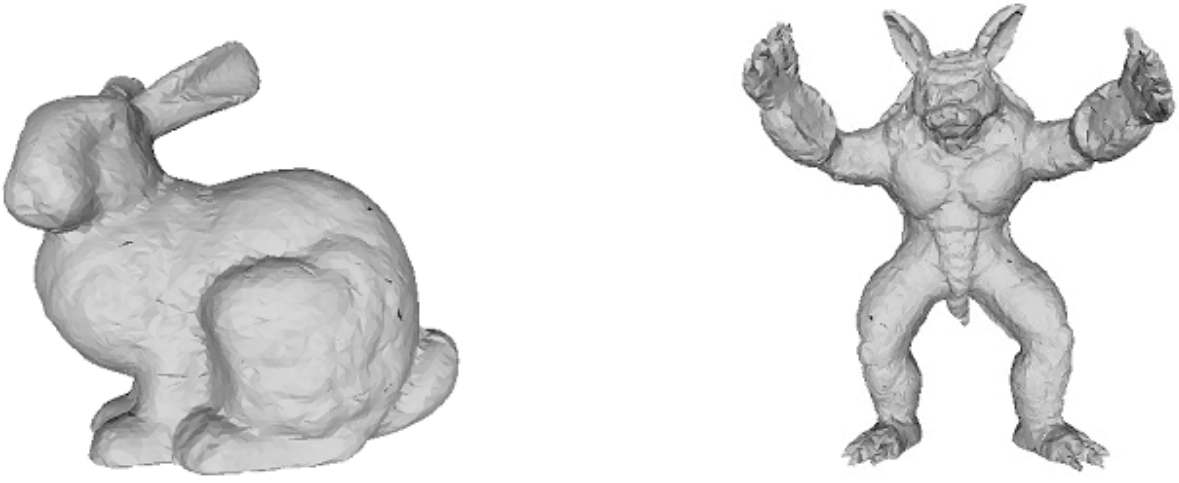


Figura 4.4: Simplificação através do método *EdgeLength* (10000 vértices)

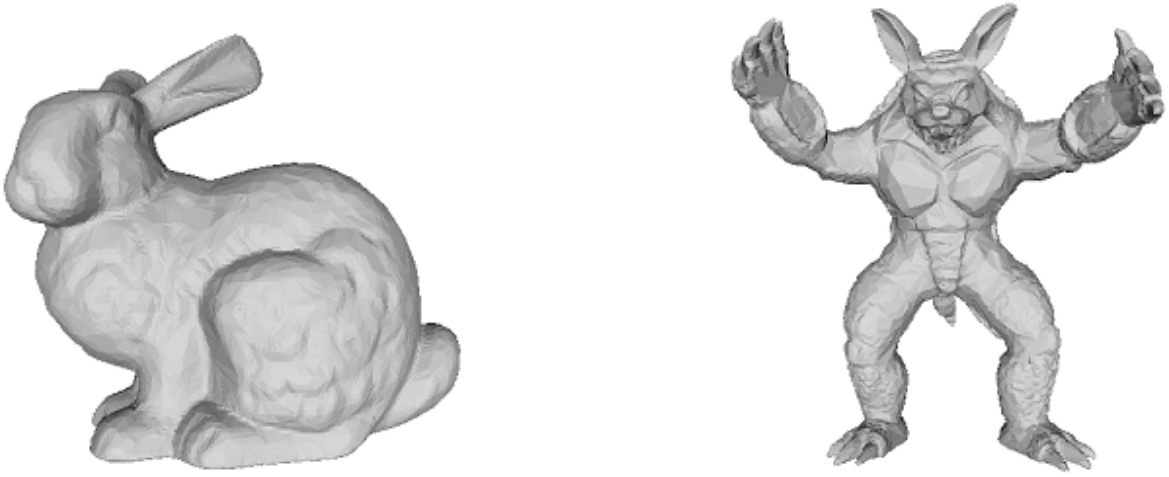


Figura 4.5: Simplificação através do método *Normal Deviation* (10000 vértices)

A fim de avaliar o erro de cada malha simplificada em relação as originais, foi aplicada a ferramenta *Metro*, retornando dessa forma o erro médio para cada método, como mostra a tabela 4.1.

Tabela 4.1: Erro médio de cada método para as malhas de Stanford simplificadas

Malha Original	**Aspect Ratio**	**Normal Deviation**	**Quadric**	**EdgeLength**
Coelho	0.000101	0.000044	0.000027	0.000086
Armadillo	0.168120	0.104369	0.051805	0.119594

Vale salientar que o erro médio é calculado a partir da malha simplificada até a malha original, possibilitando então a análise da discrepância entre detalhes mais específicos do objeto e superfícies mais planas ou ausentes de detalhes específicos.

Através deste experimento, foi possível observar que todos os métodos mostraram-se eficientes quando aplicados em uma malha com poucos detalhes, no caso, o coelho. Através da *Metro*, foram obtidos baixos valores para os erros médios de cada malha simplificada em relação a original.

Já o *Armadillo*, que no caso é uma malha mais complexa, os métodos *Quadric* e *Normal Deviation* foram mais eficientes. Tal resultado pode ser observado tanto visualmente, quanto a partir dos valores numéricos obtidos pela *Metro*, que mostraram-se menores para esses métodos.

4.3 Kinect

Atualmente, existem diversos sensores de profundidade, com métodos de capturas diferentes, vantagens e desvantagens, custos e limitações(MADSEN et al., 2011). Nesta categoria, inclui-se o

Kinect, um periférico desenvolvido pela Microsoft para o videogame Xbox 360 (SOLONY; ZEMCIK, 2011), o qual tornou-se um importante sensor 3D de luz estruturada, devido ao seu baixo custo, confiabilidade e velocidade de captura (SMISEK; JANCOSEK; PAJDLA, 2011). Com este dispositivo é possível obter os dados da superfície a ser escaneada e reconstruída, a partir de sua respectiva nuvem de pontos, e com isso, alcançar o ponto de partida para a reconstrução virtual do objeto, utilizando-se algum algoritmo de reconstrução. O Kinect foi usado para obter a captura das nuvens de pontos, e partir destes dados, foi feita a aplicação do algoritmo de Poisson, obtendo-se então uma superfície reconstruída, configurando as primeiras etapas do pipeline que foram desenvolvidas por (ONO et al., 2012).

4.4 Malhas Obtidas a Partir do Uso do Kinect

Após os experimentos com as malhas de Stanford, foram aplicados os algoritmos de simplificação em malhas obtidas através do Microsoft Kinect. Esta etapa é muito importante, pois será verificado de fato a eficiência dos algoritmos, quando utilizados para complementar o *pipeline* já desenvolvido. As malhas aqui utilizadas foram obtidas através do algoritmo de Poisson aplicado em nuvem de pontos obtidas pelo Kinect, através do trabalho de (ONO et al., 2012)

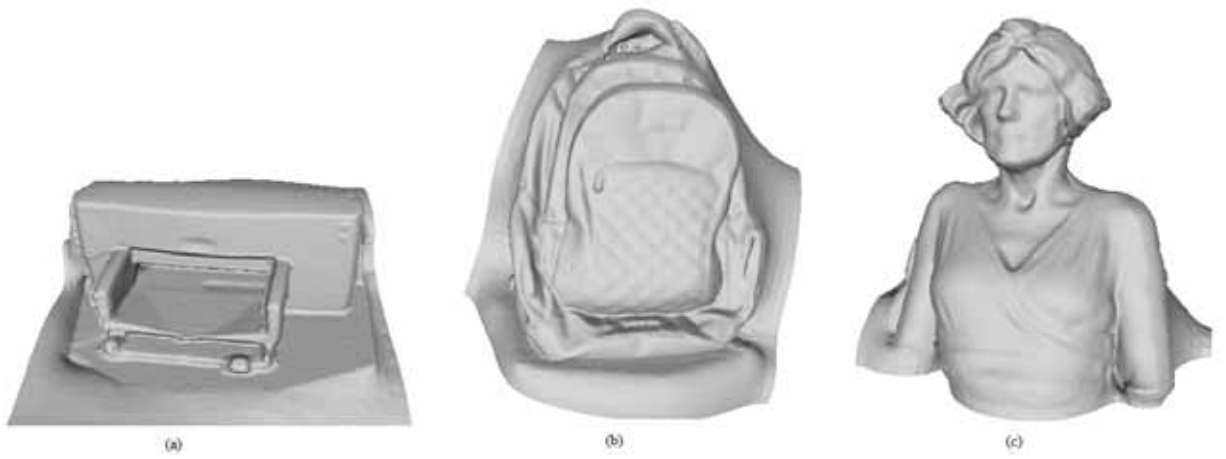


Figura 4.6: Malhas obtidas através do Kinect, utilizados nesta etapa dos experimentos. (ONO et al., 2012)

A Figura 4.6 (a) trata-se de uma impressora, e possui 114403 vértices. A Figura 4.6 (b) trata-se de uma mochila, e possui 112510 vértices, e a 4.6 (c) é a reconstrução de uma pessoa, possuindo 98186 vértices. Estas malhas também foram reduzidas para 10000 vértices, para que um padrão seja criado, e a validação dos métodos seja mais coerente. Abaixo segue o resultado da aplicação de cada método nas malhas mostradas acima.

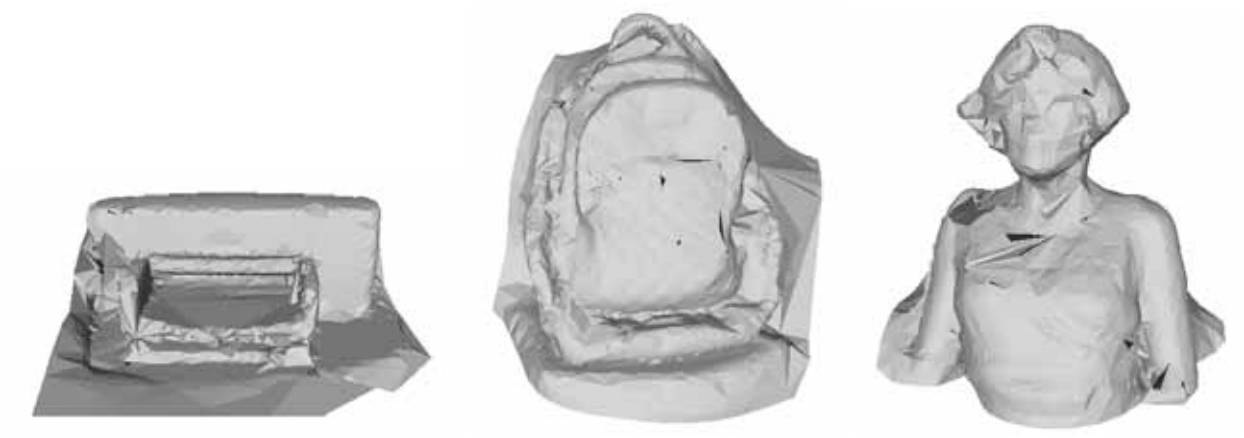


Figura 4.7: Aplicação do algoritmo *Aspect Ratio* (SCHROEDER; ZARGE; LORENSEN, 1992)

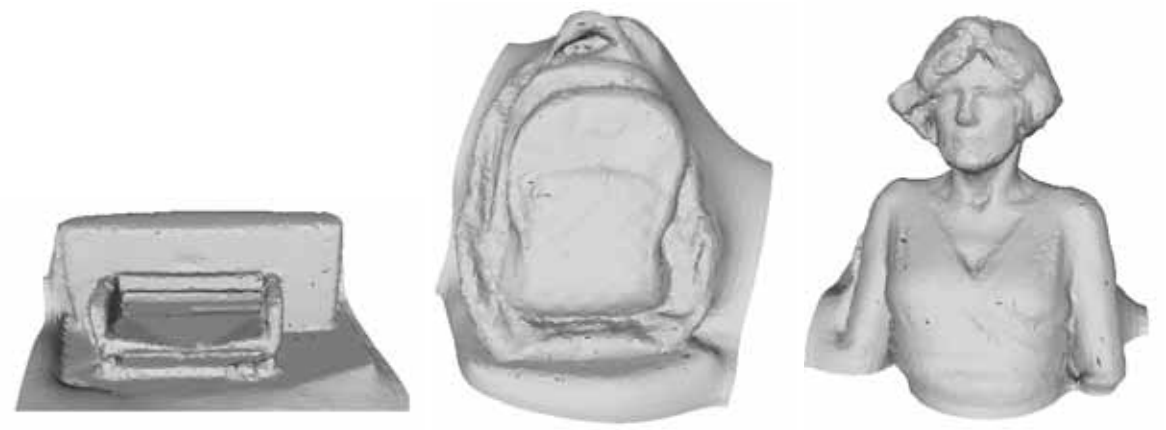


Figura 4.8: Aplicação do algoritmo *Edge Length* (LINDSTROM; TURK, 1998)

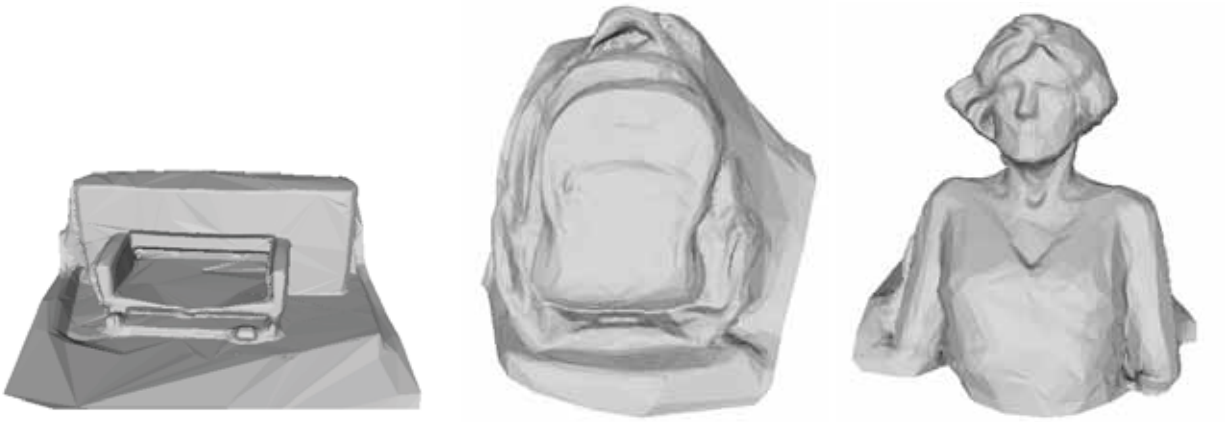


Figura 4.9: Aplicação do algoritmo *Normal Deviation* (HINKER; HANSEN, 1993)

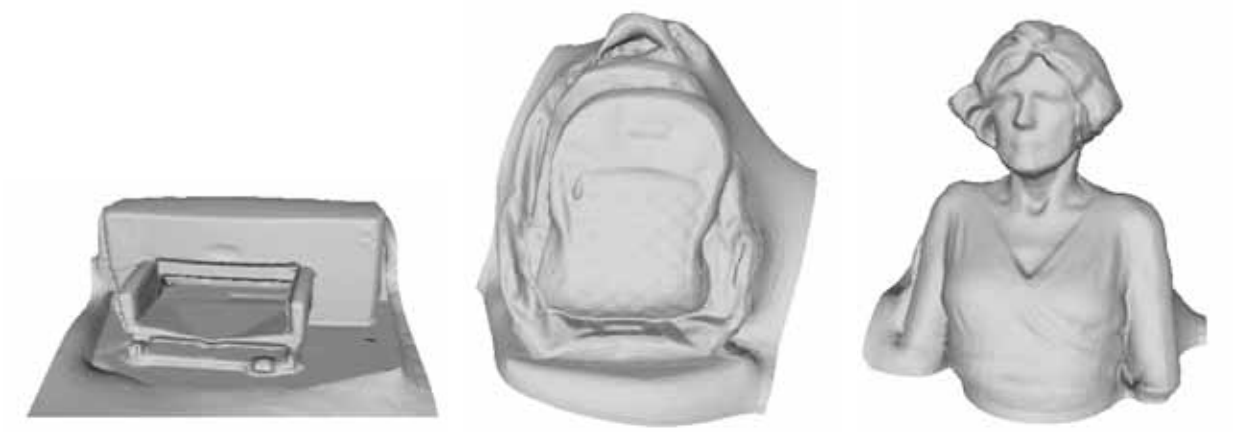


Figura 4.10: Aplicação do algoritmo *Quadric* (GARLAND; HECKBERT, 1997)

Para a validação dos resultados, a ferramenta *Metro* foi aplicada nas malhas simplificadas, e o erro médio em relação as malhas originais foi calculado, como mostra a tabela 4.2.

Tabela 4.2: Erro médio de cada método para as malhas obtidas pelo Kinect simplificadas

Malha Original	**Aspect Ratio**	**Normal Deviation**	**Quadric**	**EdgeLength**
Impressora	0.001539	0.000734	0.000060	0.000191
Mochila	0.001728	0.000554	0.000075	0.000219
Pessoa	0.002132	0.000749	0.000082	0.000205

É possível observar que, visualmente e numericamente falando, o algoritmo *Quadric* mostra-se mais eficiente para todas as malhas, podendo facilmente ser escolhido como o principal método para complementar o *pipeline* de reconstrução já desenvolvido.

4.5 Considerações finais

Foi feito a aplicação dos algoritmos implementados em duas malhas obtidas do repositório público de Stanford, sendo que cada uma possui um nível de detalhe diferente. Tal fato permitiu verificar como os algoritmos trabalham muitos ou poucos detalhes de uma malha. Além disso, foram aplicados também em malhas obtidas pelo Kinect, a fim de verificar o comportamento dos algoritmos no pipeline desenvolvido por (ONO et al., 2012). A partir dos experimentos, foi possível observar a superioridade do algoritmo *Quadric* para todas as malhas utilizadas. O algoritmo apresentou os melhores resultados visuais, além de apresentar o menor erro numérico retornado pela *Metro*. Desta forma, o método pode ser facilmente integrado ao *pipeline* de reconstrução já desenvolvido, configurando o acréscimo da etapa de simplificação no mesmo.

Capítulo 5

CONCLUSÕES

Combinou-se o *pipeline* com outros métodos de simplificação, a fim de verificar qual a melhor solução a se combinar com a etapa de reconstrução. As superfícies com menor detalhamento (como foi o caso dos experimentos com o Coelho de Stanford) apresentaram bons resultados quando utilizados todos os métodos, sendo possível observar a preservação da topologia nos resultados visuais, e os baixos erros numéricos retornados pela Metro. Nas malhas mais complexas (como o *Armadillo*), obteve-se uma distorção um pouco maior, especialmente quando utilizado o método *Aspect Ratio*. Porém, os resultados obtidos a partir dos métodos *Quadric* e *Normal Deviation* foram melhores, pois obteve-se uma malha simplificada muito pouco distorçidas em relação as originais, além de apresentarem baixos erros numéricos quando aplicada a *Metro*. Isso comprova a superioridade dos dois métodos em relação aos demais, podendo eles ser usados tanto para superfícies simples, como complexas. Além disso, a combinação dos algoritmos com as etapas do *pipeline* já desenvolvidas mostrou-se válida, ao verificar o bom resultado visual e numérico da simplificação das malhas obtidas pelo Kinect.

Desta forma, é possível concluir que o *pipeline* apresentado pode ser utilizado para reconstruir objetos em diferentes níveis de detalhe, que podem ser inseridos em aplicações de realidade virtual e aumentada, sendo recomendada a utilização do método *Quadric* para a etapa de simplificação.

5.1 Trabalhos Futuros

A partir deste conjunto de etapas desenvolvido, uma série de novos trabalhos podem ser feitos para complementar o *pipeline* implementado. Pretende-se estudar e implementar novas métricas, para avaliar as malhas simplificadas, principalmente do ponto de vista qualitativo. Desta forma, é necessário ainda aplicar alguma métrica para verificar a diferença em relação a topologia das malhas, de forma a se obter uma validação completa dos algoritmos, do ponto de vista numérico e visual.

Referências Bibliográficas

BOTSCH, M.; STEINBERG, S.; BISCHOFF, S.; KOBBELT, L. Openmesh - a generic and efficient polygon mesh data structure. *1st OpenSG Symposium*, 2002.

CIGNONI, P.; CALLIERI, M.; CORSINI, M.; DELLEPIANE, M.; GANOVELLI, F.; RANZUGLIA, G. Meshlab: an open-source mesh processing tool. *Eurographics Italian Chapter Conference*, 2008.

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

DANIELS, J.; SILVA, C.; SHEPHERD, J.; COHEN, E. Quadrilateral mesh simplification. In: *ACM Transactions on Graphics (TOG)*. [S.l.: s.n.], 2008. v. 27, p. 148.

FLORIANI, L.; CIGNONI, P.; PUPPO, E.; SCOPIGNO, R. *Surface Simplification Algorithms Overview*. 1999. Disponível em: <http://www.webalice.it/giorgio.faconti/EG99tutorials/T2/lod3.pdf>.

GARLAND, M.; HECKBERT, P. Simplification using quadric error metrics. *SIGGRAPH '97 Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, p. 209–216, 1997.

HINKER, P.; HANSEN, C. Geometric optimization. In: *In Proc. Visualization 93*. [S.l.: s.n.], 1993. v. 189-195, p. 189–195.

HOPPE, H. Progressive meshes. In: *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*. [S.l.: s.n.], 1996. p. 99–108.

HOPPE, H.; DeRose, T.; DUCHAMP, T.; McDonald, J.; STUETZLE, W. Mesh optimization. In: *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*. New York, NY, USA: ACM, 1993. (SIGGRAPH '93), p. 19–26. ISBN 0-89791-601-8. Disponível em: <http://doi.acm.org/10.1145/166117.166119>.

LEE, A. W. F.; SWELDENS, W.; SCHRODER, P.; COWSAR, L.; DOBKIN, D. MAPS: multiresolution adaptive parameterization of surfaces. In: *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*. New York, NY, USA: ACM, 1998. (SIGGRAPH '98), p. 95–104. ISBN 0-89791-999-8. Disponível em: <http://doi.acm.org/10.1145/280814.280828>.

LINDSTROM, P.; TURK, G. Fast and memory efficient polygonal simplification. In: *Visualization'98. Proceedings*. [S.l.: s.n.], 1998. p. 279–286.

LORENSEN, W. E.; CLINE, H. E. Marching cubes: A high resolution 3d surface construction algorithm. *SIGGRAPH*, v. 21, 1987.

MADSEN, R.; KEHL, C.; NICA, V.; BARBULESCU, A.; TSESMELIS, T. *3D Scanning and Reconstruction of Large Scale Environments*. [S.l.], 2011.

MINGYI, H.; BANGSHU, X.; HUAJING, Y. Multi-resolution surface reconstruction. In: *Image Processing, 2004. ICIP'04. 2004 International Conference on*. [S.l.: s.n.], 2004. v. 3, p. 1971–1974.

ONO, J. H. P.; SEMENTILLE, A. C.; CALDEIRA, M. A. C.; MARAR, J. F. Poisson surface reconstruction with local mesh simplification. In: GULIATO, T. V. D. (Ed.). *Workshop of Works in Progress (WIP) in SIBGRAPI 2012 (XXV Conference on Graphics, Patterns and Images)*. Ouro Preto, MG, Brazil: [s.n.], 2012. Disponível em: <http://www.decom.ufop.br/sibgrapi2012/index.php/call/wip>.

PAULY, M.; GROSS, M.; KOBBELT, L. Efficient simplification of point-sampled surfaces. In: *Visualization, 2002. VIS 2002. IEEE*. [S.l.: s.n.], 2002. p. 163–170.

SALEEM, W. *A flexible framework for learning-based surface reconstruction*. Tese (Doutorado) — Masters thesis, Computer Science Department, University of Saarland, Postfach 15 11 50, 66041 Saarbrücken, 2004.

SCHROEDER, W. J.; ZARGE, J. A.; LORENSEN, W. E. Decimation of triangle meshes. *SIGGRAPH Comput. Graph.*, v. 26, n. 2, p. 65–70, jul. 1992. ISSN 0097-8930. Disponível em: <http://doi.acm.org/10.1145/142920.134010>.

SMISEK, J.; JANCOSEK, M.; PAJDLA, T. 3D with kinect. In: *I IEEE Workshop on Consumer Depth Cameras for Computer Vision*. [S.l.: s.n.], 2011.

SOLONY, M.; ZEMCIK, P. Scene reconstruction from kinect motion. In: *Conference and Competition Student EEICT*. [S.l.: s.n.], 2011.

STANFORD. *The Stanford 3D Scanning Repository*. University of Stanford, 2011. Disponível em: <http://graphics.stanford.edu/data/3Dscanrep/>.

TURK, G. Re-tiling polygonal surfaces. In: *Proceedings of the 19th annual conference on Computer graphics and interactive techniques*. New York, NY, USA: ACM, 1992. (SIGGRAPH '92), p. 55–64. ISBN 0-89791-479-1. Disponível em: <http://doi.acm.org/10.1145/133994.134008>.

APÊNDICE A - RESUMO SUBMETIDO AO XXV CONGRESSO DE INICIAÇÃO CIENTÍFICA DA UNESP - APROVADO PARA A 2ª FASE

FERNANDES, F. V. L., SEMENTILLE, A.C. *Simplificação de Malhas Visando Representações em Multirresolução.* XXV Congresso de Iniciação Científica da Unesp. Bauru, SP, Brasil: 2013.

XXV Congresso de Iniciação Científica da UNESP
I Fórum Internacional de Iniciação Científica da UNESP

Simplificação de Malhas Visando Representações em Multirresolução

Fernando Vinicius Lima Fernandes, Antonio Carlos Sementille

Universidade Estadual Paulista – Unidade de Bauru – Faculdade de Ciências

Introdução

Em produções cinematográficas, televisivas e de entretenimento, tem se tornado muito comum a inserção de elementos virtuais tridimensionais nas cenas que compõem o conteúdo gerado. Quando os modelos virtuais correspondem a elementos reais, pode-se usar a reconstrução tridimensional. Esta reconstrução normalmente depende de uma série de etapas, denominada *pipeline* de reconstrução. Este *pipeline* consiste na captura e obtenção dos dados físicos do objeto a ser modelado e a aplicação de um algoritmo de reconstrução, resultando em uma malha poligonal. Posteriormente, pode ser efetuada a simplificação desta malha[1]. A etapa de simplificação mostra-se importante em alguns casos, pois a malha pode ter sua resolução dependente, por exemplo, da distância da mesma até o observador da cena virtual. Neste caso, teria-se a necessidade de múltiplas resoluções de uma mesma malha (multirresolução).

Considerando este contexto, o presente trabalho, tem como objeto de estudo, os algoritmos de simplificação.

Objetivos

Estudar, implementar e testar métodos de simplificação de malhas poligonais e verificar qual o melhor método para complementar o *pipeline* de reconstrução.

Material e Método

Foi utilizada a biblioteca OpenMesh e a programação foi realizada em linguagem C++. Os métodos implementados foram: eliminação de vértices que estão no mesmo plano[2]; eliminação de vértices distantes do plano[3]; eliminação de arestas maiores que um valor pré-estabelecido[4]; e eliminação de vértices segundo uma função quadrática[5].

Resultados e Discussão

Os métodos foram aplicados em diferentes malhas obtidas de diferentes métodos de captura. A Figura 1 traz um exemplo dos resultados. Para efetuar uma comparação quantitativa dos resultados, utilizou-se a ferramenta Metro em todas as malhas. A ferramenta calcula o erro com base na distância de determinados pontos da malha simplificada em relação à superfície da malha original[6]. Os métodos implementados trouxeram bons resultados visuais e numéricos, com o método da função quadrática mostrando-se mais eficiente, pois preservou mais detalhes do coelho, além de apresentar o menor erro.

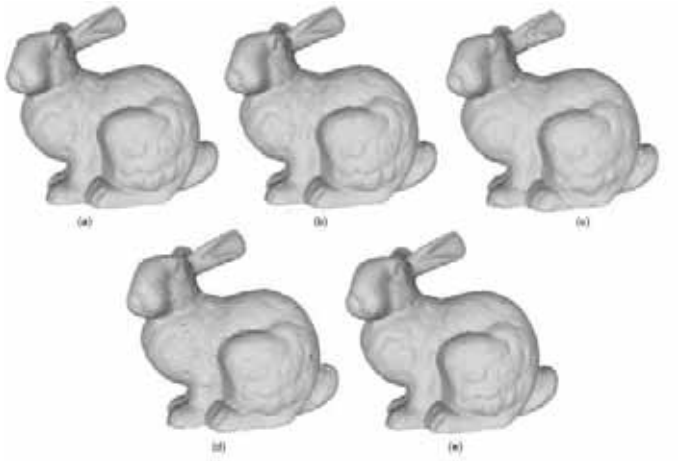


Figura 1. Resultado do processo de simplificação do coelho de Stanford. (a) Malha original contendo 35497 vértices; (b) vértices coplanares; (c) vértices distantes do plano; (d) eliminação de arestas; (e) função quadrática. Todas as malhas simplificadas (b) a (e) contêm 10000 vértices.

Conclusões

Todos os métodos mostraram-se eficientes e apresentaram bons resultados. Pretende-se ainda aplicar uma métrica para analisar a fidelidade da malha simplificada quanto à topologia do modelo, além da verificação do esforço computacional de cada método.

Agradecimentos

Agradecimentos pela utilização do laboratório SACI/DCo/FC, e ao CNPq pela bolsa concedida.

Financiamento

Bibliografia

[1]Saleem, W. A flexible framework for learning-based surface reconstruction, Computer Science Department, University of Saarland, Postfach 15 11 50, 66041 Saarbrücken, **2004**.
[2]Hinker, P.; Hansen, C. Geometric Optimization, Los Alamos, Advanced Computing Laboratory, **1993**.
[3]Schroeder, W. J.; Zarge, A. J.; Lorensen, W. E. Decimation of Triangle Meshes, NY, General Electric Company Schenectady, **1992**.
[4]Lindstrom, P.; Turk, G. Fast and memory efficient polygonal simplification, IEEE Visualization, p. 279-286, **1998**.
[5]Garland, M.; Heckbert, P.S. Simplification Using Quadric Error Metrics, Carnegie Mellon University, **1997**.
[6]Cignoni, P. ; Rocchini, C. ; Scopigno, R. Metro: measuring error on simplified surfaces, Blackwell Publishers, vol. 17(2), p. 167-174, **1998**.