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

Jorge Henrique Piazentin Ono

UM MÉTODO PARA RECONSTRUÇÃO DE SUPERFÍCIES BASEADO EM NUVEM DE PONTOS VISANDO REPRESENTAÇÕES EM MULTIRRESOLUÇÃO

UNESP

2012

Jorge Henrique Piazentin Ono

UM MÉTODO PARA RECONSTRUÇÃO DE SUPERFÍCIES BASEADO EM NUVEM DE PONTOS VISANDO REPRESENTAÇÕES EM MULTIRRESOLUÇÃO

Prof. Dr. Antonio Carlos Sementille
Prof. Dr. Marco Antonio Corbucci Caldeira

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

2012

Jorge Henrique Piazentin Ono

UM MÉTODO PARA RECONSTRUÇÃO DE SUPERFÍCIES BASEADO EM NUVEM DE PONTOS 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. Marcos Antônio Cavenaghi
UNESP - Bauru

Bauru, 29 de Outubro de 2012.

Dedico este trabalho à minha mãe, não apenas pelo amor, confiança e incentivo, mas por estar presente em todos os momentos de minha vida e nunca me deixar desistir de meus objetivos.

AGRADECIMENTOS

Agradeço primeiramente a Deus.

Aos professores Antonio Carlos Sementille e Marco Antonio Corbucci Caldeira o apoio, amizade e incontáveis reuniões e revisões de trabalhos.

A todos os professores que tive durante minha formação e, principalmente, à minha mãe, os valores que me ensinou e a atenção em minha educação básica.

À família, o incentivo e a segurança durante todo este período.

Aos amigos, o carinho e companheirismo.

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 de duas maneiras: manualmente, com ferramentas CAD (*Computer Aided Design*), ou de forma automática, por meio de técnicas de reconstrução de superfícies. Quanto mais simples o modelo 3D, mais fácil é processá-lo e armazená-lo. Métodos de reconstrução em multirresolução são capazes de gerar malhas poligonais em diferentes níveis de detalhe e, para melhorar o tempo de resposta de uma aplicação, objetos distantes do observador podem ser representados de forma mais simples, enquanto modelos mais detalhados são utilizados apenas em objetos próximos. Este trabalho apresenta uma nova abordagem para a reconstrução de superfícies em multirresolução, particularmente interessante para dados ruidosos e de baixa definição, por exemplo, nuvens de pontos provenientes do sensor Microsoft Kinect.

Palavras-chave: Reconstrução de superfícies, multirresolução, modelagem geométrica.

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 in 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. Multiresolution reconstruction methods can generate polygonal meshes in different levels of detail and, to improve the response time of a computer program, distant objects can be represented with few details, while more detailed models are used in closer objects. This work presents a new approach to multiresolution surface reconstruction, particularly interesting to noisy and low definition data, for example, point clouds captured with Kinect sensor.

Keywords: Surface reconstruction, multiresolution, geometric modelling.

Lista de Figuras

Lista de Tabelas

Lista de Algoritmos

Sumário

Capítulo 1

INTRODUÇÃO

Segundo Saleem (2004), o problema da Reconstrução de Superfícies pode ser dividido em três etapas principais, executadas em uma ordem fixa, e comumente representadas por um *pipeline*. A entrada do *pipeline* é um objeto do mundo real, que é amostrado para alguma representação digital (nuvem de pontos, camadas de imagens bidimensionais, entre outras) . Geralmente, a representação gerada é uma malha de triângulos, que pode ser simplificada, em uma etapa posterior de pós-processamento. Cada um destes passos – aquisição, geração de modelo e simplificação da malha – ainda pode ser dividido em sub-passos. A Figura 1.1 ilustra este *pipeline* de uma forma mais geral. Um exemplo do processo completo de reconstrução, com os dados de entrada (nuvem de pontos), reconstrução e simplificação é apresentado na figura 1.2.

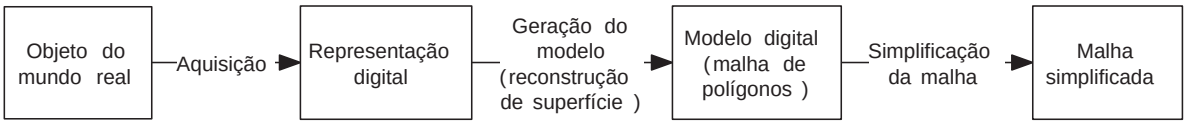


Figura 1.1: *Pipeline* de Reconstrução. Adaptado de (SALEEM, 2004).

(a) Nuvem de pontos (b) Malha poligonal reconstruída (c) Malha poligonal simplificada

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

A ênfase deste trabalho são as duas etapas finais desse *pipeline*: a geração de modelos e sua simplificação. Existem diversos desafios nestas duas etapas, como por exemplo: conectar/aproximar corretamente os pontos obtidos da fase de aquisição, processar uma grande quantidade de dados (nuvem de pontos de alta densidade) e gerar malhas em diferentes níveis de detalhes, a fim de atender às necessidades de diferentes aplicações.

A geração de modelo é, basicamente, uma questão de montagem de uma superfície sobre a coleção de dados adquiridos. Existem diferentes abordagens para este problema. Alguns métodos tentam montar uma superfície diretamente sobre a nuvem de pontos, através de *splines* ou quádricas. A própria nuvem de pontos pode ser triangularizada, gerando uma malha cujos vértices são os pontos amostrados. Outras abordagens podem ser utilizadas, por exemplo aproximar a superfície com funções implícitas ou representá-la com estruturas de dados volumétricas.

Malhas geradas por triangularização direta ou por métodos volumétricos podem ter uma distribuição ruim de seus vértices. Um outro aspecto a ser considerado é que, se a nuvem de pontos é muito densa, a malha poligonal gerada pode conter informações irrelevantes para usuários comuns. Técnicas de simplificação podem ser aplicadas de forma a minimizar dados desnecessários, mantendo as características mais importantes do modelo. Diversas vantagens são obtidas com essa simplificação. Quanto menor for a malha, maior será a facilidade em armazená-la, transmití-la e processá-la. Um outro aspecto, particularmente interessante à este projeto é a simplificação da malha para fins de visualização, onde, para assegurar um processamento rápido, uma malha pode ser gerada e armazenada em vários níveis de detalhe (*level of detail* ou LOD) e resoluções (multirresolução) (GERSTNER, 2002; BOTSCH; KOBBELT, 2001). Para melhorar a performance de renderização, é comum a geração de diversas versões de um modelo em vários níveis de detalhe (LOD). Uma malha detalhada é utilizada quando o objeto está próximo da câmera e aproximações mais simplificadas podem ser utilizadas conforme o objeto se distancia. Como mudanças instantâneas entre

malhas de diferentes LODs são visíveis, transições suavizadas com *geomorphing* podem ser aplicadas em malhas com diferentes resoluções (CLARK, 1976). A figura 1.3 ilustra um objeto em multirresolução renderizado em diferentes distâncias.

Figura 1.3: Modelo Davi (Michelangelo) renderizado em diferentes níveis de detalhe. Da esquerda para a direita, 10.000, 20.000, 60.000, 200.000 e 2.000.000 vértices (PAULY; GROSS; KOBBELT, 2002).

Considerando o contexto exposto, este trabalho investiga o processo de reconstrução e simplificação de superfícies baseada em nuvens de pontos. As principais técnicas são apresentadas e comparadas. Aproveitando-se das vantagens de dois algoritmos, o método de reconstrução de Poisson (KAZHDAN; BOLITHO; HOPPE, 2006) e a simplificação de contração de arestas (HOPPE, 1996), foi desenvolvido um *pipeline* capaz de gerar malhas em multirresolução preservando-se a topologia do objeto real com total controle do nível de simplificação (redução de polígonos).

1.1 Objetivos

1.1.1 Objetivos Gerais

A partir da investigação de técnicas de reconstrução de superfícies, elaborar e implementar um algoritmo de reconstrução aplicado à multirresolução, visando a geração de várias representações, variando o seu nível de detalhamento e número de polígonos.

1.1.2 Objetivos Específicos

Elaborar e implementar um algoritmo de reconstrução tridimensional aplicado à representação de superfícies em multirresolução, utilizando a linguagem C++ e o *Microsoft Kinect* como sensor para captura dos dados tridimensionais. O resultado deste trabalho será um sistema de reconstrução com as seguintes funcionalidades:

- Integração e controle do *Kinect*;

- Criação da nuvem de pontos;
- Reconstrução tridimensional em vários níveis de detalhe;
- Visualização da malha resultante.

O foco deste trabalho será a implementação do algoritmo de reconstrução de superfícies, portanto será dada ênfase ao seu estudo e embasamento matemático, enquanto para os outros módulos será utilizada a biblioteca auxiliar PCL e CGAL, agilizando a implementação de um sistema funcional e tornando o projeto exequível no prazo estipulado (de março a outubro de 2012, correspondendo às disciplinas de Projeto e Implementação de Sistemas I e II).

1.2 Estrutura da Monografia

Esta monografia está estruturada da seguinte maneira:

- Capítulo 2: apresenta os dispositivos de aquisição de dados 3D mais conhecidos, dando especial atenção ao sensor Micrisoft Kinect, utilizado neste trabalho;
- Capítulo 3: descreve os principais algoritmos de reconstrução de superfícies;
- Capítulo 4: trata dos métodos de simplificação de superfícies, responsáveis pela criação dos objetos em multirresolução;
- Capítulo 5: aborda o método desenvolvido;
- Capítulo 6: descreve como foi implementado *pipeline* de reconstrução;
- Capítulo 7: expõe os resultados obtidos através de experimentos com reconstrução e simplificação de superfícies;
- Capítulo 8: apresenta as conclusões deste trabalho baseadas na pesquisa e no desenvolvimento da aplicação.

Capítulo 2

DISPOSITIVOS DE AQUISIÇÃO DE DADOS 3D

Diversas técnicas de aquisição de dados tridimensionais foram desenvolvidas, motivadas, principalmente, pela medicina e engenharia. Atualmente, diversos sensores 3D estão se tornando cada vez mais populares, movidos pela interação homem-computador e pelos jogos digitais.

2.1 Considerações iniciais

Dispositivos de aquisição de dados 3D, também conhecidos como *scanners* 3D, são ferramentas que analisam objetos do mundo real e coletam dados sobre sua geometria. Diversos métodos de captura podem ser empregados nesses sensores, por exemplo, técnicas estereoscópicas, táteis, magnéticas e ópticas (MADSEN et al., 2011; SALEEM, 2004).

Este capítulo apresentará, primeiramente, os principais métodos de captura. Em seguida, a ferramenta de luz estruturada *Microsoft Kinect*, utilizada nos experimentos deste trabalho, é detalhada.

2.2 Principais Dispositivos de aquisição 3D

Os principais tipos de dispositivos de aquisição 3D atuais são:

- Câmeras Estereoscópicas;
- *Scanners* táteis;
- Câmeras *time of flight*;
- *Scanners* de superfícies de contornos;
- Radares e sonares;

- Luz estruturada.

Uma das técnicas mais antigas de aquisição 3D são as câmeras estereoscópicas, que utilizam duas ou mais lentes separadas, aproximadamente, pela distância dos olhos (6,35 cm), simulando a visão binocular humana (MADSEN et al., 2011). Para recuperar as informações de profundidade da cena, é utilizada a triangulação, sabendo-se a posição das imagens em um ponto no espaço. Uma grande vantagem deste sistema é a possibilidade de utilizar a câmera tanto em ambientes abertos como fechados, dado que a aquisição dos dados de profundidade é baseada em imagens RGB. Entretanto, as lentes devem ser cuidadosamente calibradas, o que torna o custo do equipamento bem elevado (MADSEN et al., 2011; SALEEM, 2004). A câmera estereoscópica binocular STH-MDCS3-VARX, produzida pela *SRI International Company of America*, é apresentada na figura 2.1.

Figura 2.1: Câmera Binocular STH-MDCS3-VARX (XU; CHEN; GAO, 2011).

Scanners táteis possuem mais limitações que outros equipamentos, devido ao processo de escaneamento ser físico. Entretanto, podem ser utilizados para capturar dados de profundidade com um nível de detalhamento muito alto. O equipamento "*Roland PIX-30 Tactile Scanner*", por exemplo, tem uma precisão de aproximadamente 0,04 mm (MULLER, 1997; BÖHLER; MARBS, 2002). A Máquina de Medição de Coordenadas (*Coordinate Measuring Machine* - CMM), apresentada na figura 2.2, é um exemplo de *scanner* tátil.

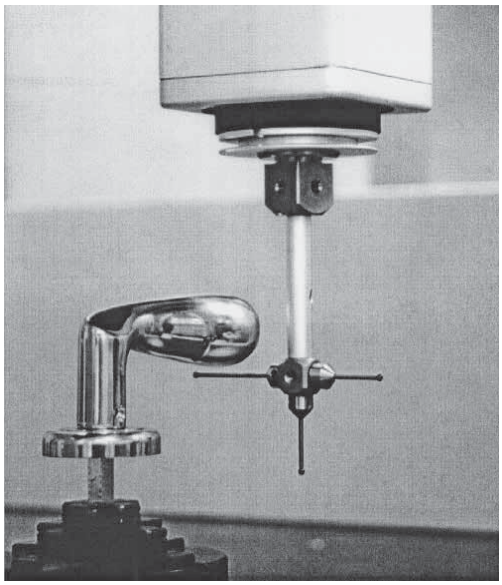


Figura 2.2: Medição de um puxador em uma Máquina de Medição de Coordenadas (CARBONE et al., 2001) .

Câmeras *Time Of Flight* (TOF) enviam pulsos de luz para o objeto alvo, a fim de medir a distância entre o emissor e a superfície. Por meio do cálculo do tempo entre a emissão e a recepção do feixe de luz, calcula-se o valor, dado que a velocidade da luz é conhecida e o tempo medido representa o tempo de ida e volta (*round trip time*) da luz(BÖHLER; MARBS, 2002). O *scanner* portátil *FARO Focus 3DA*, ilustrado na figura 2.3a, emprega *laser* para calcular os valores de profundidade. Já o modelo D-IMager da Panasonic, apresentado na figura 2.3b, utiliza LEDs de luz próxima ao infra-vermelho, podendo ser utilizado em ambientes com muita, ou nenhuma, iluminação.

(a) FARO Focus 3D (FARO, 2012) (b) Panasonic D-IMager (PANASONIC, 2012)

Figura 2.3: *Scanners Time of Flight*.

A técnica de superfícies de contornos é muito utilizada na medicina, onde camadas de "fotos" do objeto de interesse são capturadas por equipamentos especiais (Ressonância Magnética e Tomografia Computadorizada) para depois serem reconstruídas em estruturas tridimensionais. Os métodos mais comuns se aproveitam de características dos dados de entrada, como o fato de as camadas serem paralelas (SALEEM, 2004). A figura 2.4 apresenta um equipamento de ressonância magnética fabricado pela empresa *General Eletrics*.

Figura 2.4: Equipamento de Ressonância magnética (GE Healthcare, 2012).

Podem ser utilizados radares e sonares para a captura dos dados, observando-se o tempo que a onda leva para ser projetada no alvo e refletida. Esses equipamentos são mais utilizados em sensoreamento remoto de longo alcance, pois são muito precisos quando aplicados a objetos grandes, mas sua utilização em itens menores é inviável, devido ao tempo de medição ser da ordem de picos-segundos, sendo necessários circuitos muito rápidos para medir variações de tal ordem (SALEEM, 2004).

Métodos de luz estruturada consistem em produzir imagens 2D contendo os valores correspondentes à distância entre um ponto na cena e um outro conjunto de pontos. Um projetor é utilizado para apresentar padrões de luz em um objeto (grades, padrões de quadrados, etc) e uma câmera para captar as deformações nesses padrões. Dessa forma, através de triangulação, pode-se inferir a profundidade dos pontos na cena (NAVARRO, 2009). A figura 2.5 apresenta um esquema em que

um projetor P projeta padrões de linhas em um objeto B, cuja deformação é captada pela câmera C. Embora este método seja mais simples, possui a desvantagem de ser ineficaz quando aplicado à superfícies reflexivas ou transparentes, pois podem enviar a luz para direções opostas ao sensor (MADSEN et al., 2011).

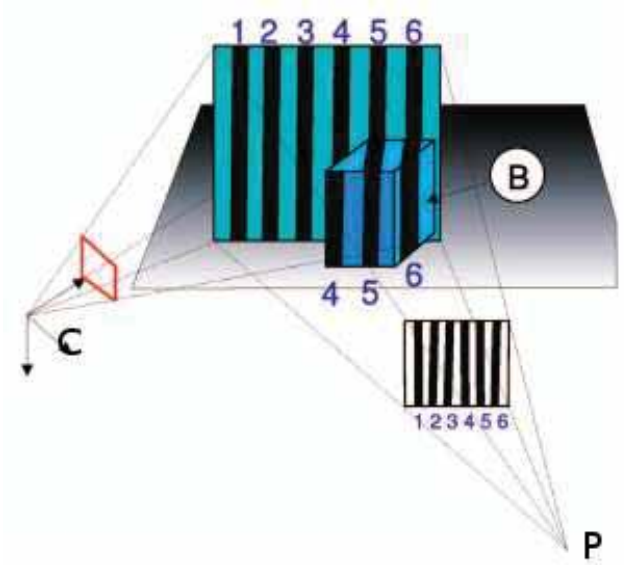


Figura 2.5: Esquema de captura com luz estruturada (NAVARRO, 2009).

2.3 Kinect

Kinect é um dispositivo desenvolvido pela *Microsoft* em conjunto com a *PrimeSense* para o *console* de jogos *Xbox 360* e tornou-se um importante sensor 3D de luz estruturada, devido ao seu baixo custo, confiabilidade e velocidade de medição. Seu anúncio em 2009 causou grandes expectativas nas comunidades acadêmicas de Computação Gráfica e Visão Computacional, já que o produto prometia um novo modo de interação, completamente baseado em gestos e voz. O equipamento possui um sensor de profundidade, uma câmera RGB , um acelerômetro, um motor, um vetor de microfones e um núcleo de processamento desenvolvido pela *PrimeSense* (CRUZ; LUCIO; VELHO, 2012).

Esse sensor pertence à classe de câmeras RGB-D (HENRY et al., 2010), pois possui uma câmera RGB padrão combinada com um emissor e um receptor infravermelho (IV), responsáveis pela imagem de profundidade (SOLONY; ZEMCIK, 2011; SMISEK; JANCOSEK; PAJDLA, 2011). Por utilizar luz IV, o *Kinect* não funciona bem com iluminação solar, sendo um dispositivo para interiores (CRUZ; LUCIO; VELHO, 2012). A Figura 2.6 ilustra esse equipamento fechado e aberto.

(a) Fechado (b) Aberto com detalhes dos sensores

Figura 2.6: Sensor Microsoft Kinect(SMISEK; JANCOSEK; PAJDLA, 2011).

A câmera IV opera a uma frequência de 30 Hz, capturando imagens de resolução 1200x960 *pixels*, que, por limitações de transferência da porta USB 2.0, são reduzidas para 640x480 *pixeis*. Mesmo assim, todos os dados provenientes do *Kinect* (RGB, profundidade, som e acelerômetro) consomem aproximadamente 70% de um *hub* USB e, por isso, não é possível conectar dois equipamentos no mesmo *hub* (CRUZ; LUCIO; VELHO, 2012). A figura 2.7 apresenta o padrão projetado pelo *Kinect* em uma superfície lisa.

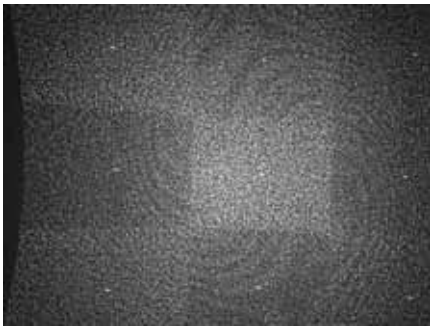


Figura 2.7: Padrão infravermelho projetado pelo *Kinect* (CRUZ; LUCIO; VELHO, 2012).

Como o mapa de profundidade é computado através da câmera IV, distante 7,5cm da câmera RGB, para se associar um valor de profundidade a um *pixel* de cor, deve-se calibrar as câmeras do equipamento dados os seus parâmetros intrínsecos e extrínsecos, como por exemplo posição e orientação das câmeras no mundo real, distâncias focais das lentes e distorções radiais e tangenciais (SOLONY; ZEMCIK, 2011; MADSEN et al., 2011). Algumas ferramentas podem ser utilizadas para realizar esses ajustes, por exemplo, a biblioteca *OpenNI*. O processo de calibragem é detalhado no trabalho de Smisek, Jancosek e Pajdla (2011) .

A captura dos valores de profundidade é realizada através de uma técnica de luz estruturada, que consiste em projetar um padrão de pixels em uma cena e capturar as deformações na projeção. O padrão utilizado no *Kinect* é patenteado pela *PrimeSense* e foi especialmente desenvolvido para minimizar ruídos provenientes do equipamento IV (CRUZ; LUCIO; VELHO, 2012).

O vetor de profundidade do *Kinect* reserva 11 bits por elemento (SOLONY; ZEMCIK, 2011), portanto tem-se uma faixa de 2^{11} = 2048 valores possíveis para distância. Pode-se obter valo-

res de 2,5 pés (800mm) a 13 pés (4000mm), embora a faixa recomendada é de 3 pés (914mm) a 12 pés (3657mm) (WEBB; ASHLEY, 2012). Valores superiores ou inferiores são considerados "desconhecidos". A Figura 2.8 ilustra uma cena capturada pela câmera de profundidade com sua correspondente RGB.

(a) Imagem RGB

(b) Imagem de profundidade. Espaços na cor preta representam profundidades desconhecidas, devido à oclusão ou material reflexivo atingido.

Figura 2.8: Cena capturada pelo sensor RGB-D(SOLONY; ZEMCIK, 2011).

2.4 Considerações finais

Neste capítulo, foram estudadas diferentes técnicas de escaneamento 3D, dando-se especial atenção ao sensor *Microsoft Kinect*. Apesar de ser um equipamento de baixo custo, o *Kinect* possibilita uma série de experimentos, tanto de interfaces homem-máquina quanto de reconstrução tridimensional. Embora possua limitações com relação à baixa resolução (em comparação com sensores profissionais) e a incapacidade de escanear objetos reflexivos ou transparentes, é capaz de capturar a maior parte das cenas de ambientes domésticos.

Capítulo 3

RECONSTRUÇÃO DE SUPERFÍCIES

Representações virtuais de objetos podem ser criadas em ferramentas CAD (*Computer Aided Design*) ou geradas a partir de modelos físicos existentes, através de técnicas de reconstrução de superfícies. O objetivo da reconstrução 3D é encontrar uma superfície a partir de um conjunto finito de valores geométricos amostrados (MULLER, 1997). Este capítulo discute as principais técnicas de reconstrução de superfícies.

3.1 Considerações iniciais

O processo de reconstrução tridimensional para aplicações com nuvem de pontos (conjunto de pontos no espaço) envolve um *pipeline* (conjunto de etapas) complexo, constituído pela captura do objeto, sincronização das câmeras RGB e de profundidade (alinhamento das imagens e calibragem da câmera), criação da nuvem de pontos (redução de ruídos, simplificação do conjunto de pontos e tratamento das regiões vazias) e transformação dos *pixels* 2D para *pixels* 3D (valor de cor e profundidade), gerando assim uma nuvem de pontos desordenada para então reconstruir a malha poligonal tridimensional (MADSEN et al., 2011). Pode-se exportar a superfície resultante para um formato predefinido, como por exemplo DFX (*Digital Effects*) , OBJ (*Wavefront Object File Format*) e VRML (*Virtual Reality Modelling Language*) , para posteriores ajustes e uso (MADSEN et al., 2011). O processo é apresentado na Figura 3.1.

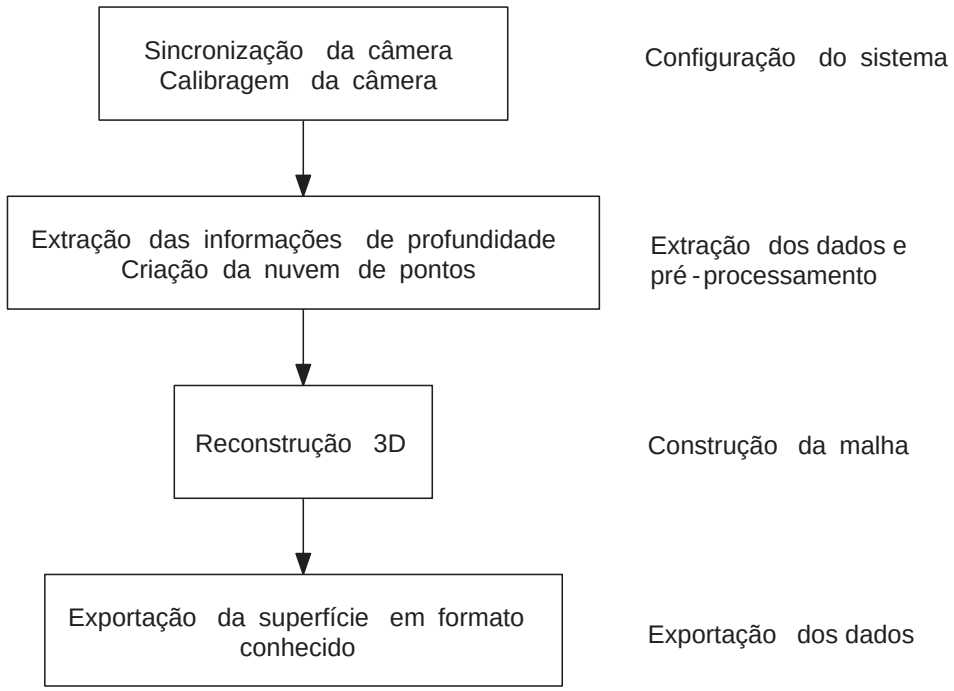


Figura 3.1: *Pipeline* adotado para reconstrução tridimensional (MADSEN et al., 2011).

A maior parte dos métodos de reconstrução tridimensional têm como objetivo gerar malhas poligonais (usualmente malhas triangulares) a partir de um conjunto de dados espaciais. Entretanto, se representações suavizadas forem necessárias, superfícies *spline* podem ser utilizadas. A geração de representações em LOD (*Level Of Detail*) também é possível, descrevendo-se superfícies em diferentes resoluções de acordo com os requisitos de visualização (MULLER, 1997). De acordo com Madsen et al. (2011), os principais algoritmos de reconstrução são: *Marching Cubes*, *Octrees*, Superfícies *Spline*, Algoritmo *Graham* 3D, Árvores Espaciais de Particionamento Binário e *KD-Trees*. Existem ainda algoritmos aproximadores de funções que podem ser aplicados ao processo de reconstrução, gerando resultados muito satisfatórios, por exemplo *Radial Basis Functions* (RBF) (CARR et al., 2001) e o método de Poisson (KAZHDAN; BOLITHO; HOPPE, 2006; CARR et al., 2001).

Este capítulo está dividido em seis subseções: primeiramente, será apresentada uma introdução aos métodos de reconstrução tridimensional, delimitando-se suas três principais classes de algoritmos. Em seguida, serão descritos dois métodos muito utilizados na reconstrução de superfícies, o algoritmo *Marching Cubes* e a estrutura de dados *Octree*. Os principais métodos de reconstrução a partir de nuvens de pontos serão discutidos. Na subseção final, o método de reconstrução de Poisson será apresentado.

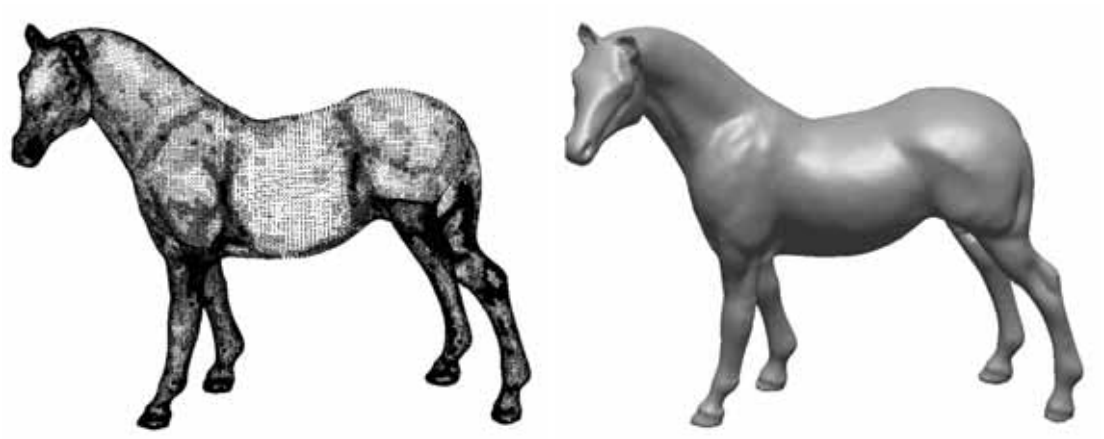
3.2 Introdução aos métodos de reconstrução

O conjunto de algoritmos de geração de malhas superficiais podem ser divididos em três classes: reconstrução a partir de seções planares, reconstrução a partir de nuvem de pontos e reconstrução de dados volumétricos (MULLER, 1997).

A primeira classe foi motivada pelos equipamentos de aquisição de imagens médicas, como ressonância magnética e tomografia computadorizada, que capturam fatias bidimensionais de objetos tridimensionais e, após processamento, representam-no no espaço. As etapas básicas para se atingir tal resultado são (GOIS, 2004):

- Detectar contornos no vetor de imagens;
- Gerar pontos sobre estes contornos;
- Conectar os contornos para definir um objeto tridimensional.

A segunda classe é mais recente, e engloba o grupo de algoritmos de reconstrução a partir de um conjunto de pontos no espaço, denominado nuvem de pontos. Isso significa que os dados não estão organizados em camadas, e, inicialmente, não existe relação entre eles. As amostras de pontos são obtidas de radares, sondas sísmicas e escâneres 3D (GOIS, 2004). De modo geral, esses pontos estão espalhados no espaço, sem nenhuma restrição, apenas com a premissa de que pertencem à mesma superfície (MULLER, 1997). A Figura 3.2 ilustra uma entrada para essa classe de algoritmos e sua respectiva saída (superfície reconstruída).



(a) Nuvem de pontos (b) Superfície reconstruída

Figura 3.2: Reconstrução a partir de uma nuvem de pontos. Nuvem fornecida por(KAZHDAN, 2012).

A classe de reconstrução de dados volumétricos trabalha com situações em que a superfície é dada implicitamente, representada através da forma $f(x,y,z) = c$, sendo f uma função de imagem

real e c uma constante real. A superfície é definida como o conjunto de pontos $p = (x, y, z)$ que satisfazem a equação. Se os pontos estiverem igualmente espaçados em uma grade, a abordagem *Marching Cubes* (LORENSEN; CLINE, 1987) é o método padrão para reconstruir a superfície. Caso contrário, pode-se interpolar esses pontos em uma superfície implícita e, em seguida, utilizar métodos de aproximação de superfícies implícitas para obter a malha final, por exemplo, o algoritmo *Marching Cubes*.

3.3 *Marching Cubes*

Marching Cubes (MC) é, atualmente, um dos algoritmos mais populares para extração de superfícies volumétricas e renderização (NEWMAN; YI, 2006). Publicado em 1987 por Lorensen e Cline, foi criado com o objetivo de criar modelos triangulares de densidade constante a partir de dados médicos (LORENSEN; CLINE, 1987). Partindo de informações volumétricas escalares, o MC processa o conjunto de dados considerando as fatias que compõe este volume (NEWMAN; YI, 2006). A Figura 3.3 ilustra um cubo formado pelas superfícies S: k e k+1.

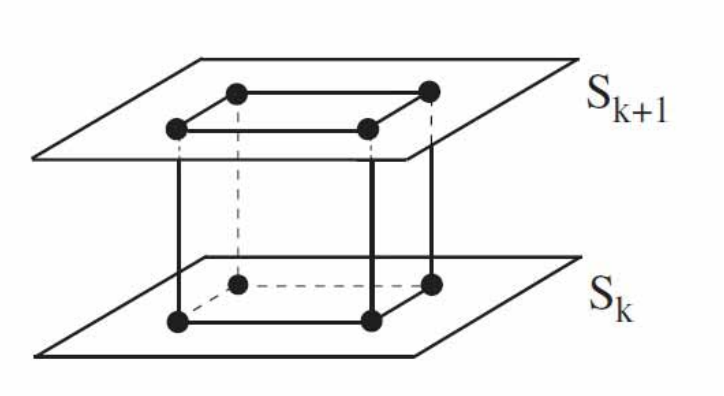


Figura 3.3: Cubo formado pelas fatias k e k+1 (NEWMAN; YI, 2006).

O algoritmo determina como a superfície intercepta seu cubo (ou vizinhança de pontos 2x2x2 (MADSEN et al., 2011)), procura na tabela de vértices esta configuração e adiciona os triângulos resultantes à superfície, e então vai pra o próximo cubo (LORENSEN; CLINE, 1987). Como existem oito vértices em cada cubo e dois estados, dentro e fora da superfície, existem 2^8 = 256 possibilidades de superfície interceptar o cubo. Enumerando essas 256 possibilidades e eliminando redundâncias e simetrias, chega-se a 15 padrões. A Figura 3.4 apresenta a triangulação desses 15 casos e a intersecção dos vértices para cada configuração de cubo.

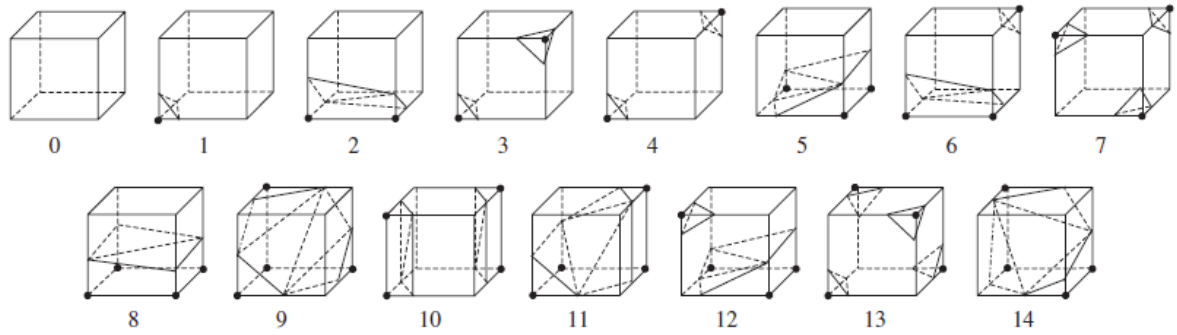


Figura 3.4: Triangulação das 15 possibilidades do algoritmo Marching Cubes (LORENSEN; CLINE, 1987).

3.4 *Octrees*

Uma *Octree* é uma árvore cujos nós não folha possuem interligação com mais oito nós na estrutura de dados. *Octrees* possuem diversas aplicações, entre elas, a de algoritmo de modelagem volumétrica. Ele funciona usando a cena completa (do mundo real) como o nó raiz, representado por um cubo (*Voxel*) . Os próximos nós são recursivamente divididos em 8 filhos, dividindo o tamanho de seus pais na metade em cada eixo tridimensional (MADSEN et al., 2011; SZELISKI, 1993), como mostra a Figura 3.5. O processo é repetido até que uma das seguintes condições seja alcançada (MADSEN et al., 2011):

- Cada nó esteja completamente cheio ou completamente vazio;
- Um coeficiente de erro seja atingido;
- O número máximo recursões seja alcançado.

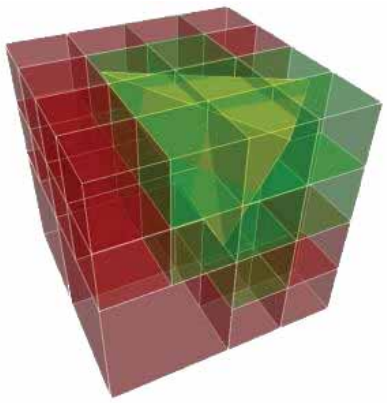


Figura 3.5: Representação de um objeto através de Octrees (MADSEN et al., 2011).

A Figura 3.6 apresenta a configuração dos *voxels* no espaço e na estrutura de árvore em diferentes níveis. Cada cubo é representado por um ponto central, que pode possuir um valor booleano

ou uma informação de intensidade.

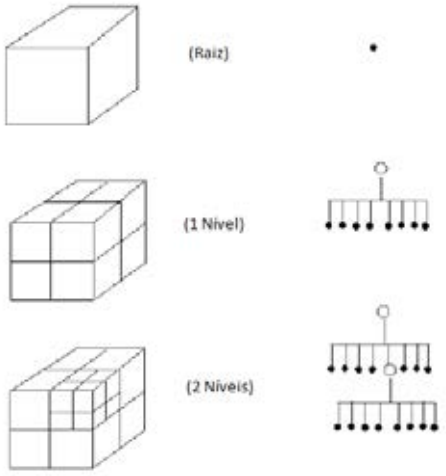


Figura 3.6: Estrutura da Octree em diferentes níveis (MADSEN et al., 2011).

3.5 Reconstrução de superfícies a partir de nuvens de pontos

De acordo com Mencl e Muller (1999), os algoritmos de reconstrução a partir de nuvens de pontos se dividem em: funções de distância, subdivisão espacial, deformações de malha e superfícies incrementais. Essas classes serão discutidas a seguir:

Função de distância

Funções de distância calculam a distância D entre qualquer ponto e a superfície. Para $D = 0$, temos uma superfície implícita que aproxima a superfície em questão (MENCL; MULLER, 1999). Hoppe et al. (1994) apresentaram uma solução para o cálculo dessa distância: primeiramente, define-se um plano orientado para cada ponto da superfície amostrada, baseando-se na redondeza deste ponto. Esses planos servem como uma interpolação linear da superfície em questão e por isso são utilizados para o cálculo da função de distância. Roth e Wibowoo (1997), por sua vez, desenvolveram uma função de distância cujo objetivo principal é calcular as distâncias entre os vértices em uma determinada grade de *voxels* e os pontos ao seu redor.

Subdivisão espacial

Métodos de subdivisão espacial decompõe um conjunto de pontos P em células (caixas delimitadoras), selecionando apenas as que estão relacionadas com a forma descrita por P. Exemplos típicos

destes métodos são *Octrees* (esquemas adaptativos) e malhas tetraédricas (esquemas irregulares) (MENCL; MULLER, 1999).

O algoritmo de Algorri e Schmitt (1996) divide o espaço em uma grade de *voxels* igualmente espaçados. Em seguida, o algoritmo extrai os *voxels* que possuem pelo menos um ponto de P. Na próxima etapa, os quadriláteros que compõem cada um dos voxels escolhidos são considerados a primeira aproximação da superfície. Para se conseguir um resultado visualmente mais agradável, pode-se decompor cada quadrilátero em dois triângulos e aplicar um filtro passa-baixa para suavizar a malha resultante.

Outra possibilidade para a reconstrução por subdivisão espacial é a abordagem de Hoppe et al. (1994) que, através de uma função de distância, constrói uma superfície implícita e a transforma em uma malha triangular através do algoritmo MC.

Métodos de deformações

Algoritmos que utilizam deformações alteram uma superfície inicial de modo que ela se aproxime de um dado conjunto de pontos P (MENCL; MULLER, 1999). Diferentes abordagens foram utilizadas para realizar esta tarefa, por exemplo o método de deformação do espaço, proposto por Ruprecht, Nagel e Muller (1995) e a abordagem com Mapas de Kohonen, proposta por Baader e Hirzinger (1993).

Saleem (2004) apresenta um estudo sobre o algoritmo de Malhas Neurais, que utiliza ferramentas do aprendizado de máquina para deformar uma superfície inicial e gerar uma malha que aproxima a nuvem de pontos. A Figura 3.7 apresenta a execução desse algoritmo.

Figura 3.7: Aprendizado do algoritmo Malhas Neurais. Da esquerda para a direita, a malha base e reconstruções de um coelho com 100, 250 e 500 vértices (SALEEM, 2004).

Superfícies incrementais

A ideia de algoritmos orientados a superfícies incrementais é construir uma superfície que aproxime o conjunto de pontos de entrada diretamente, baseando-se nas suas características (MENCL; MULLER, 1999).

A abordagem de Boissonnat (1984) inicia a criação da superfície gerando uma aresta entre os dois pontos mais próximos do conjunto de dados e, em seguida, adiciona novos triângulos para essa e para as próximas arestas que se formarem. No final da execução do algoritmo, não haverão mais arestas livres e a malha triangular estará formada.

Já Mencl (1995) inicia o processo de reconstrução através da extração de características da nuvem de pontos, e, conectando características semelhantes, é gerada uma aproximação inicial em *wireframe* do objeto. Em seguida, preenchem iterativamente o objeto *wireframe* com triângulos, aumentando o nível de detalhamento da malha final.

3.6 O método de reconstrução de Poisson

O algoritmo de *Poisson* (KAZHDAN; BOLITHO; HOPPE, 2006) é um método de reconstrução de superfícies que gera malhas poligonais fechadas, aproximando os dados amostrados através de uma equação de *Poisson*.

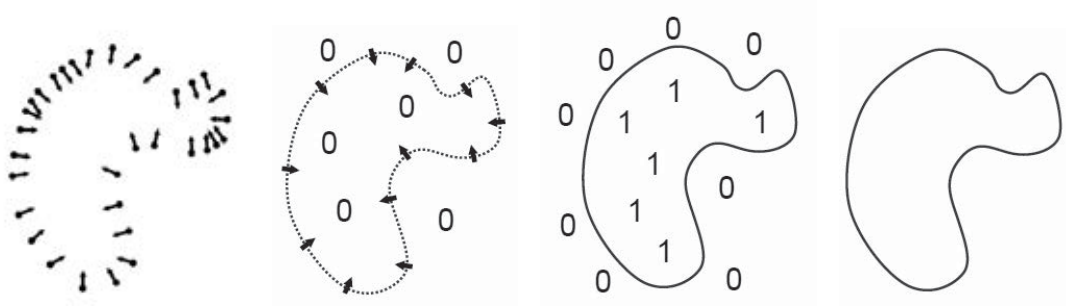
Dada uma nuvem de pontos orientada (pontos combinados com seus vetores normais), o método calcula uma função indicadora $\mathcal{X}_{\mathrm{M}}$ que aponta se um elemento faz parte de um subconjunto M. No caso, $\mathcal{X}_{\mathrm{M}}$ é definida como 1 em pontos dentro do modelo 3D e 0 em pontos fora dele.

$$\mathcal{X}_{\mathrm{M}}(x) = \begin{cases} 1 & \text{se} x \in M \\ 0 & \text{se} x \notin M \end{cases}$$

Aproveitando-se da relação intrínseca entre a nuvem de pontos e sua função indicadora, o método considera que os pontos orientados $\vec{V}$ são amostras do gradiente função indicadora $\Delta\mathcal{X}_M$. Dessa forma, identificar a função indicadora significa inverter o operador gradiente, ou seja, encontrar a função escalar $\mathcal{X}$ cujo gradiente aproxima o campo vetorial $\vec{V}$, resolvendo o problema de otimização $min_{\mathcal{X}} \left\| \mathcal{X}_{\mathrm{M}} - \vec{\mathrm{V}} \right\|$. Aplicando-se o operador de divergência, a equação se transforma em um problema de *Poisson* tradicional: calcular a função escalar $\mathcal{X}$ cujo Laplaciano $\triangle\mathcal{X}$ é igual ao do campo vetorial $\vec{V}$, ou seja,

$\triangle\mathcal{X} \equiv \nabla \cdot \nabla\mathcal{X} = \nabla \cdot \vec{V}$

Efetuado o cálculo de $\mathcal{X}_{\mathrm{M}}$, pode-se calcular os contornos da função indicadora através de um algoritmo de *zero-crossing* e extrair a superfície com o algoritmo *Marching Cubes*. A figura 3.8 ilustra o algoritmo aplicado a um conjunto de dados no $\mathbb{R}^2$.



(a) Pontos orientados $\vec{V}$ (b) Gradiente da Função Indicadora $\Delta\mathcal{X}_M$ (c) Função Indicadora $\mathcal{X}_M$ (d) Superfície δM

Figura 3.8: Ilustração do método de reconstrução de Poisson em 2D (KAZHDAN; BOLITHO; HOPPE, 2006).

Soluções exatas são necessárias apenas nas proximidades da superfície, portanto uma estrutura *octree* é aplicada para aproximar os dados originais e, escolhendo-se uma profundidade máxima para a árvore, pode-se criar superficies com vários LOD (aplica-se o algoritmo MC nas soluções aproximadas). A reconstrução ocorre em uma grade regular no $\mathbb{R}^3$ de resolução $i \text{x } j \text{x } k$, sendo que $i = j = k = 2^{nivel}$. A Figura 3.9 ilustra uma malha reconstruída com o método de poisson utilizando três níveis de detalhe: de cima para baixo, *octree* de profundidade 6, 8 e 10 (grade de resolução 64^3, 256^3 e 1024^3). Quanto mais profunda a árvore, mais finos são os detalhes que a reconstrução consegue representar. Nesse exemplo, as escamas do dragão tornam-se visíveis apenas com resoluções mais altas.

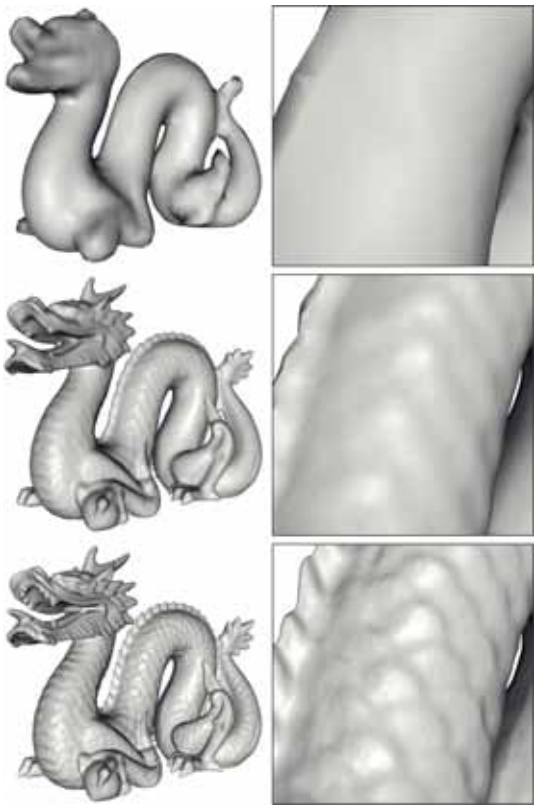
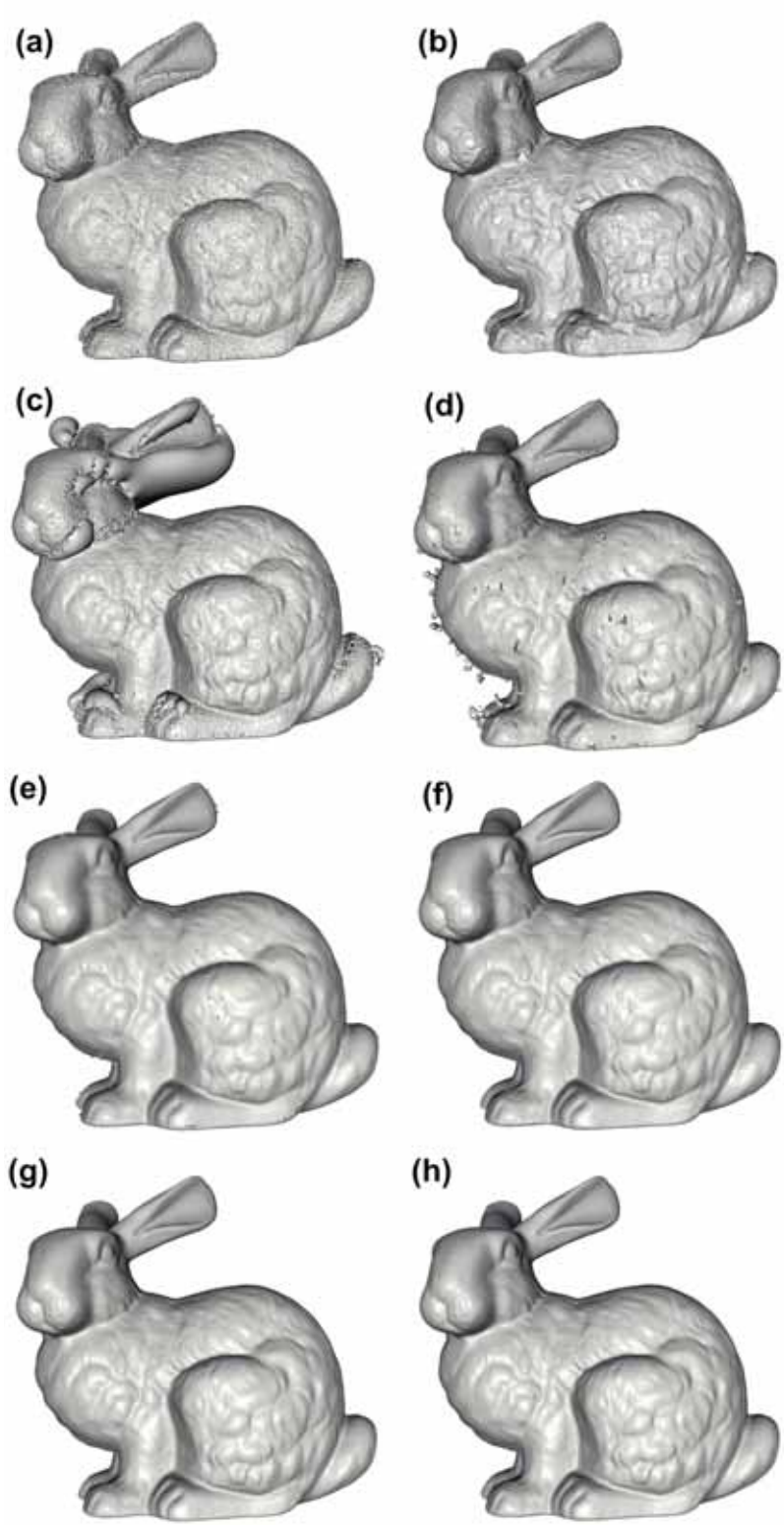


Figura 3.9: Dragão de *Stanford* reconstruído com o método de Poisson em 3 níveis de detalhe (KAZHDAN; BOLITHO; HOPPE, 2006).

A formulação da reconstrução se superfícies como um problema de Poisson oferece diversas vantagens. Enquanto muitos métodos de aproximação de superfícies segmentam os dados em regiões, ajustando-as localmente para depois combiná-las, o método de Poisson considera todo o conjunto de dados de uma vez. Dessa forma, cria superfícies bem suavizadas que aproximam robustamente dados ruidosos. A figura 3.10 apresenta a reconstrução do Coelho de *Stanford* com diversos métodos de reconstrução tridimensional. Observa-se que a reconstrução com o método de Poisson realmente obtem a solução mais suavizada entre os algoritmos comparados no trabalho de Kazhdan, Bolitho e Hoppe (2006).



Power Crust (a), *Robust Cocone* (b), *Fast RBF* (c), *MPU* (d), Hoppe et al. (1994) (e), *VRIP* (f), Reconstrução baseada em *FFT*(g) e Reconstrução de *Poisson* (h)

Figura 3.10: Reconstruções do Coelho de *Stanford* com diferentes métodos (KAZHDAN; BOLITHO; HOPPE, 2006).

3.7 Considerações Finais

Considerando o contexto exposto, observa-se que o método de reconstrução de *Poisson* destaca-se dos demais, pois apresenta uma boa aproximação da superfície de forma suavizada e é capaz de representar detalhes finos, desde que a amostra seja suficientemente densa.

Capítulo 4

SIMPLIFICAÇÃO DE SUPERFÍCIES

Recentemente, uma grande quantidade de dados volumétricos têm sido captados por equipamentos de medição de alta resolução ou de simulações numéricas de larga escala. A visualização interativa de tais dados é muito difícil e diversos algoritmos são encontrados para recriá-los de forma rápida, mantendo o seu significado original (GERSTNER, 2002). Algoritmos de multirresolução são ferramentas muito populares na visualização de grandes conjuntos de dados, pois constroem versões simplificadas, mas ainda assim estruturalmente semelhantes aos dados originais (GERSTNER; PAJAROLA, 2000).

4.1 Considerações Iniciais

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. Tais algoritmos têm sido aplicados com sucesso em diversas vertentes da computação gráfica, entre elas, mapeamento de textura e reconstrução de superfícies. Na reconstrução tridimensional, essas técnicas decompõe o espaço em aproximações dos dados, permitindo, além de um processamento mais rápido, a extração de superfícies com LOD variado (GERSTNER, 2002). Com um LOD menor, o número de polígonos também é reduzido, possibilitando que a malha seja renderizada mais rapidamente (BOTSCH; KOBBELT, 2001). Diversas abordagens podem ser utilizadas na representação em multirresolução, entre elas abordagens hierárquicas, piramidais e com Octrees (NEWMAN; YI, 2006). A Figura 4.1 exibe um exemplo de representação em múltiplas resoluções.

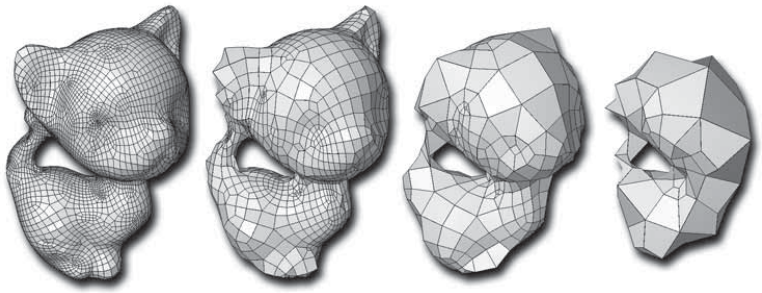


Figura 4.1: Representação de um gato em múltiplas resoluções(DANIELS et al., 2008).

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

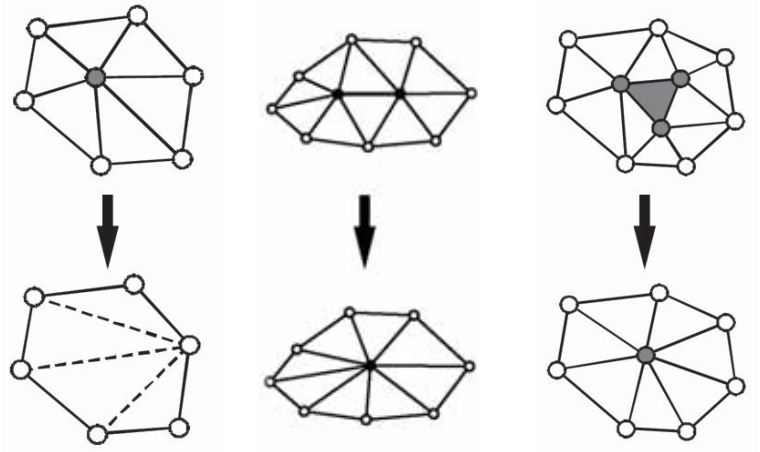
4.2 Métodos incrementais

Técnicas incrementais de simplificação de superfícies procedem como 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 4.1 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 4.2.



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

Figura 4.2: Operações de atualização local (FLORIANI et al., 1999).

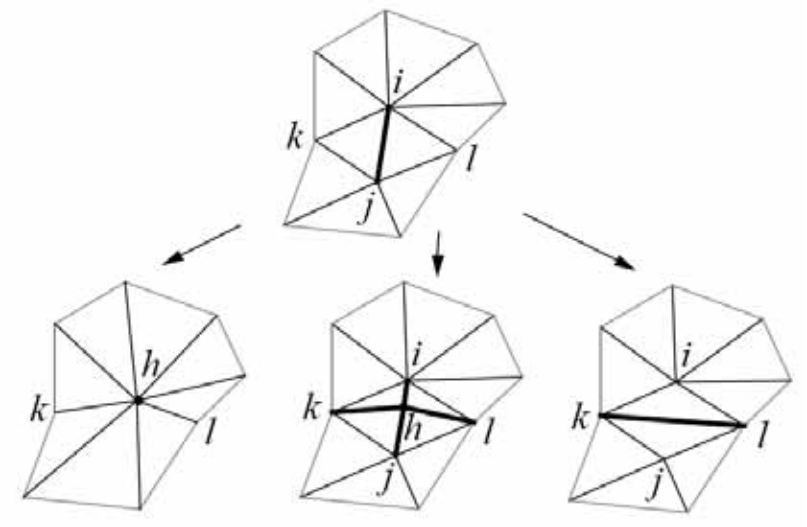
Hoppe et al. (1993) desenvolveram um método de simplificação que consiste na execução iterativa de contração de aresta (*collapse*), partição de aresta (*split*) e troca de aresta (*swap*), de modo a otimizar a função de energia, definida como $E(M) = Edist(M) + Erep(M) + Espring(M)$, onde

Edist: soma das distâncias dos pontos originais a M;

Erep: fator proporcional ao número de vértices em M;

Espring: soma dos comprimentos das arestas.

A qualidade da aproximação (simplificação) é avaliada através dessa função de energia. A figura 4.3 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 4.3: 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 LODs diferentes), exemplificada na figura 4.4 (HOPPE, 1996).

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

Figura 4.4: Exemplo de *Geomorph*, de 500 faces (a) a 1.000 faces (e), de acordo com um parâmetro de transição α (HOPPE, 1996).

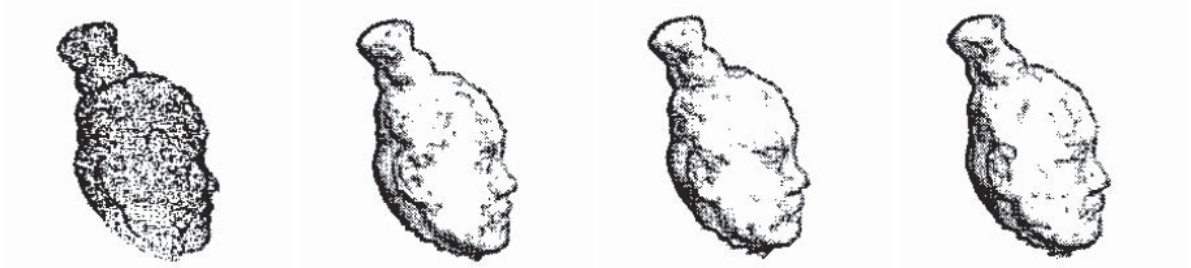
Nesse método, otimiza-se uma função de energia mais complexa, definida como
$E(M) = Edist(M) + Espring(M) + Escalar(M) + Edisc(M)$,
sendo que os dois primeiros valores são os mesmos de (HOPPE et al., 1993) e
Escalar(M): responsável por preservar os cantos da malha;
Edisc(M): responsável pela simplificação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

4.3 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. Estabelecendo-se um limiar para as variações nos vetores normais de cada superfície, pode-se verificar se ela é lisa ou áspera (verifica-se coplanaridade). Este resultado possibilita simplificações na representação da malha, sendo que para um limiar maior, obtêm-se uma malha mais simplificada. A figura 4.5 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 4.5: 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 de custo, e gera uma malha mais simplificada através de uma técnica de *remeshing* (calculo da 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 4.6.

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

Figura 4.6: Execução do método de Lee et al. (1998).

Lee, Juho e Yang (2002) desenvolveram um modelo para representações LOD que mistura características do MC com *Octree*. Esta solução suporta operações de simplificações adaptativas, compressão de dados, transmissão progressiva, renderização dependente do ponto de vista e detecção de colisões, muito utilizadas em softwares de computação gráfica. Para gerar representações em várias resoluções, o algoritmo calcula uma nova posição para cada vértice dos triângulos gerados pelo MC, utilizando a função de distância de Hoppe et al. (1994). A figura 4.7 mostra um exemplo de reconstrução utilizando esse método.

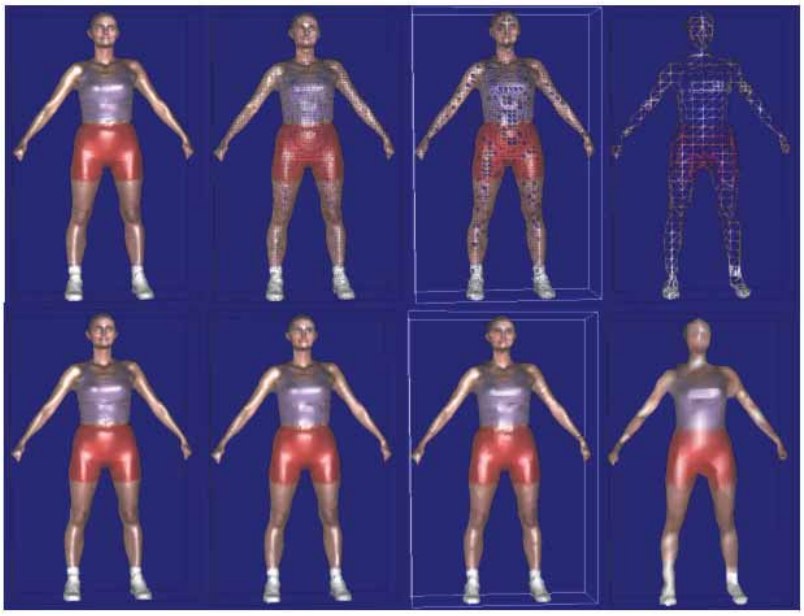


Figura 4.7: Reconstrução de um modelo feminino com o método de Lee, Juho e Yang (2002).

Métodos de *clustering* (agrupamento) detectam e unem grupos de vértices próximos, sendo que todas as faces com 2 ou 3 vértices num grupo são removidas (FLORIANI et al., 1999). A figura 4.8 ilustra um exemplo para malhas em 2D. Pode-se observar a aplicação desse método na figura

4.8, que aplica uma versão chamada *Superfaces*. Esse método tem a desvantagem de produzir resultados de baixa qualidade de aproximação (FLORIANI et al., 1999).

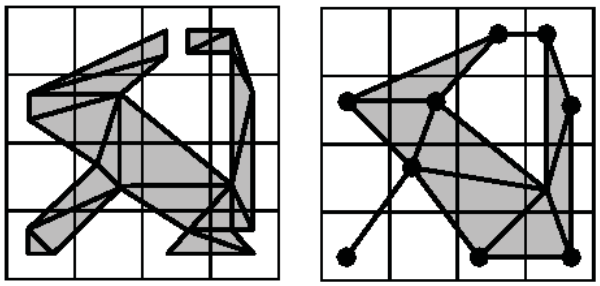


Figura 4.8: Simplificação através de *clustering* em 2D *(FLORIANI et al., 1999).*

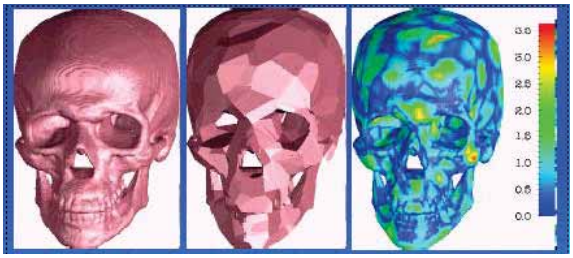


Figura 4.9: Aplicação do método *Superfaces* a um modelo do crânio humano (FLORIANI et al., 1999).

Abordagens com análise multirresolução e subdivisão adaptativa utilizam *wavelets* para representar uma função em múltiplos níveis de detalhe, parametrizando-a suavemente em qualquer LOD. Pode-se observar um modelo reconstruído com diferentes coeficientes de erro (simplificação) através de *wavelets* na figura 4.10.

Figura 4.10: Modelo reconstruído utilizando-se *wavelets* com diferentes coeficientes de erro. Da esquerda para a direita, $\varepsilon = 9,0$, $\varepsilon = 2,75$ e $\varepsilon = 0,1$(FLORIANI et al., 1999).

Essa técnica se baseia na ideia de decompor uma função complexa em uma parte mais simples (baixa resolução), e uma coleção de perturbações, chamadas de coeficientes *wavelet*, necessárias para recompor a função original(LOUNSBERY; DEROSE; WARREN, 1997). Lounsbery, Derose e

Warren (1997) apresentam um algoritmo capaz de aplicar análise multirresolução à superfícies de topologias arbitrárias.

Gross, Gatti e Staadt (1995) utilizam aproximações por meio de *wavelets* para analisar a superfície a ser reconstruída. Verifica-se, através da análise multirresolução, quais são os locais com poucos detalhes na superfície original. Para esses espaços, pode-se utilizar triângulos maiores para aproximar a superfície, sem perder detalhes muito finos. Nesse trabalho, é apresentada uma nova classe de filtros espaciais *wavelets*, que atuam como lentes de aumento, possibilitando a representação de superfícies com LOD variado.

4.4 Considerações finais

Diversos métodos de simplificação de superficies foram apresentados neste capítulo. Observou-se que o método proposto por Lindstrom e Turk (1998) destaca-se dos demais, pois fornece um grande controle sobre a qualidade da aproximação da superfície e apresenta um esforço computacional inferior em relação aos trabalhos correlatos de simplificação incremental.

Capítulo 5

MÉTODO DE POISSON COM SIMPLIFICAÇÃO LOCAL DE MALHAS

O objetivo central deste projeto é a geração de malhas em multirresolução a partir de uma nuvem de pontos. Para isso, foi desenvolvido um *pipeline* que combina a técnica de reconstrução de superfícies de Poisson (KAZHDAN; BOLITHO; HOPPE, 2006) e o método de simplificação de malhas poligonais por contração de arestas (LINDSTROM; TURK, 1998).

5.1 Considerações Iniciais

Por meio de experimentos preliminares com o método de reconstrução de superfícies de Poisson, observou-se que a malha resultante do processo de reconstrução é muito próxima à superfície escaneada, utilizando-se níveis de *octree* maiores que 7 (resoluções mais altas). Entretanto, quando deseja-se reconstruir superfícies com diferentes LODs e aplica-se níveis de *octree* mais baixos, percebe-se que muitos detalhes que se deseja preservar são perdidos. O método também não permite um controle fino sobre o nível de simplificação (porcentagem de simplificação, número de polígonos, etc), o que dificulta a geração de malhas para aplicações com *geomorphing* (transição suave entre malhas de diferentes resoluções). O gráfico representando a quantidade de polígonos por profundidade de *octree,* calculado com base na nuvem de pontos do Coelho de *Stanford* (STANFORD, 2011), é apresentado na figura 5.1. Destaca-se que a quantidade de polígonos cresce exponencialmente, até atingir uma assíntota horizontal, em que a reconstrução do modelo não pode ser mais detalhada.

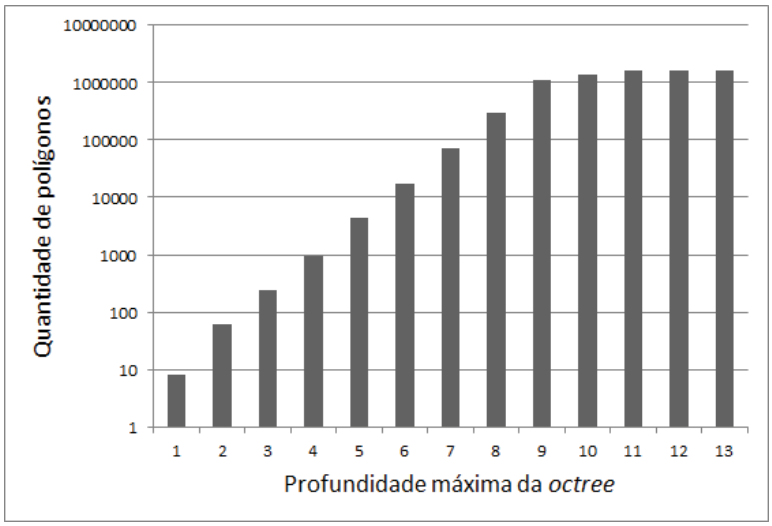


Figura 5.1: Gráfico da quantidade de polígonos da reconstrução do Coelho de *Stanford* por profundidade de *octree*.

A técnica de simplificação por contração de arestas de Lindstrom e Turk (1998) foi selecionada para diminuir o número de polígonos gerados pela reconstrução. Ele possibilita um grande controle sobre a qualidade da aproximação da superfície e requer um menor número de operações para realizar a simplificação, em comparação com os demais métodos iterativos estudados.

Foi desenvolvido um *pipeline* completo de reconstrução 3D, representado pela figura 5.2. Nesse capítulo, cada uma dessas etapas serão detalhadas.

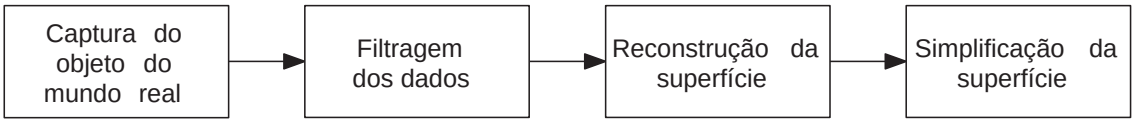


Figura 5.2: *Pipeline* desenvolvido.

5.2 Captura da nuvem de pontos

O sensor Microsoft Kinect foi utilizado para capturar as nuvens de pontos utilizadas neste trabalho. Como interface Kinect / Computador, empregou-se a biblioteca PCL em conjunto com os *drivers* e pacotes da biblioteca OpenNI. Duas abordagens de captura foram empregadas: captura de apenas um único frame RGB-D e combinação de vários frames, utilizando o algoritmo ICP (*Iterative Closest Points*) (BESL; McKay, 1992) para registrar as nuvens de pontos. As duas abordagens serão

descritas e comparadas a seguir.

5.2.1 Captura de um único frame RGB-D

Esta foi a primeira abordagem utilizada neste trabalho. O sensor é posicionado em frente ao objeto a ser escaneado e captura-se apenas um frame RGB-D da cena. Este é o método mais simples de captura e possui as seguintes desvantagens:

- Superfícies côncavas não são representadas corretamente;
- Como apenas um ponto de vista é capturado, a superfície não pode ser completamente reconstruída, e consequentemente visualizada em 360°;
- O cálculo dos vetores normais à superfície (requisito para o método de reconstrução de Poisson) pode gerar valores incorretos, pois apenas uma parte da superfície é considerada.

A figura 5.3 ilustra o problema das oclusões (principalmente nas regiões destacadas com elipses vermelhas). A região referente à pia não foi destacada pois superfícies reflexivas não são bem capturadas pelo Kinect.

(a) Imagem RGB (b) Profundidade representada em tons de cinza

Figura 5.3: Oclusões (Washington, 2012).

O processo de captura desenvolvido é descrito no algoritmo 5.1.

Algoritmo 5.1 Captura do *frame* RGB-D.

Inicializar a interface de captura do Kinect;
Defina i = 0;
Faça
    Apresentar o *frame* atual na tela;
    Se usuário pressionar a tecla <espaço>:
        Salvar o *frame* em arquivo com o nome "Cloud" + *str*(i);
        i = i + 1;
Até que: usuário finalize a aplicação;

5.2.2 Combinação de vários frames

A combinação de vários frames RGB-D faz parte do problema de alinhamento geométrico de conjuntos de pontos no espaço. Dadas duas nuvens de pontos, P e Q, o método de combinação deve encontrar um movimento euclidiano (matriz de transformação) que leva P ao mais próximo alinhamento com Q (CHETVERIKOV et al., 2002). O algoritmo ICP, proposto por Besl e McKay (1992) é uma solução bem conhecida para o problema de alinhamento e funciona em três etapas, executadas iterativamente, conforme o algoritmo 5.2.

Algoritmo 5.2 *Iterative Closest Points* (BESL; McKay, 1992).

Faça
    Associar cada ponto em P com um correspondente em Q;
    Calcular o movimento que minimiza o Erro Quadrático Médio entre as nuvens;
    Aplicar o movimento a P e atualizar o Erro Quadrático Médio;
Até que: Erro Quadrático Médio < Erro estipulado;

Esse algoritmo é preferencialmente utilizado em nuvens de pontos sobrepostas (previamente alinhadas) e tem garantia de convergência em termos do Erro Quadrático Médio (BESL; McKay, 1992).

No presente trabalho, a captura e combinação de vários frames RGB-D é realizada através de uma versão paralelizada em GPU (*Graphics Processing Unit*) do algoritmo ICP, disponível na biblioteca PCL - módulo KinFu. Essa implementação é descrita no projeto Kinect Fusion (IZADI et al., 2011), que apresenta um mecanismo para reconstrução tridimensional em tempo real através da movimentação do Microsoft Kinect ao redor de uma cena.

Essa abordagem traz diversas vantagens, descritas a seguir:

- O processo de captura e o registro podem ser executados em tempo real (resultados variam de acordo com a capacidade de processamento da placa de vídeo);
- Vários pontos de vista são capturados e processados, diminuindo a quantidade de oclusões e regiões sem pontos na nuvem;
- O algoritmo descrito por Izadi et al. (2011) realiza uma filtragem espacial nos dados amostrados, diminuindo a quantidade de ruído de acordo com a quantidade de frames capturados.

O processo de captura e registro das nuvens de pontos em tempo real é descrito no algoritmo 5.3.

Algoritmo 5.3 Captura e registro das nuvens de pontos em tempo real.

Inicializar a interface de captura do Kinect;
Defina nuvem = ∅;
Faça
    Defina nuvemAtual = *Frame* atual;
    matrizTransformacao = ICP(nuvem,nuvemAtual); /*encontra a matriz de transformação (rotação e translação)*/
    AplicaTransformacao (matrizTransformacao, nuvemAtual);
    nuvem = Combinar (nuvem, nuvemAtual);
Até que: usuário pressione a tecla <espaço>;
Salvar a nuvem em arquivo;

5.3 Filtragem dos dados

A filtragem dos dados ocorre em três etapas:

1. Filtragem de ruídos;
2. Remoção de valores inválidos;
3. Segmentação do objeto a ser reconstruído.

5.3.1 Filtragem de ruídos

A filtragem de ruídos é realizada enquanto o objeto é escaneado, por meio do método descrito por (IZADI et al., 2011). A cada iteração do algoritmo 5.3, durante a combinação das nuvens de pontos, ocorre um processo de subamostragem, em que pontos são removidos, de modo a eliminar ruídos. Nota-se que uma nuvem de pontos capturada dessa maneira possui muito menos ruído do que uma única nuvem "crua" produzida pelo *Microsoft Kinect*. A figura 5.4 ilustra essa conclusão.

(a) Imagem RGB

(b) Reconstrução da superfície proveniente de um *frame* RGB-D.

(c) Reconstrução da superfície proveniente de uma sequência de *frames* RGB-D

Figura 5.4: Comparação entre dados de um frame do *Kinect* e dados do *Kinect Fusion*. Adaptado de (IZADI et al., 2011).

5.3.2 Remoção de valores inválidos

Nessa etapa, os valores que o sensor não conseguiu definir (regiões na cor preta da figura 5.3) são removidos da nuvem de pontos. Valores desconhecidos possuem uma constante dentro da biblioteca PCL, cujo mnemônico é NaN (*Not a Number*). O algoritmo 5.4 ilustra essa operação.

Algoritmo 5.4 Remoção de valores inválidos

```
Leia a nuvem de pontos fornecida pelo usuário;
Para cada ponto p pertencente à nuvem:
    Se p == NaN:
        Remova p da nuvem
Salve a nuvem de pontos em arquivo;
```

5.3.3 Segmentação do objeto a ser reconstruído

A segmentação de objetos 3D pode ser definida como a tarefa de identificar limites para um objeto no espaço, identificando e separando os componentes da cena (LI et al., 2006). A figura 5.5 ilustra um exemplo de segmentação com a biblioteca PCL, em que cada objeto segmentado recebe uma cor diferente.

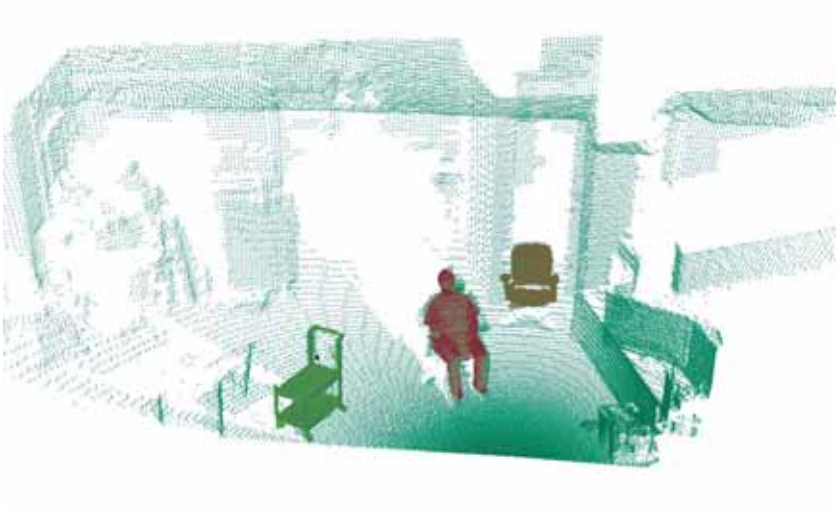


Figura 5.5: Exemplo de segmentação com a biblioteca PCL (RUSU; COUSINS, 2011).

Para este projeto, deseja-se que a malha reconstruída contenha apenas o objeto desejado, e não a cena completa. Duas soluções de segmentação foram utilizadas: uma semi-automática e uma manual.

5.3.3.1 Segmentação semi-automática

Foi desenvolvida uma solução de segmentação semi-automática, que segmenta e remove a superfície plana em que o objeto está situado. Para isso, foi utilizado o aproximador de funções RANSAC (*Random Sample Consensus*) (FISCHLER; BOLLES, 1981), que encaixa modelos aos dados experimentais. No caso, define-se um plano como o modelo a ser "encaixado" e RANSAC o transformará para que coincida com o plano na nuvem de pontos.

Como parâmetro, o usuário deve fornecer um limiar (*threshold*) de aceitação do plano presente na nuvem. A figura 5.6 ilustra a execução deste algoritmo com dois *thresholds*.

(a) Cena original (b) Cena segmentada com *threshold* 0,1 (c) Cena segmentada com *threshold* 0,05

Figura 5.6: Segmentação com remoção de planos.

Os passos para a remoção do plano são apresentados no algoritmo 5.5.

Algoritmo 5.5 Segmentação semi-automática do plano.

Leia a nuvem de pontos;
Leia o limiar para segmentação;
Encontre o plano dominante na cena através do RANSAC;
Remova o plano;
Salve a nuvem de pontos em arquivo;

Embora esse método tenha a vantagem de não necessitar de muita intervenção do usuário, são raras as cenas onde existe apenas um objeto sobre um plano. Quando isso não ocorre, esse modelo de segmentação falha, como mostra a figura 5.7. Observa-se que o objeto atrás da estátua ainda permanece na cena depois da segmentação.

(a) Nuvem de pontos original.

(b) Cena segmentada com *threshold* 0,1

(c) Cena segmentada com *threshold* 0,05

Figura 5.7: Exemplo de erro durante a segmentação.

5.3.3.2 Segmentação manual

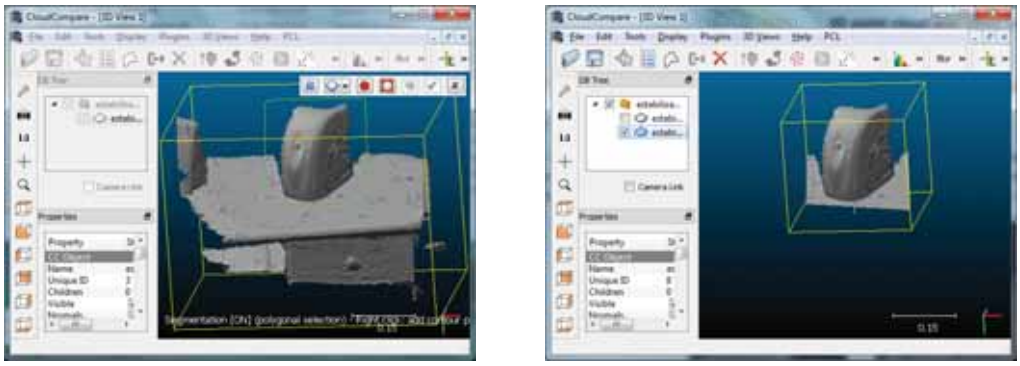
A maior parte das cenas capturadas possuem diversos objetos, o que dificulta o processo de segmentação. No exemplo da figura 5.5, observa-se 4 grupos de objetos segmentados. Nesse contexto, dois casos podem ocorrer:

- Os grupos de pontos foram segmentados corretamente e pode-se extraír o objeto desejado da cena;
- O objeto não foi identificado corretamente pelo algoritmo e deve ser segmentado manualmente.

Em ambos os casos, há uma maior necessidade de interação com o usuário.

Como o foco deste trabalho é a reconstrução tridimensional e não segmentação espacial, optou-se por utilizar um software de código aberto para realizar a segmentação manual, quando o método semi-automático não funciona. A ferramenta *Cloud Compare* (GIRARDEAU, 2012) foi empregada para realizar a segmentação manual das nuvens de pontos.

O usuário seleciona, com uma ferramenta parecida com o "laço poligonal" dos editores de imagens, a área que deseja separar da nuvem de pontos em uma projeção 2D da cena. O programa divide a nuvem e o usuário escolhe qual parte deseja eliminar. O processo pode ser repetido, alterando-se a projeção 2D da cena (através de rotações e translações) até que apenas a região de interesse seja visível. A figura 5.8 ilustra esse processo.



(a) Seleção da área a ser segmentada (seleção verde)

(b) Segmentação da região selecionada

Figura 5.8: Segmentação manual da nuvem através do software *Cloud Compare.*

5.4 Reconstrução e simplificação da superfície

Para a geração de modelos tridimensionais em multirresolução, foram utilizadas a reconstrução de superficies de Poisson (KAZHDAN; BOLITHO; HOPPE, 2006) e a simplificação por contração de arestas com a abordagem de Lindstrom e Turk (1998).

Primeiramente, deve-se estimar os vetores normais para cada ponto do conjunto de dados amostrados. Para isso, utilizou-se o algoritmo desenvolvido por Rusu (2009), que consiste em aproximar o vetor normal de um ponto da superfície como sendo a normal de um plano tangente à superfície nesse ponto. A estimação de vetores normais torna-se, portanto, um problema de ajuste de planos.

De posse dos pontos e seus respectivos vetores normais, é possível reconstruir a superfície com o método de Poisson (KAZHDAN; BOLITHO; HOPPE, 2006). Adaptou-se o algoritmo disponibilizado por Kazhdan, Bolitho e Hoppe (2006) para exportar as malhas poligonais no formato OFF (*Object File Format*). Dessa forma, o objeto pôde ser interpretado pela biblioteca CGAL, responsável pela simplificação *edge collapse*. Tomou-se o cuidado de gerar apenas malhas poligonais *2-manifold* (cada aresta incide apenas em duas faces e cada face incidente em um vértice forma um leque aberto ou fechado (SHENE, 2010)) durante a reconstrução da superfície, pois essa é uma restrição da estrutura de dados *Polyhedron*, responsável por armazenar as malhas poligonais na biblioteca CGAL.

Dois parâmetros devem ser passados para a etapa de reconstrução em multirresolução (reconstrução simplificação da superfície): a profundidade máxima da *octree* que será utilizada durante a reconstrução com o método de Poisson e a porcentagem de simplificação que se deseja obter com a operação de contração de arestas. De acordo com esses parâmetros, podem-se obter malhas com mais ou menos detalhes e, através de experimentos, encontrar as malhas poligonais com o melhor custo-benefício para a aplicação que utilizará os modelos 3D.

Ao final da etapa de simplificação, uma malha poligonal é gerada para o usuário no formato

Wavefront Obj, podendo ser importada pela maior parte de *softwares* modeladores 3D (*3Ds Max*, *Maya*, *Blender* e *ZBrush*, por exemplo) e visualizada.

A figura 5.9 ilustra as etapas para a geração da malha poligonal.

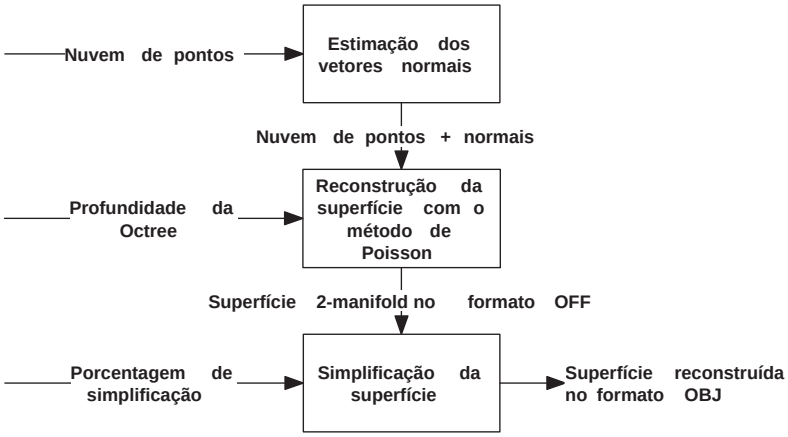


Figura 5.9: Etapas da geração da malha poligonal.

5.5 Considerações finais

O método reconstrução de superfícies de Poisson não oferece um controle preciso da resolução da malha gerada. Observa-se, no gráfico da figura 5.1, que o crescimento do número de polígonos é exponencial, até se atingir um limite (assíntota horizontal), onde a diferença entre uma reconstrução em *octree* n e $n+1$ é mínima. Com a adição da etapa de pós-simplificação, torna-se possível controlar a porcentagem de simplificação da malha, e, consequentemente, mais resoluções podem ser obtidas, de acordo com a necessidade da aplicação que utilizará o modelo 3D. Com base no método proposto, foi implementado um *pipeline* completo de reconstrução de superfícies, apresentado no capítulo 6.

Capítulo 6

Implementação do método proposto

Com o objetivo de validar o método de reconstrução proposto, um protótipo foi desenvolvido. Todas as etapas foram implementadas, possibilitando a construção de um sistema completo de reconstrução tridimensional.

6.1 Considerações iniciais

O *pipeline* de reconstrução 3D apresentado no capítulo 5 foi implementado na linguagem C++ com o ambiente de desenvolvimento Microsoft Visual Studio 2010. Foram utilizadas duas bibliotecas para auxiliar o desenvolvimento do protótipo: PCL, especializada em processamento de nuvens de pontos, e CGAL, destinada ao processamento geométrico. Neste capítulo, os detalhes de implementação serão discutidos. Primeiramente, o ambiente experimental será apresentado. Uma visão geral das bibliotecas será exposta. Em seguida, serão abordadas todas as etapas do processo de reconstrução, desde a aquisição dos dados até a pós-simplificação da superficie.

6.2 Ambiente experimental

Configurou-se o ambiente experimental da seguinte maneira:

6.2.1 Hardware

- 1 Computador pessoal:
  - Processador: Intel(R) Core(TM) i5-2400. 3.10GHz;
  - Memória RAM: 8 GB;
  - Placa de vídeo: GeForce GTX 550 Ti (Clock de 900 MHz, 1GB de memória GDDR5, 192 processadores);

  - Disco rígido: SATA 1 TB;
  - Placa mãe: Asus LGA1155 P8H61 PRO.

- 1 Sensor Microsoft Kinect (Modelo XBOX 360).

6.2.2 Software

- Sistema operacional: Windows 7 Ultimate SP1;
- Ambiente de desenvolvimento: Microsoft Visual Studio 2010;
- Sistema de configuração de projetos: CMake 2.8.9;
- Biblioteca PCL 1.6.0 e dependências:
  - Boost 1.49;
  - Eigen 3.0.5;
  - FLANN 1.7.1;
  - VTK 5.8.0;
  - Qt 4.8.0;
  - QHull 2011.1 (6.2.0.1385);
  - OpenNI 1.3.2;
  - Sensor 5.0.3.3.
- Biblioteca CGAL 4.0.

6.3 Bibliotecas utilizadas

(a) PCL (RUSU; COUSINS, 2011) (b) CGAL (PROJECT, 2012)

Figura 6.1: Logotipo das bibliotecas utilizadas

6.3.1 Biblioteca PCL

Com o surgimento de equipamentos baratos de aquisição 3D, tornou-se possível a aplicação da reconstrução de superfícies em ambientes robóticos. Neste contexto, surge a PCL (*Point Cloud Library*) , uma biblioteca robusta e de código aberto, desenvolvida para trabalhar com nuvens de pontos n-dimensionais. Foi escrita em C++ empregando-se diversas bibliotecas de otimização, entre elas a *Intel Threading Building Blocks* para paralelização, FLANN para busca de vizinhos mais próximos, BOOST para o uso de ponteiros compartilhados e *threading*. A PCL inclui algoritmos de filtragem, estimação de características, segmentação, reconstrução de superfícies, visualização, entre outros. Além disso, é integrada com a biblioteca OpenNI, portanto pode adquirir dados diretamente de dispositivos como o Microsoft Kinect, PrimeSensor 3D e Asus XTionPRO (RUSU; COUSINS, 2011).

6.3.1.1 O formato PCD

PCD (*Point Cloud Data*) é um formato de arquivo adotado pela biblioteca PCL para armazenar nuvens de pontos (RUSU; COUSINS, 2011). Atualmente, esse formato encontra-se na versão 0.7.

O cabeçalho deste formato tem a seguinte estrutura:

- Versão (*VERSION*): versão do arquivo PCD;
- Campos (*FIELDS*): descrição dos campos (dimensão) do arquivo, por exemplo, “x y z”, “x y z rgb” e “x y z normal_x normal_y normal_z”;
- Tamanho (*SIZE*): tamanho, em bytes, de cada dimensão do arquivo;
- Tipo (*TYPE*): especifica o tipo de cada dimensão, por exemplo, inteiro ou caractere;
- Quantidade (*COUNT)*: quantidade de pontos no arquivo;
- Altura (*HEIGHT*) e Largura (*WIDTH*): podem especificar a quantidade de pontos na arquivo ou as dimensões de altura e largura da nuvem de pontos;
- Ponto de vista (*VIEWPOINT*): especifica o ponto de vista da aquisição da nuvem de pontos, se disponível;
- Dados (*DATA*): especifica se o arquivo é armazenado em formato binário ou ASCII.

Após o cabeçalho, segue-se um conjunto de linhas (ASCII) ou *stream* de bytes (binário) contendo os pontos do arquivo. Observa-se na figura 6.2 um exemplo de arquivo PCD.

```
# . PCD v.7 - Point Cloud Data file format VERSION .7

FIELDS x y z rgb

SIZE 4 4 4 4

TYPE F F F F

COUNT 1 1 1 1

WIDTH 213

HEIGHT 1

VIEWPOINT 0 0 0 1 0 0 0

POINTS 213

DATA ascii

0.93773 0.33763 0 4.2108e +06

0.90805 0.35641 0 4.2108e +06

0.81915 0.32 0 4.2108e +06

0.97192 0.278 0 4.2108e +06

0.944 0.29474 0 4.2108e +06

0.98111 0.24247 0 4.2108e +06

...
```

Figura 6.2: Formato do arquivo PCD

6.3.2 Biblioteca CGAL

CGAL (*Computational Geometry Algorithms Library*) é um projeto de código aberto, escrito em C++, que oferee estruturas de dados e algoritmos geométricos, de modo eficiente e robusto. A biblioteca possui algoritmos de volumes envoltórios, superfícies poliédricas, operações booleanas, triangulações, diagramas de *Voronoi*, geração, subdivisão e simplificação de malhas, entre muitos outros (PROJECT, 2012). Foi escrita de forma genérica e baseada em *templates* para facilitar o desenvolvimento de algoritmos geométricos.

6.3.2.1 O formato OFF

A biblioteca CGAL utiliza como entrada e saída padrão de dados geométricos o formato OFF (*Object File Format*), também utilizado pelo programa *Geomview*. O formato possui ferramentas de definição de geometria e materiais, podendo ser armazenado em formato binário ou texto (ASCII) (PROJECT, 2012).

O arquivo tem início com a palavra-chave "OFF". Na linha seguinte são escritos o número de vértices e faces. Um exemplo de arquivo OFF é apresentado na figura 6.3 (o texto após "#" é um comentário de linha):

```
# início do arquivo

OFF

# número de vértices e faces

7488 14947

# listagem de vértices

0 -0.095455 0.707741 -0.047400

-0.095455 0.724000 -0.069262

...

# listagem de faces

3 1926 1928 1927

3 1926 1923 1919
```

Figura 6.3: Formato do arquivo OFF

6.4 Implementação do *pipeline*

Nesta seção serão apresentados detalhes de implementação do *pipeline* de reconstrução 3D.

6.4.1 Captura do *frame* RGB-D

A captura do frame RGB-D é feita através da classe *OpenNIGrabber*, da biblioteca PCL. O algoritmo 6.1 apresenta um trecho de código C++ com a utilização desta classe. Primeiramente, a classe *OpenNIGrabber* é instanciada e registra-se uma função de *callback* para ela (cada vez que o *Kinect* fornecer uma nuvem, a função é chamada). Na função de *callback*, a nuvem é salva com o nome "nuvem.pcd" e o programa termina.

Algoritmo 6.1 Captura de um *frame* RGB-D.

```
int main(void){
//Programa principal
    kinectGrabber = new OpenNIGrabber();
    boost::function<void (const PointCloud<PointXYZRGB>::ConstPtr&)> f = bo-
ost::bind(&grabberCallback, _1);
    kinectGrabber->registerCallback(f);
}
//Função de callback OpenNI
void grabberCallback(const PointCloud<PointXYZRGB>::ConstPtr& cloud){
    io::savePCDFile(“nuvem.pcd”, *cloud, true);
    exit(0);
}
```

O processo de captura e combinação de vários *frames* RGB-D por meio do algoritmo ICP (BESL; McKay, 1992) é mais complexo e não será apresentado neste trabalho. Caso o leitor se interesse, recomenda-se o estudo do módulo *KinFu*, disponível na biblioteca PCL (RUSU; COUSINS, 2011).

6.4.2 Remoção de valores inválidos

A remoção dos valores inválidos é apresentada no algoritmo 6.2. A nuvem de pontos é lida. Removem-se os valores inválidos e, por fim, a nuvem filtrada é salva em arquivo.

Algoritmo 6.2 Remoção de valores inválidos da nuvem

```
pcl::io::loadPCDFile (“nuvem_origem.pcd”, cloud_blob);
pcl::fromROSMsg (cloud_blob, *cloud);//leitura da nuvem
pcl::removeNaNFromPointCloud(*origem,*destino);//remoção dos valores inválidos
pcl::io::savePCDFile ("nuvem_filtrada.pcd", cloud);
```

6.4.3 Cálculo dos vetores normais

Para o cálculo dos vetores normais, a classe *NormalEstimation*, da biblioteca PCL, é utilizada. Instancia-se a classe com *template* da nuvem de pontos desejada. Cria-se uma árvore de busca e define-se quantos vizinhos de cada ponto serão considerados para a estimação dos planos. Computam-se as normais e, por fim, cria-se uma nuvem de pontos contendo pontos e normais.

Algoritmo 6.3 Cálculo dos vetores normais

```
pcl::NormalEstimation<pcl::PointXYZ, pcl::Normal> n; //instancia a classe de estimação
pcl::PointCloud<pcl::Normal>::Ptr normals (new pcl::PointCloud<pcl::Normal>); //nuvem de pontos contendo apenas normais
pcl::search::KdTree<pcl::PointXYZ>::Ptr tree (new pcl::search::KdTree<pcl::PointXYZ>); //instancia-se uma árvore de busca
/*————————————————-cálculo das normais————————————————*/
tree->setInputCloud (cloud);
n.setInputCloud (cloud);
n.setSearchMethod (tree);
n.setKSearch (20); //define-se que 20 vizinhos serão considerados
n.compute (*normals); //calculam-se as normais
/*————————————combinação dos pontos com as normais————————————*/
pcl::PointCloud<pcl::PointNormal>::Ptr cloud_with_normals (new pcl::PointCloud<pcl::PointNormal>);
pcl::concatenateFields (*cloud, *normals, *cloud_with_normals);
/*————————————————salva a nuvem em arquivo————————————————*/
pcl::io::savePCDFile ("nuvem_com_normais.pcd", cloud_with_normals);
```

6.4.4 Reconstrução de Poisson

O código-fonte da reconstrução de Poisson disponibilizado por Kazhdan (2012) foi utilizado neste *pipeline.* Uma função para exportar a superfície reconstruída para o formato OFF foi implementada e adicionada ao projeto. Os seguintes parâmetros são passados para o algoritmo de reconstrução:

- Nuvem de pontos de origem;
- Arquivo de destino;
- Profundidade máxima da octree;
- Garantir superfície *2-manifold.*

6.4.5 Simplificação da superfície

A simplificação da superfície com o método de contração de arestas de Lindstrom e Turk (1998) é realizada com o auxílio da biblioteca CGAL. Utiliza-se a classe *edge_collapse*, do módulo *CGAL::Surface_mesh_simplification*, para realizar a simplificação da malha. O trecho de código responsável pela simplificação é apresentado no algoritmo 6.4.

Algoritmo 6.4 Simplificação de contração de artestas

```
std::ifstream is("superficie.off") ; //leitura do arquivo
CGAL::scan_OFF(is, surface, true); //armazenamento da superfície na variável "surface"
stop = 0.1 //simplifica a superfície até que o número de vértices seja 10% do original (simplificação de 90%).
CGAL::Surface_mesh_simplification::edge_collapse (surface ,stop); //Realiza a simplificação
```

6.5 Interface desenvolvida

Uma interface gráfica (GUI) foi desenvolvida para facilitar o processo de reconstrução de superfícies, desde a captura da nuvem de pontos até a pós-simplificação da malha. A linguagem C# foi utilizada para o desenvolvimento desse programa.

A tela de captura do Kinect oferece opções sobre o formato da nuvem de pontos e o tipo de captura (1 *frame* ou combinação de *frames* (IZADI et al., 2011)). A figura 6.4 ilustra essa opção.

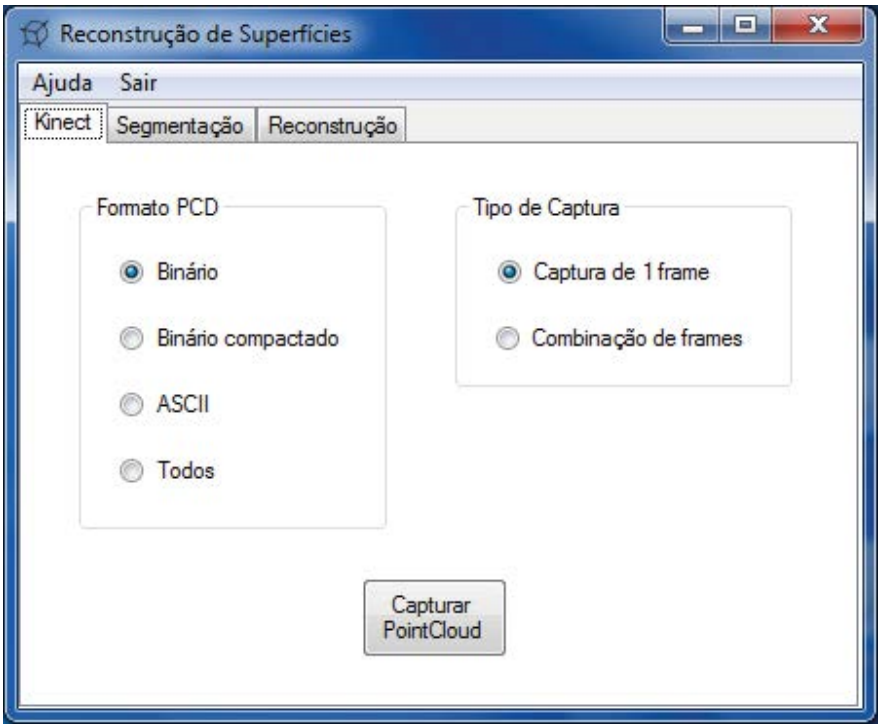


Figura 6.4: Interface de captura da nuvem de pontos.

A GUI de segmentação semi-automática é apresentada na figura 6.5.

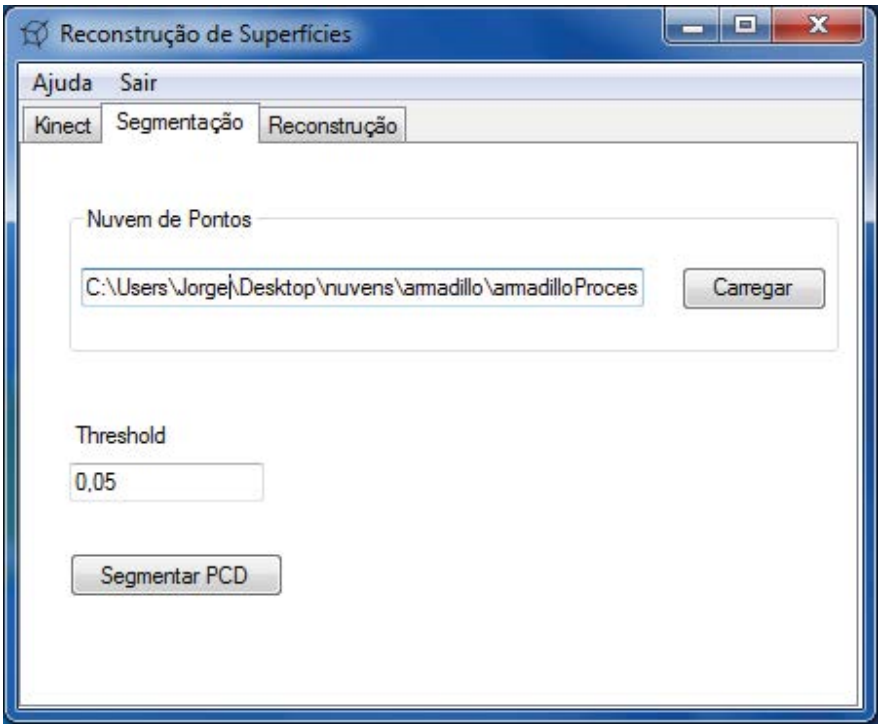


Figura 6.5: Interface de segmentação semi-automática.

A GUI de reconstrução e simplificação de superfícies solicita ao usuário a profundidade máxima de *octree* e a porcentagem de pós-simplificação, como apresentado na figura 6.6.

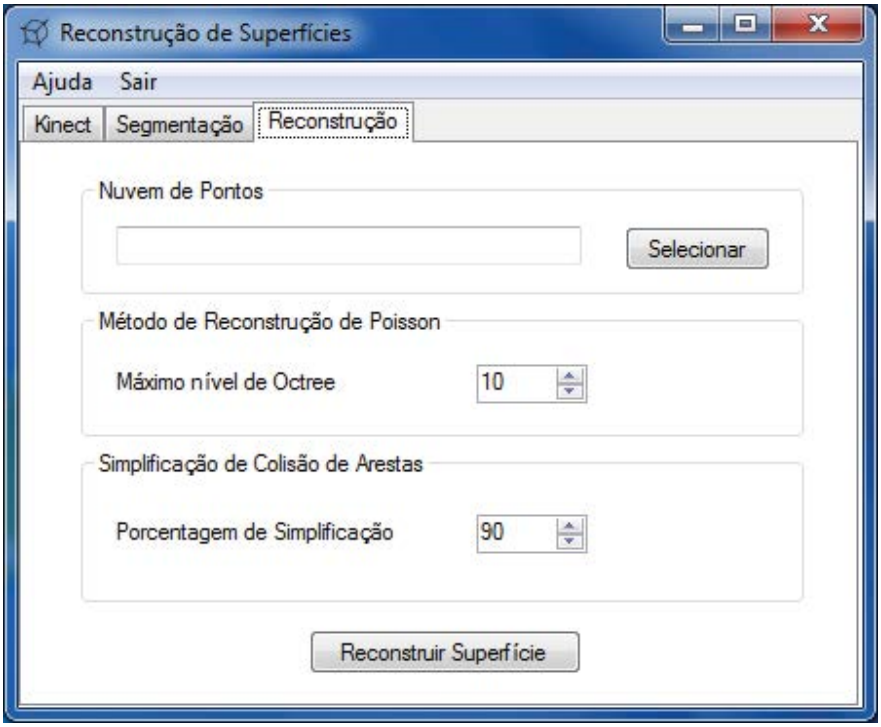


Figura 6.6: Interface de reconstrução e simplificação.

6.6 Considerações finais

Os algoritmos de captura da nuvem de pontos, reconstrução e simplificação de superfícies foram desenvolvidos. Para facilitar a utilização dos programas, uma interface gráfica foi construída. Isso possibilita que usuários básicos (*designers* e usuários domésticos, por exemplo) consigam interagir com o software sem a necessidade de trabalhar com a linha de comando. De posse do *pipeline* de reconstrução implementado, é possível testar a validade do método proposto. O capítulo 7 apresenta os experimentos realizados neste trabalho.

Capítulo 7

EXPERIMENTOS

Neste capítulo serão apresentados os experimentos realizados.

7.1 Considerações iniciais

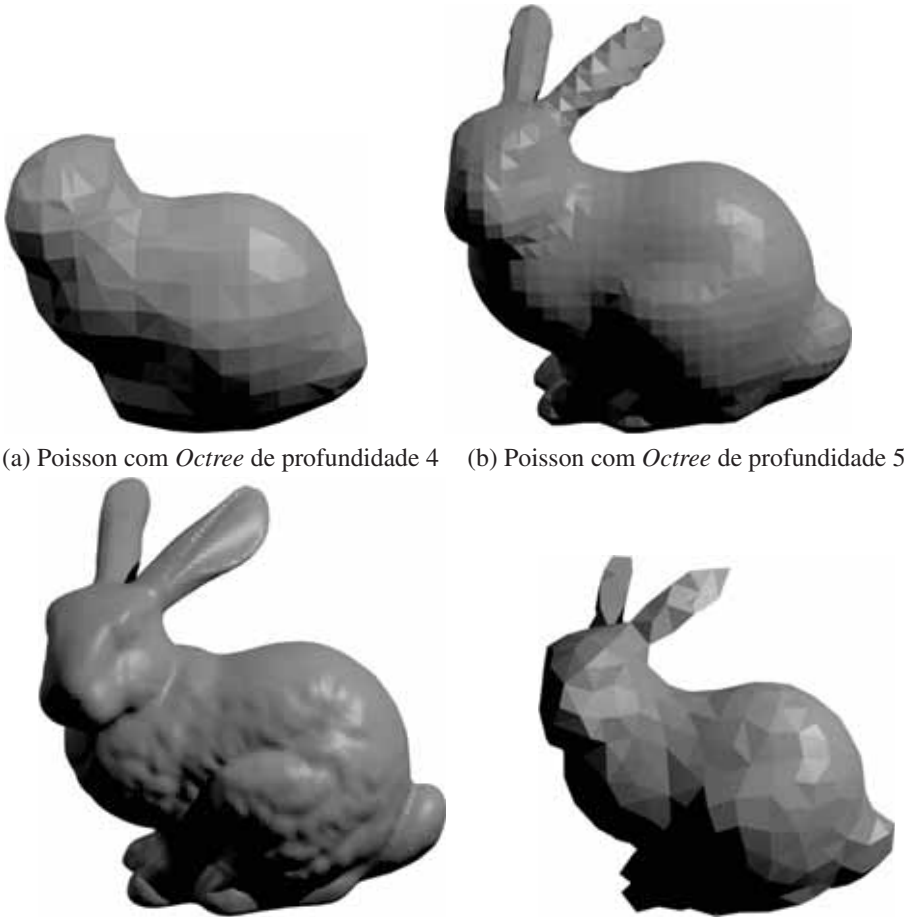
O algoritmo de reconstrução proposto foi testado em dados experimentais (nuvens de pontos provenientes do dispositivo Microsoft Kinect) e de domínio público (Repositório 3D de Stanford (STANFORD, 2011) e Washington RGB-D *Dataset (Washington, 2012)*). Os objetos escaneados foram reconstruídos em 3 profundidades de *octree* e, em seguida, a reconstrução de maior LOD foi simplificada. Para cada experimento, escolheu-se uma porcentagem de simplificação tal que a quantidade de vértices da malha simplificada seja próxima a da reconstrução de menor LOD, permitindo a comparação visual dos resultados.

7.2 Repositório 3D de Stanford

Aplicou-se o algoritmo proposto ao modelo "Stanford *Bunny*" (STANFORD, 2011), apresentado na figura 7.1. A tabela 7.1 apresenta a quantidade de vértices e triângulos de malha. Observa-se que, em resoluções mais baixas, perdem-se detalhes muito importantes para a representação do objeto (na figura 7.1a, as orelhas do coelho). Ao se combinar a reconstrução de Poisson com a simplificação de contração de arestas, é possível reconstruir malhas simplificadas sem perder tais detalhes. Destaca-se que a quantidade de vértices das reconstruções em 7.1a e 7.1d é praticamente a mesma.

O algoritmo também foi aplicado ao modelo *Armadillo* (STANFORD, 2011). As reconstruções são apresentadas na figura 7.2 e na tabela 7.2. Novamente, obtêm-se modelos simplificados superiores com o método proposto. Ressalta-se que, desta vez, a quantidade de vértices da superfície reconstruída com Poisson em *octree* 5 (figura 7.2a) é superior à reconstrução com o método pro-

posto (figura 7.2d).



(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 7.1: Algoritmo aplicado aos dados do Coelho de Stanford (STANFORD, 2011).

	Nº de Vértices	Nº de Faces
Poisson com *Octree* de Profundidade 4	490	976
Poisson com *Octree* de Profundidade 5	2160	4316
Poisson com *Octree* de Profundidade 7	36322	72640
Poisson com *Octree* de Profundidade 7 e pós-simplificação	491	978

Tabela 7.1: Resultados da reconstrução do Coelho de Stanford.

(a) Poisson com *Octree* de profundidade 6 (b) Poisson com *Octree* de profundidade 7

(c) Poisson com *Octree* de profundidade 8 (d) Poisson com *Octree* de profundidade 8 e pós-simplificação

Figura 7.2: Algoritmo aplicado aos dados do modelo *Armadillo* de Stanford (STANFORD, 2011).

	Nº de Vértices	Nº de Faces
Poisson com *Octree* de Profundidade 6	7318	14636
Poisson com *Octree* de Profundidade 7	32364	64716
Poisson com *Octree* de Profundidade 8	141114	282260
Poisson com *Octree* de Profundidade 8 e pós-simplificação	7040	42336

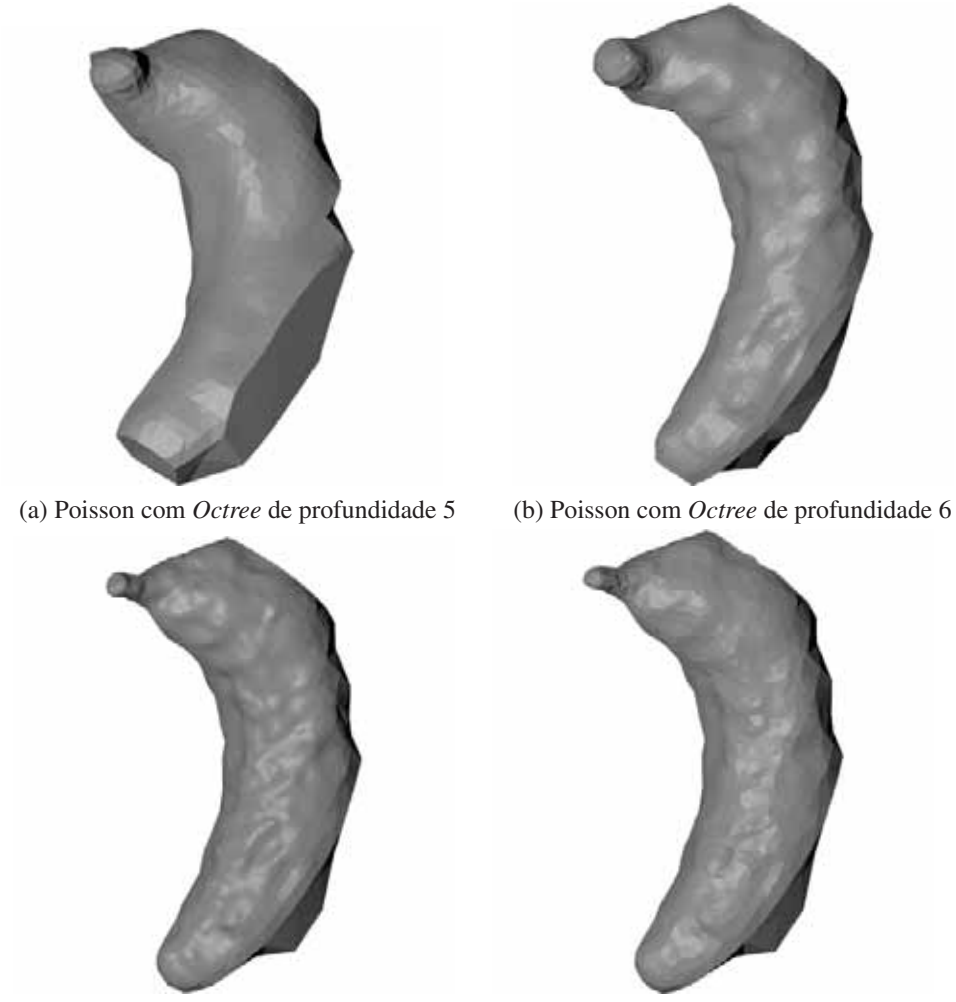
Tabela 7.2: Resultados da reconstrução do *Armadillo* de Stanford.

7.3 Washington RGB-D *Dataset*

O Repositório RGB-D de Washington (Washington, 2012) é um banco de dados com 300 objetos domésticos, organizados em 51 categorias. O instrumento de captura desse repositório foi o Microsoft Kinect. Foram realizados alguns experimentos com objetos dessa base de dados. Observou-se

que, como apenas um quadro RGB-D foi utilizado para reconstruir cada superfície, os resultados de reconstrução apresentaram muitos ruídos.

A figura 7.3 ilustra os experimentos realizados com a nuvem de pontos do objeto "banana". A quantidade de vértices e triângulos é apresentada na tabela 7.3. Mesmo com uma grande redução do número de polígonos, observa-se que o modelo reconstruído com o algoritmo proposto (com pós-simplificação) apresenta mais detalhes do que a reconstrução com *octree.*



(a) Poisson com *Octree* de profundidade 5

(b) Poisson com *Octree* de profundidade 6

(c) Poisson com *Octree* de profundidade 10

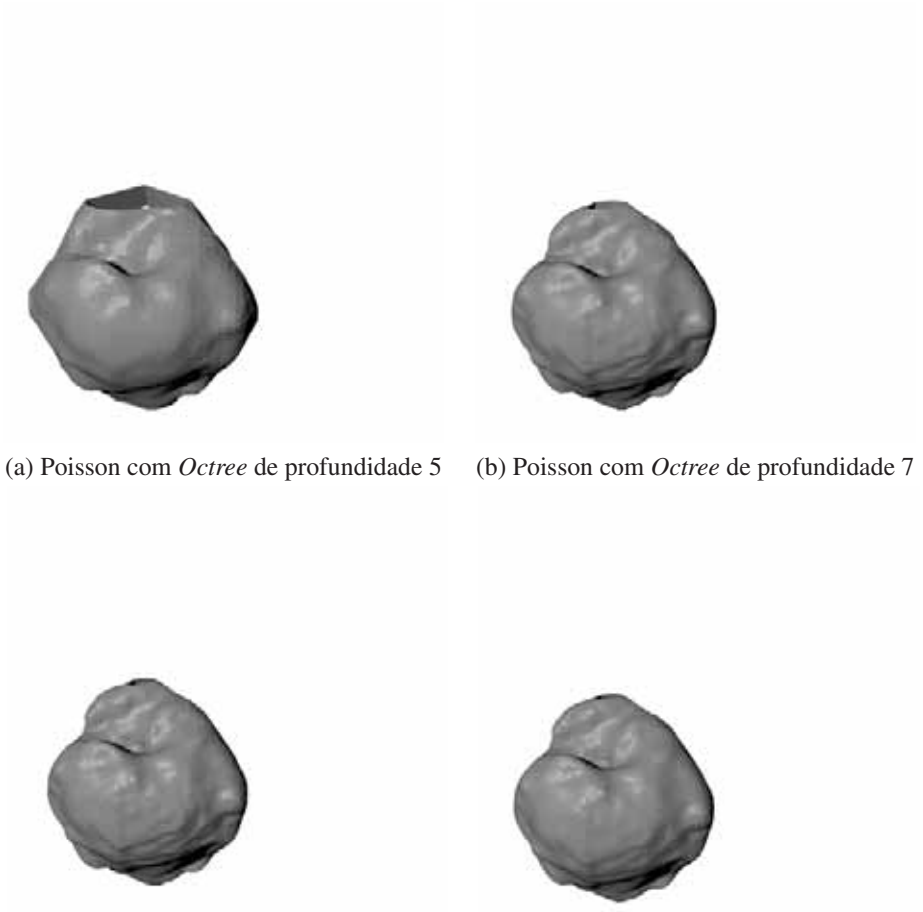
(d) Poisson com *Octree* de profundidade 10 e pós-simplificação

Figura 7.3: Algoritmo aplicado aos dados de uma banana.

	Nº de Vértices	Nº de Faces
Poisson com *Octree* de Profundidade 5	863	1678
Poisson com *Octree* de Profundidade 6	2546	5052
Poisson com *Octree* de Profundidade 10	8436	16843
Poisson com *Octree* de Profundidade 10 e pós-simplificação	933	1845

Tabela 7.3: Resultados da reconstrução de uma banana.

A figura 7.4 e a tabela 7.4 apresentam os resultados das reconstruções do modelo maçã, do repositório de Washington.



(a) Poisson com *Octree* de profundidade 5

(b) Poisson com *Octree* de profundidade 7

(c) Poisson com *Octree* de profundidade 10

(d) Poisson com *Octree* de profundidade 10 e pós-simplificação

Figura 7.4: Algoritmo aplicado aos dados de uma maçã.

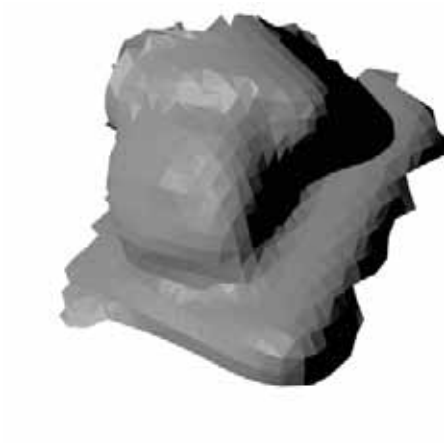
	Nº de Vértices	Nº de Faces
Poisson com *Octree* de Profundidade 5	1932	3814
Poisson com *Octree* de Profundidade 7	9070	18110
Poisson com *Octree* de Profundidade 10	12122	24213
Poisson com *Octree* de Profundidade 10 e pós-simplificação	1338	2656

Tabela 7.4: Resultados da reconstrução da maçã.

7.4 Dados do Kinect

Foram realizados experimentos com as nuvens de pontos provenientes do *Microsoft Kinect*. Os resultados são apresentados a seguir.

A figura 7.5 ilustra a reconstrução de um telefone. A quantidade de polígonos é apresentada na tabela 7.5.



(a) Poisson com *Octree* de profundidade 5

(b) Poisson com *Octree* de profundidade 7

(c) Poisson com *Octree* de profundidade 10

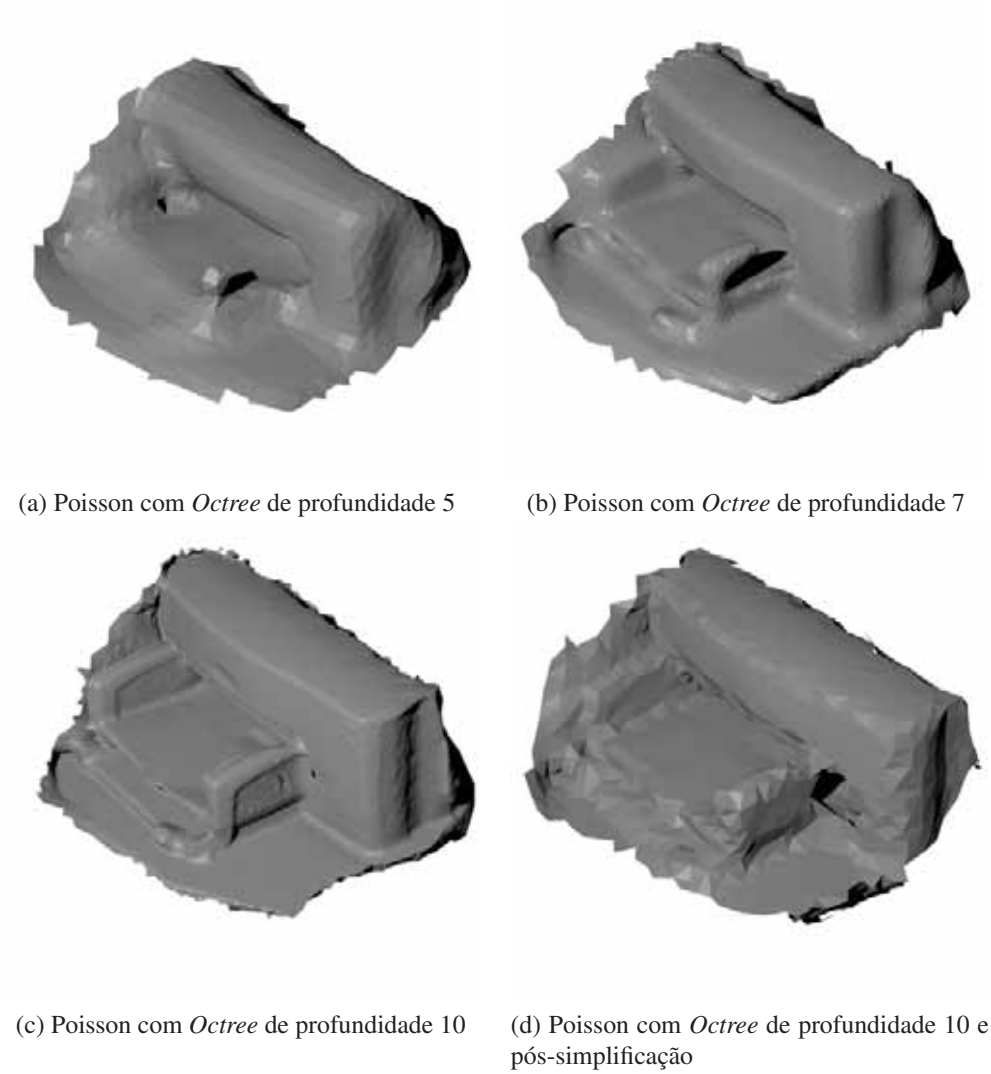
(d) Poisson com *Octree* de profundidade 10 e pós-simplificação

Figura 7.5: Algoritmo aplicado aos dados de um telefone.

	Nº de Vértices	Nº de Faces
Poisson com *Octree* de Profundidade 5	1430	2819
Poisson com *Octree* de Profundidade 7	15762	31444
Poisson com *Octree* de Profundidade 10	216748	433316
Poisson com *Octree* de Profundidade 10 e pós-simplificação	2502	4832

Tabela 7.5: Resultados da reconstrução de um telefone.

A figura 7.6 ilustra a reconstrução de uma impressora. A quantidade de polígonos é apresentada na tabela 7.6.



(a) Poisson com *Octree* de profundidade 5

(b) Poisson com *Octree* de profundidade 7

(c) Poisson com *Octree* de profundidade 10

(d) Poisson com *Octree* de profundidade 10 e pós-simplificação

Figura 7.6: Algoritmo aplicado aos dados de uma impressora.

	Nº de Vértices	Nº de Faces
Poisson com *Octree* de Profundidade 5	1430	2819
Poisson com *Octree* de Profundidade 7	15762	31444
Poisson com *Octree* de Profundidade 10	216748	433316
Poisson com *Octree* de Profundidade 10 e pós-simplificação	2502	4832

Tabela 7.6: Resultados da reconstrução de uma impressora.

A figura 7.7 ilustra a reconstrução de uma figura humana. A quantidade de polígonos é apresentada na tabela 7.7.

(a) Poisson com *Octree* de profundidade 5

(b) Poisson com *Octree* de profundidade 7

(c) Poisson com *Octree* de profundidade 10

(d) Poisson com *Octree* de profundidade 10 e pós-simplificação

Figura 7.7: Algoritmo aplicado aos dados de uma figura humana.

	Nº de Vértices	Nº de Faces
Poisson com *Octree* de Profundidade 5	1938	3815
Poisson com *Octree* de Profundidade 7	21669	43239
Poisson com *Octree* de Profundidade 10	285886	571574
Poisson com *Octree* de Profundidade 10 e pós-simplificação	2024	3856

Tabela 7.7: Resultados da reconstrução de uma figura humana.

7.5 Considerações finais

Através dos experimentos, conclui-se que o algoritmo proposto obtém reconstruções superiores de objetos em LOD menor, em comparação com o método de Poisson. Como o usuário pode especificar a porcentagem da simplificação desejada, é possível obter um número praticamente ilimitado de reconstruções. Selecionando-se as melhores combinações de parâmetros para reconstruções em vários LOD (profundidade da *octree* e porcentagem de simplificação), pode-se obter transições quase imperceptíveis entre níveis de detalhe (*geomorphing*). Observa-se também que o método de reconstrução de Poisson perde detalhes essenciais para a identificação da superfície quando níveis de *octree* muito baixos são empregados (por exemplo, na reconstrução do coelho de Stanford, figura 7.1a), o que indica que tais profundidades não devem ser utilizadas no processo de reconstrução.

Capítulo 8

CONCLUSÕES E TRABALHOS FUTUROS

Neste capítulo, serão apresentadas as conclusões da pesquisa e os trabalhos futuros.

8.1 Conclusões

O desenvolvimento do presente trabalho proporcionou a investigação do processo de reconstrução e simplificação de superfícies. As principais técnicas referentes a captura e processamento de dados provenientes de sensores espaciais foram apresentadas e comparadas. Essa investigação permitiu a combinação de métodos existentes na criação de uma nova solução para a reconstrução de superfícies visando a multirresolução. A viabilidade desta nova solução foi comprovada por meio de sua implementação e testes experimentais.

Por meio dos experimentos realizados, observou-se que o dispositivo Microsoft Kinect é capaz de trabalhar como um *scanner* 3D e, apesar de ser uma opção de baixo custo, pode adquirir nuvens de pontos com boa precisão, desde que sejam utilizados métodos para combinação de frames RGB-D, por exemplo, o registro das nuvens de pontos com o algoritmo *Iterative Closest Points* (BESL; McKay, 1992).

Verificou-se que o método de reconstrução de Poisson apresenta aproximações suavizadas, mas ainda assim muito próximas, à superfície amostrada. Por isso, é ideal para reconstruir objetos escaneados por meio do sensor Kinect, pois seus dados são ruidosos e não apresentam resoluções muito altas. Entretanto, seu resultado para reconstruções em multirresolução (variando-se a profundidade máxima da *octree*) não é satisfatório, pois embora a quantidade de polígonos da malha reconstruída seja consideravelmente menor, a topologia do objeto original muda completamente quando se deseja resoluções muito baixas (ver figura 7.1a).

Para solucionar esse problema, o método de reconstrução de Poisson foi combinado com a sim-

plificação por contração de arestas, tornando possível atingir malhas mais simplificadas, mantendo-se a topologia original da superfície. Para atingir tal resultado, são necessários parâmetros adequados para a reconstrução de Poisson (Profundidade da *Octree*) e a simplificação *Edge Collapse* (porcentagem de simplificação).

Conclui-se que o método apresentado neste trabalho é válido e pode ser utilizado para reconstruir objetos em diferentes níveis de detalhe, que podem ser inseridos em ferramentas de realidade misturada em geral (jogos, ferramentas de treinamento virtual, simulação e estúdios virtuais aumentados, por exemplo).

Este trabalho de conclusão de curso deu origem a dois trabalhos científicos, publicados e apresentados: "*Poisson Surface Reconstruction with Local Mesh Simplification*", Sibgrapi 2012 (disponível no Apêndice A) e "Reconstrução de Superfícies com Método de Poisson e Simplificação Local de Malhas", CIC 2012 (disponível no Apêndice B).

8.2 Trabalhos futuros

Visualizaram-se quatro trabalhos futuros, descritos a seguir:

8.2.1 Verificação da qualidade das malhas

Utilização de métricas para verificar a qualidade das malhas em multirresolução. A ferramenta Metro (CIGNONI; ROCCHINI; SCOPIGNO, 1998) pode ser utilizada para comparar duas malhas poligonais, por exemplo, a malha reconstruída e a malha simplificada, verificando-se diferenças de volume, área, diâmetro e erro médio.

8.2.2 Desenvolvimento de uma segunda abordagem de simplificação

Implementar o algoritmo de simplificação de malhas por coplanaridade descrito por Mingyi, Bangshu e Huajing (2004) e adicioná-lo ao *pipeline* de reconstrução 3D. Deseja-se substituir o algoritmo de contração de arestas por essa abordagem de simplificação e comparar a qualidade das malhas resultantes.

8.2.3 Aplicação de um filtro de suavização na imagem de profundidade

O ruído multiplicativo, característico da radiação infravermelha, está presente nos dados de profundidade captados pelo sensor Kinect (GULO, 2012). O método variacional, desenvolvido por Jin e Yang (2011), pode ser aplicado na correção desse tipo de ruído. Gulo et al. (2012) paralelizaram o método variacional por meio da arquitetura CUDA. Os autores obtiveram uma taxa de filtragem de

aproximadamente 6 quadros por segundo (tempo médio de suavização de 255 quadros: 39 segundos, medidos em 50 execuções do algoritmo) em uma placa de vídeo GeForce GTX 450 (Nvidia) com 196 *CUDA Cores* e 1GB de memória. Como trabalho futuro, deseja-se adicionar esse filtro ao *pipeline* apresentado neste documento, melhorando-se ainda mais os resultados da reconstrução.

8.2.4 Investigação do uso de *Wavelets* para realizar a simplificação

O método de reconstrução de Poisson (KAZHDAN; BOLITHO; HOPPE, 2006) aproxima uma nuvem de pontos através de uma solução implícita da equação de Poisson. Um possível trabalho futuro é a investagação do uso de *Wavelets* para simplificar essa aproximação, de modo que detalhes importantes não sejam perdidos em LODs menores, como ocorre com a multirresolução do método de Poisson.

Referências Bibliográficas

ALGORRI, M.; SCHMITT, F. Surface reconstruction from unstructured 3D data. In: *Computer graphics forum*. [S.l.: s.n.], 1996. v. 15, p. 47–60.

BAADER, A.; HIRZINGER, G. Three-dimensional surface reconstruction based on a self-organizing feature map. In: *Proc. 6th Int. Conf. Advan. Robotics*. [S.l.: s.n.], 1993. p. 273–278.

BESL, P.; McKay, H. A method for registration of 3-d shapes. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, v. 14, n. 2, p. 239 –256, fev. 1992. ISSN 0162-8828.

BÖHLER, W.; MARBS, A. 3D scanning instruments. *Proceedings of the CIPA WG*, v. 6, p. 9e18, 2002.

BOISSONNAT, J. Geometric structures for three-dimensional shape representation. *ACM Transactions on Graphics (TOG)*, v. 3, n. 4, p. 266–286, 1984.

BOTSCH, M.; KOBBELT, L. A robust procedure to eliminate degenerate faces from triangle meshes. In: *Proc. of Vision, Modeling, and Visualization*. [S.l.: s.n.], 2001. v. 1, p. 283–289.

CARBONE, V. et al. Combination of a vision system and a coordinate measuring machine for the reverse engineering of freeform surfaces. *The International Journal of Advanced Manufacturing Technology*, v. 17, n. 4, p. 263–271, 2001. ISSN 0268-3768. 10.1007/s001700170179. Disponível em: <http://dx.doi.org/10.1007/s001700170179>.

CARR, J. et al. Reconstruction and representation of 3D objects with radial basis functions. In: *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*. [S.l.: s.n.], 2001. p. 67–76.

CHETVERIKOV, D. et al. The trimmed iterative closest point algorithm. In: *Pattern Recognition, 2002. Proceedings. 16th International Conference on*. [S.l.: s.n.], 2002. v. 3, p. 545–548.

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.

CLARK, J. Hierarchical geometric models for visible surface algorithms. *Communications of the ACM*, v. 19, n. 10, p. 547–554, 1976.

CRUZ, L.; LUCIO, D.; VELHO, L. Kinect and RGBD images: Challenges and applications. In: WALTER, L. Y. M. (Ed.). *SIBGRAPI 2012 (XXV Conference on Graphics, Patterns and Images Tutorials (SIBGRAPI-T))*. Ouro Preto, MG, Brazil: [s.n.], 2012. Disponível em: <http://www.decom.ufop.br/sibgrapi2012/index.php/call/tp>.

DANIELS, J. et al. Quadrilateral mesh simplification. In: *ACM Transactions on Graphics (TOG)*. [S.l.: s.n.], 2008. v. 27, p. 148.

FARO. *LASER SCANNER*. 2012. Disponível em: <http://www.faro.com/focus/br>.

FISCHLER, M.; BOLLES, R. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, v. 24, n. 6, p. 381–395, 1981.

FLORIANI, L. et al. *Surface Simplification Algorithms Overview*. 1999.

GE Healthcare. *Magnetic Resonance Imaging*. 2012. Disponível em: <http://www.gehealthcare.com/euen/mri/products/>.

GERSTNER, T. Multiresolution extraction and rendering of transparent isosurfaces. *Computers & Graphics*, v. 26, n. 2, p. 219–228, 2002.

GERSTNER, T.; PAJAROLA, R. Topology preserving and controlled topology simplifying multiresolution isosurface extraction. In: *Proceedings of the conference on Visualization'00*. [S.l.: s.n.], 2000. p. 259–266.

GIRARDEAU, D. *Cloud Compare*. 2012. Disponível em: <http://www.danielgm.net/cc/>.

GOIS, J. P. *Reconstrução de superfícies a partir de nuvem de pontos*. Tese (Doutorado) — ICMC - USP, 2004.

GROSS, M. H.; GATTI, R.; STAADT, O. Fast multiresolution surface meshing. In: *Proceedings of the 6th conference on Visualization '95*. Washington, DC, USA: IEEE Computer Society, 1995. (VIS '95), p. 135–. ISBN 0-8186-7187-4. Disponível em: <http://dl.acm.org/citation.cfm?id=832271.833866>.

GULO, C. et al. Método de suaviza\ccão de imagem baseado num modelo variacional paralelizado em arquitetura CUDA. In: *CSBC 2012-XXXII Congresso da Sociedade Brasileira de Computa\ccão*. [S.l.: s.n.], 2012.

GULO, C. A. S. J. *Técnicas de paralelização em GPGPU aplicadas em algoritmo para remoção de ruído multiplicativo*. Tese (Mestrado) — Universidade Estadual Paulista "Júlio de Mesquita Filho", Bauru - SP, 2012.

HENRY, P. et al. RGB-D mapping: Using depth cameras for dense 3D modeling of indoor environments. In: *Proceedings of the 12th International Symposium on Experimental Robotics*. [S.l.: s.n.], 2010.

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. et al. 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>.

HOPPE, H. et al. *Surface reconstruction from unorganized points*. Tese (Doutorado) — University of Washington, 1994.

IZADI, S. et al. KinectFusion: real-time 3D reconstruction and interaction using a moving depth camera. In: *Proceedings of the 24th annual ACM symposium on User interface software and technology*. [S.l.: s.n.], 2011. p. 559–568.

JIN, Z.; YANG, X. A variational model to remove the multiplicative noise in ultrasound images. *Journal of Mathematical Imaging and Vision*, v. 39, n. 1, p. 62–74, 2011.

KAZHDAN, M. *Screened Poisson Surface Reconstruction (Version 4)*. 2012. Disponível em: <http://www.cs.jhu.edu/ misha/Code/PoissonRecon/Version4/>.

KAZHDAN, M.; BOLITHO, M.; HOPPE, H. Poisson surface reconstruction. In: *Proceedings of the fourth Eurographics symposium on Geometry processing*. [S.l.: s.n.], 2006. p. 61–70.

LEE, A. W. F. et al. 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>.

LEE, H.; JUHO, L.; YANG, H. *Real-time LOD: Marching-cube-and-octree-based 3D Object Level-of-detail Modeling*. [S.l.]: VRRC, 2002.

LI, K. et al. Optimal surface segmentation in volumetric images-a graph-theoretic approach. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, v. 28, n. 1, p. 119–134, 2006.

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 '87, ACM*, ACM-0-89791-227-6/87/007/01, p. 163–169, 1987.

LOUNSBERY, M.; DEROSE, T.; WARREN, J. Multiresolution analysis for surfaces of arbitrary topological type. *ACM Transactions on Graphics (TOG)*, v. 16, n. 1, p. 34–73, 1997.

MADSEN, R. et al. *3D Scanning and Reconstruction of Large Scale Environments*. [S.l.], 2011.

MENCL, R. A graph–based approach to surface reconstruction. In: *Computer Graphics Forum*. [S.l.: s.n.], 1995. v. 14, p. 445–456.

MENCL, R.; MULLER, H. Interpolation and approximation of surfaces from three-dimensional scattered data points. In: *Dagstuhl '97, Scientific Visualization*. Washington, DC, USA: IEEE Computer Society, 1999. p. 223–232. ISBN 0-7695-0505-8. Disponível em: <http://dl.acm.org/citation.cfm?id=647367.723590>.

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.

MULLER, H. Surface reconstruction - an introduction. In: *Scientific Visualization Conference, 1997*. [S.l.: s.n.], 1997. p. 239.

NAVARRO, S. F. *3D reconstruction of object shape using structured light*. Tese (Doutorado) — University of Girona, Department of Computer Architecture and Technology, 2009.

NEWMAN, T. S.; YI, H. A survey of the marching cubes algorithm. *Computers & Graphics*, v. 30, n. 5, p. 854 – 879, 2006. ISSN 0097-8493.

ONO, J. H. P. et al. 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>.

PANASONIC. *D-IMager 3D sensing technology*. 2012. Disponível em: <http://www2.panasonic.biz/es/densetsu/device/3DImageSensor/en/index.html>.

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.

PROJECT, T. C. *CGAL User and Reference Manual*. 4.0. ed. [S.l.]: CGAL Editorial Board, 2012. Http\string://www.cgal.org/Manual/4.0/doc_html/cgal_manual/packages.html.

ROTH, G.; WIBOWOO, E. An efficient volumetric method for building closed triangular meshes from 3-d image and point data. In: *Graphics Interface*. [S.l.: s.n.], 1997. v. 97, p. 173–180.

RUPRECHT, D.; NAGEL, R.; MULLER, H. Spatial free-form deformation with scattered data interpolation methods. *Computers & graphics*, v. 19, n. 1, p. 63–71, 1995.

RUSU, R. B. *Semantic 3D Object Maps for Everyday Manipulation in Human Living Environments*. Tese (Doutorado) — Computer Science department, Technische Universitaet Muenchen, Germany, out. 2009.

RUSU, R. B.; COUSINS, S. 3d is here: Point cloud library (pcl). In: *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. [S.l.: s.n.], 2011. p. 1–4.

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>.

SHENE, C.-K. *Mesh Basics*. Michigan Technological University: [s.n.], 2010. Disponível em: <http://www.cs.mtu.edu/ shene/COURSES/cs3621/SLIDES/Mesh.pdf>.

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/>.

SZELISKI, R. Rapid octree construction from image sequences. *CVGIP Image Understanding*, v. 58, p. 23–23, 1993.

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>.

Washington. *RGB-D Object Dataset*. University of Washington, 2012. Disponível em: <http://www.cs.washington.edu/rgbd-dataset/>.

WEBB, J.; ASHLEY, J. *Beginning Kinect Programming With the Microsoft Kinect Sdk*. [S.l.]: Apress, 2012.

XU, G. Y.; CHEN, L. P.; GAO, F. Study on binocular stereo camera calibration method. In: *Image Analysis and Signal Processing (IASP), 2011 International Conference on*. [S.l.: s.n.], 2011. p. 133 –137.

APÊNDICE A - ARTIGO APRESENTADO NO CONGRESSO SIBGRAPI 2012

ONO, J. H. P., SEMENTILLE, A. C., CALDEIRA, M. A. C., MARAR, J. F. *Poisson Surface Reconstruction with Local Mesh Simplification.* Workshop of Works in Progress (WIP) in SIBGRAPI 2012 (XXV Conference on Graphics, Patterns and Images). In: SIBGRAPI. Ouro Preto, MG, Brasil: 2012.

Poisson Surface Reconstruction with Local Mesh Simplification

Jorge Henrique Piazentin Ono, Antônio Carlos Sementille, Marco Antônio Corbucci Caldeira, João Fernando Marar
Department of Computing, School of Sciences
Universidade Estadual Paulista - UNESP
Bauru, Brasil

***Abstract*—Surface reconstruction techniques have applications in many fields, such as cartography, augmented reality and reverse engineering. In real time systems, meshes in multiple resolutions can be used to ensure fast processing, where more detailed representations are used only when necessary. This paper proposes and validates the simplification of multiresolution meshes generated with Poisson Surface Reconstruction. We achieved simplifications of almost 90% of the reconstructed model maintaining it's average form.**

***Keywords*-Surface reconstruction; multiresolution; meshing; geometric modelling;**

I. Introduction

Virtual representations of objects can be created in CAD (Computer Aided Design) tools or generated from existent physical models, through surface reconstruction techniques. The goal of surface reconstruction is to find a surface from a finite set of sampled geometric values (usually points in space) [1]. The generated surface can be, then, discretized, producing a triangle mesh.

The simpler the mesh, the easier it is to store and process it. Multiresolution methods can be used to generate meshes in different level of detail (LOD) and, to ensure fast processing, an object which is far from the observer can be rendered in a low LOD, while higher LOD could be used for closer objects [2], [3]. Simplified meshes allow not only improvements in rendering speed, but also fast transmission of 3D models in network-based applications [4].

Contributions: This paper proposes an improvement to the Poisson surface reconstruction algorithm [5], by adding mesh simplification to the multiresolution result. Two approaches are studied: edge collapse operations [6] and a solution inspired by the work of Mingyi et al. [7], which defines a local normal change threshold to reconstruct a surface from the point set.

A. Related work

Surface reconstruction algorithms from scattered data points can be divided in 4 groups: distance functions, spacial subdivision, spatial free form warping and incremental surface-oriented construction [1]. Distance functions calculate the distance D between any point in space and the surface, therefore, when $D \approx 0$, the surface is approximated by an implicit function. Spatial subdivision methods decompose a set of points P in cells, selecting only the cells related to the form described by P. Spatial warping techniques deform an initial surface in order to approximate it to the point set. Incremental construction build an approximating/interpolating surface directly from properties of the data points [1].

The Poisson surface reconstruction [5] is an approach that expresses surface reconstruction as the solution to a Poisson equation. This method uses an implicit solution to approximate the surface and then extract the isosurface using an adaptation of the Marching Cubes [8] algorithm. Since accurate solutions are only needed near the surface, an octree structure is applied to approximate the raw data and, by choosing the maximum octree depth, one can create surfaces of various LOD. Fig. 1 shows our experiments with the Stanford bunny, which is reconstructed at 3 different octree depths. Table I shows the reduction of polygons as the maximum octree depth is limited.

Many methods can be used to simplify a mesh, for example, clustering algorithms, iterative simplification and particle simulation [9]. Mingyi et al. [7] evaluate the neighbourhood of every point and, if the local normal change, defined as the maximum angle between the points' normal, is less than a established threshold, these points are simplified by one plane. Edge collapse operations consist of iteratively replacing an edge with a single vertex by removing 2 triangles per collapse, but maintaining the overall look of the surface [10].

(a) Depth 5 (b) Depth 7 (c) Depth 10

Fig. 1. Stanford Bunny reconstructed with Poisson method at 3 different resolutions

II. Mesh simplification

Our technique aims at reducing the number of polygons produced by the Poisson method, by adding a surface simplification step to the reconstruction process. Two approaches are studied:

TABLE I
NUMBER OF TRIANGLES IN THE RECONSTRUCTED MODEL FOR DIFFERENT DEPTHS USING POISSON METHOD

Octree Depth	Number of vertex	Number of faces
5	2160	4316
7	36322	72640
10	682031	1364058

- Post-processing of the mesh, simplifying it with the edge collapse algorithm, well defined by [6], [11].
- Evaluation of the triangle mesh (local normal change of the faces), delimiting a threshold for coplanarity. This method is originally described by [7] to work on point sets, but can be extended for polygon meshes.

III. IMPLEMENTATION

In order to implement this multiresolution approach, a surface reconstruction pipeline was developed, consisting of data (point cloud) acquisition with Microsoft Kinect, point cloud filtering and segmentation, surface reconstruction and surface simplification.

The Point Cloud Library (PCL) [12] was used to capture and pre-process the raw data. Then we adapted the algorithm of [5] to analyse and reconstruct the surface. Finally, we adjusted the CGAL [10] implementation of the edge collapse algorithm to simplify our triangle mesh.

IV. WORK IN PROGRESS

Currently, we are working on mesh segmentation of our point sets (depth maps) using the PCL [12] version of RANSAC (Random Sample Consensus) [13] and we are developing the local normal change approach for triangle mesh simplification.

V. EXPERIMENTS

We applied our algorithm in public domain point sets (the Stanford 3D Scanning Repository and the CGAL Surface Reconstruction Examples). Fig. 2 shows a dragon reconstructed with Poisson method with octree at depth 8 and it's simplified version with Edge Collapse operation. Table II shows that even though the simplification process reduced the number of polygons to a tenth of the original model, the average form of the object is maintained.

TABLE II
SIMPLIFICATION RESULTS USING EDGE COLLAPSE ALGORITHM

	Number of vertex	Number of faces
Original Dragon	10000	19994
Simplified Dragon	1001	1996

VI. CONCLUSION AND FUTURE WORK

In this paper, we proposed and validated the simplification of multi-resolution objects created with Poisson surface reconstruction. As we finish the implementation of our prototype, we expect to achieve better multi-resolution meshes (less polygons and more detail) with the combination of these techniques than with the Poisson method itself. In order to validate our experiments, we will use Metro [14] to compare the differences between the original and the simplified surface.

(a) Original dragon (b) Simplified dragon

Fig. 2. Edge Collapse Simplification

ACKNOWLEDGMENT

The authors would like to thank the use of the laboratory SACI (Adaptive Systems and Intelligent Computing), Department of Computing, School of Sciences, UNESP Bauru.

REFERENCES

[1] H. Muller, "Surface reconstruction - an introduction," in *Scientific Visualization Conference, 1997*, Jun. 1997, p. 239.

[2] T. Gerstner, "Multiresolution extraction and rendering of transparent isosurfaces," *Computers & Graphics*, vol. 26, no. 2, p. 219–228, 2002.

[3] M. Botsch and L. Kobbelt, "A robust procedure to eliminate degenerate faces from triangle meshes," in *Proc. of Vision, Modeling, and Visualization*, vol. 1, 2001, p. 283–289.

[4] P. Cignoni, C. Montani, and R. Scopigno, "A comparison of mesh simplification algorithms," *Computers & Graphics*, vol. 22, no. 1, p. 37–54, 1998.

[5] M. Kazhdan, M. Bolitho, and H. Hoppe, "Poisson surface reconstruction," in *Proceedings of the fourth Eurographics symposium on Geometry processing*, 2006, p. 61–70.

[6] P. Lindstrom and G. Turk, "Fast and memory efficient polygonal simplification," in *Visualization'98. Proceedings*, 1998, p. 279–286.

[7] H. Mingyi, X. Bangshu, and Y. Huajing, "Multi-resolution surface reconstruction," in *Image Processing, 2004. ICIP'04. 2004 International Conference on*, vol. 3, 2004, p. 1971–1974.

[8] H. E. C. William E. Lorensen, "Marching cubes: A high resolution 3d surface construction algorithm," *SIGGRAPH '87, ACM*, vol. ACM-0-89791-227-6/87/007/01, pp. 163–169, 1987.

[9] M. Pauly, M. Gross, and L. Kobbelt, "Efficient simplification of point-sampled surfaces," in *Visualization, 2002. VIS 2002. IEEE*, 2002, p. 163–170.

[10] T. C. Project, *CGAL User and Reference Manual*, 4th ed. CGAL Editorial Board, 2012, http\string://www.cgal.org/Manual/4.0/doc_html/cgal_manual/packages.html.

[11] H. Hoppe, "Progressive meshes," in *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, 1996, p. 99–108.

[12] S. Rusu, R.B. ,Cousins, "3d is here: Point cloud library (pcl)," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, 2011, p. 1–4.

[13] M. Fischler and R. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, p. 381–395, 1981.

[14] P. Cignoni, C. Rocchini, and R. Scopigno, "Metro: measuring error on simplified surfaces," in *Computer Graphics Forum*, vol. 17, 1998, p. 167–174.

APÊNDICE B - ARTIGO APRESENTADO NO CONGRESSO DE INICIAÇÃO CIENTÍFICA - UNESP 2012

ONO, J. H. P., SEMENTILLE, *Reconstrução de Superfícies com Método de Poisson e Simplificação Local de Malhas.* XXIV Congresso de Iniciação Científica da Unesp. Bauru, SP, Brasil: 2012.

Reconstrução de Superfícies com Método de Poisson e Simplificação Local de Malhas

Jorge Henrique Piazentin Ono, Antônio Carlos Sementille. Campus Bauru, Faculdade de Ciências, Bacharelado em Ciência da Computação, jorgehpo@gmail.com

Palavras Chave: *Reconstrução de Superfícies, Modelagem Geométrica.*

Introdução

Representações virtuais de objetos podem ser criadas em ferramentas CAD (*Computer Aided Design*) ou geradas a partir de um modelo físico existente. O objetivo da reconstrução de superfícies é encontrar uma superfície em um conjunto finito de valores geométricos, produzindo uma malha poligonal [1].

Quanto mais simples a malha, mais fácil é processá-la e armazená-la. Este trabalho propõe uma alteração ao algoritmo de reconstrução de Poisson [2], aplicando métodos de simplificação de malhas, que diminuem a complexidade do objeto resultante. Duas abordagens são estudadas: operações de colisão de arestas [3] e uma solução inspirada no trabalho de Mingyi et al. [4], que define um limiar para coplanaridade durante a reconstrução.

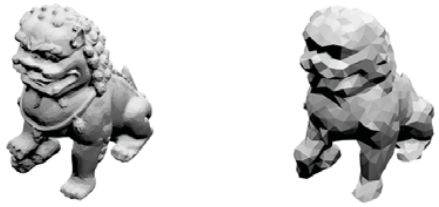
Material e Métodos

Um *pipeline* completo de reconstrução 3D foi desenvolvido, composto por aquisição da nuvem de pontos com o *Microsoft Kinect* (sensor de profundidade), filtragem dos dados, reconstrução e simplificação de superfícies. Primeiramente, configurou-se o ambiente experimental com o sensor Kinect e um computador pessoal (sistema operacional *Windows 7*, ambiente de desenvolvimento *Microsoft Visual Studio* 2010 e a biblioteca *Point Cloud Library*, utilizada para capturar e pré-processar os dados do *Kinect*).

Em seguida, o algoritmo de Kazhdan et al. (2006) foi adaptado para analisar e reconstruir a superfície. Finalmente, a implementação do algoritmo de simplificação de malhas *Edge Collapse* da biblioteca de processamento Gráfico CGAL (*Computational Geometry Algorithms Library*) foi ajustada para simplificar nossa malha poligonal. Atualmente o trabalho encontra-se na implementação da segunda abordagem de simplificação.

Resultados e Discussão

O algoritmo foi aplicado em dados experimentais e de domínio público (Repositório 3D de Stanford [5] e da biblioteca CGAL [6]). A Figura 1 mostra um dragão reconstruído com o método de Poisson e seu modelo simplificado com *Edge Collapse*.



(a) Dragão Original (b) Dragão Simplificado

Figura 1. Simplificação *Edge Collapse* aplicada ao modelo "*Chinese Dragon*" [6].

A quantidade de polígonos e vértices desses modelos é apresentada na Tabela 1. Embora o processo de simplificação tenha reduzido o número de faces (triângulos) em aproximadamente 90%, a forma geral do objeto foi mantida.

Tabela 1. Resultados da simplificação da malha.

	Nº de Vértices	Nº de faces
Dragão Original	10000	19994
Dragão Simplificado	1001	1996

Conclusões

Neste trabalho, propôs-se a simplificação de superfícies durante o processo de reconstrução. Constatou-se que, mesmo reduzindo o número de polígonos, pode-se manter a topologia original do objeto. Construído o protótipo, espera-se atingir malhas melhores, com menos polígonos e mais detalhes, em comparação com as reconstruções atingidas apenas com o método de Poisson.

Agradecimentos

Agradecemos a utilização do laboratório SACI/DCo /FC/UNESP para a realização dos experimentos.

[1] H. Muller, "Surface Reconstruction - An Introduction", Scientific Visualization Conference, 1997, 1997, p. 239.

[2] M. Kazhdan, M. Bolitho, and H. Hoppe, "Poisson surface reconstruction", Proceedings of the fourth Eurographics symposium on Geometry processing, 2006, pp. 61–70.

[3] P. Lindstrom and G. Turk, "Fast and memory efficient polygonal simplification" IEEE Visualization, pp. 279–286, 1998.

[4] H. Mingyi, X. Bangshu, and Y. Huajing, "Multi-resolution surface reconstruction", Image Processing, 2004. ICIP'04. 2004 International Conference on, 2004, vol. 3, pp. 1971–1974.

[5] The Stanford 3D Scanning Repository, disponível em: http://graphics.stanford.edu/data/3Dscanrep/ , 2012.

[6] CGAL Tutorials, disponível em : http://cgal.org/tutorials, 2012.