INSTITUTO NACIONAL DE MATEMÁTICA PURA E APLICADA

HALLISON OLIVEIRA DA PAZ

RECONSTRUÇÃO ADAPTATIVA DE SUPERFÍCIES IMPLÍCITAS A PARTIR DE IMAGENS DE PROFUNDIDADE

RIO DE JANEIRO

2017

INSTITUTO NACIONAL DE MATEMÁTICA PURA E APLICADA

HALLISON OLIVEIRA DA PAZ

RECONSTRUÇÃO ADAPTATIVA DE SUPERFÍCIES IMPLÍCITAS A PARTIR DE IMAGENS DE PROFUNDIDADE

Dissertação de Mestrado apresentada ao Instituto Nacional de Matemática Pura e Aplicada, como requisito parcial para obtenção do grau de Mestre em Matemática, opção Computação Gráfica.

Orientador: Luiz Carlos Pacheco Rodrigues Velho – Ph.D.

Rio de Janeiro

2017

INSTITUTO NACIONAL DE MATEMÁTICA PURA E APLICADA

HALLISON OLIVEIRA DA PAZ

RECONSTRUÇÃO ADAPTATIVA DE SUPERFÍCIES IMPLÍCITAS A PARTIR DE IMAGENS DE PROFUNDIDADE

Dissertação de mestrado apresentada ao Instituto Nacional de Matemática Pura e Aplicada.

Orientador: Luiz Carlos Pacheco Rodrigues Velho

Aprovado em 5 de abril de 2017 pela seguinte Banca Examinadora:

Luiz Carlos Pacheco Rodrigues Velho – Orientador, IMPA

Luiz Henrique de Figueiredo, IMPA

Diego Fernandes Nehab, IMPA

Anselmo Antunes Montenegro, UFF

RIO DE JANEIRO

2017

Dedico este trabalho aos meus pais, que com muita atenção, Amor e empenho me educaram e me incentivaram a ter disciplina para alcançar os meus sonhos

AGRADECIMENTOS

Agradeço ao professor Luiz Velho pela disponibilidade e prontidão para aconselhar, orientar e debater ideias, sempre com muito entusiasmo e espírito de colaboração. Agradeço-lhe também pela confiança depositada em mim e a liberdade de atuação proporcionada durante o trabalho.

À minha companheira de todas as horas pelo suporte e incentivo, por vezes me fazendo enxergar soluções onde não as via.

Ao Djalma, pela ideias apresentadas e pela ajuda com a realização dos experimentos deste trabalho.

Aos meus companheiros do laboratório Visgraf, pelas sugestões de novas hipóteses e abordagens, sempre que necessário. Mesmo àqueles que não tiveram um envolvimento técnico com o trabalho, agradeço pelas boas conversas e momentos de descontração, que também são relevantes para a condução sadia das atividades.

SUMÁRIO

LISTA DE ILUSTRAÇÕES

RESUMO

Este trabalho constitui um estudo sobre o problema de reconstrução de superfícies implícitas a partir de dados de imagens de profundidade, levando-se em consideração o cenário atual de dispositivos móveis. Propõe-se um método que possibilite a obtenção de modelos 3D com uma geometria adaptada à resolução da superfície representada em oposição à resolução da imagem capturada. Aborda-se a aquisição de imagens de profundidade em um dispositivo móvel a partir de pontos de vista distintos, o registro dos dados em relação a um referencial no espaço euclidiano, bem como a fusão desses dados por meio da construção de uma representação extrínseca da superfície, possibilitando a reconstrução incremental do objeto ou cena capturados. Neste trabalho, os experimentos e cálculos foram realizados em um computador convencional, mas as escolhas de soluções e algoritmos têm por objetivo testar a viabilidade de um método cujo pipeline completo possa ser executado em um dispositivo móvel. Exibimos as vantagens obtidas com o uso dos recursos disponíveis em um dispositivo móvel atual na concepção de soluções para problemas clássicos da área, ao mesmo tempo que exploramos métodos computacionalmente eficientes para operar segundo a limitação de hardware que estas plataformas possuem em relação a um computador comum.

ABSTRACT

This work constitutes a study about the problem of implicit surfaces reconstruction from depth images data, taking into account the current scenario of mobile devices. We propose a method for obtaining 3D models with a geometry adapted to the resolution of the represented surface as opposed to the resolution of the captured image. We address the acquisition of depth images in a mobile device from different points of view, the data registration in relation to a frame of reference in the Euclidean space, as well as the fusion of these data through the construction of an extrinsic representation of the surface which allows the reconstruction of the captured object or scene incrementally. In this work, the experiments and computation were performed on a regular computer, but the choices of solutions and algorithms are designed to test the viability of a method whose complete pipeline can run on a mobile device. We show the advantages from using the features available on a current mobile device in designing solutions to classic problems of the area. On the other hand, we also explore computationally efficient methods to operate according to the hardware limitation these platforms have over an ordinary computer.

1 INTRODUÇÃO

1.1 DEFINIÇÃO DO PROBLEMA

Este trabalho aborda o problema de reconstrução de modelos geométricos a partir de imagens capturadas por câmeras com sensores de profundidade. Objetiva-se reconstruir a topologia e a geometria de superfícies que representam objetos e cenas observáveis no mundo real, a partir de dados de profundidade adquiridos por câmeras posicionadas em pontos de vista diferentes. Partindo-se de dados brutos como as imagens de profundidade e de informações de registro entre as câmeras de aquisição, pretende-se gerar uma representação em malha poligonal triangular em que a densidade de polígonos seja adaptada à geometria dos entes observados.

1.2 IMPORTÂNCIA DO PROBLEMA

A área de sensoriamento de dados geométricos é uma área de pesquisa bastante ativa com aplicações que se diversificam cada vez mais à medida que novas tecnologias e equipamentos são introduzidos na academia e na indústria. Há quase duas décadas atrás, foi conduzido um grande projeto de digitalização das obras de Michelangelo e de diversas outras obras artísticas da Roma antiga. Nesse trabalho, conhecido como *The Digital Michelangelo Project* [1], foi realizada a aquisição de um grande volume de dados geométricos possibilitando a reconstrução de modelos virtuais de algumas obras de museus.

Embora a tecnologia da época permitisse a aquisição desses modelos em um ambiente controlado, também foi necessário o desenvolvimento de métodos computacionais para poder manipular e processar essa grande quantidade de dados, que podia chegar à ordem de bilhões de polígonos. Desde então, houve diversos estudos e contribuições nas áreas de simplificação de superfícies, construção de estruturas em multiresolução, remoção de ruído de aquisição, registro em relação a um referencial, reconstrução geométrica, dentre outras. Dados dessa natureza têm sido utilizados no estudo de uma série de problemas e na criação de diversas aplicações como, por exemplo, construção de maquetes virtuais, estudos topográficos e modelagem de objetos para renderização ou impressão 3D.

Recentemente, a aquisição de dados geométricos para aplicações de visão computacional tem sido amplamente explorada, seja para melhorar a solução de problemas clássicos da área tais como segmentação, reconstrução de superfícies, reconhecimento de padrões ou estimação de movimento, seja para propor novas soluções que não eram possíveis sem esses dados e/ou por limitações tecnológicas em anos anteriores. As recentes pesquisas na área de robótica e aprendizado de máquina, por exemplo, têm utilizado dados geométricos para estimação de rotas de robôs [2, 3] e compreensão de cenas [4, 5]. Novas soluções para problemas como reconhecimento de padrões possibilitaram novas abordagens para

a criação de interfaces entre homem e máquina, principalmente pela melhora na precisão de captura e reconhecimento de gestos.

1.3 MOTIVAÇÃO

Durante anos, o processo de aquisição de dados geométricos ficou restrito à academia e a alguns setores da indústria devido ao custo dos equipamentos existentes e à necessidade de operá-los em ambientes controlados, requerindo até mesmo conhecimentos técnicos específicos. Em [1], os pesquisadores descrevem a estrutura utilizada para adquirir os dados para o projeto, bem como algumas arquiteturas utilizadas em trabalhos feitos por outros grupos antes deles. Pode-se observar que o aparato utilizado era grande, pouco portátil e, dependendo da aplicação, era necessário usar várias tecnologias diferentes, tais como sensores por luz estruturada e por *time of flight*.

A inserção da primeira versão do *Kinect*, um aparelho comercial para uso doméstico capaz de capturar imagens com profundidade, tornou este processo de aquisição mais barato e mais difundido. Esse fato também possibilitou o surgimento de novas aplicações, aproximando dessa área hobbystas interessados em explorar as capacidades desse sensor, além dos próprios consumidores leigos em conhecimento técnico.

Evidentemente, a precisão e a qualidade dos dados adquiridos por um sensor de baixo custo como o Kinect são bastante inferiores às dos sensores caros e robustos que vinham sendo utilizados. Esta conjuntura trouxe a necessidade de adaptação dos problemas existentes e a preocupação com novos problemas, tais como agravamento no ruído de aquisição, falta de informações, baixa resolução e, no sentido das aplicações práticas, solução de problemas em tempo real. Dessa forma, problemas clássicos como filtragem de ruído, reconstrução de superfícies, reconhecimento de padrões etc, mantém-se bastante ativos devido aos desafios de se encontrar soluções eficazes, robustas e eficientes para operar em diversos tipos de cenários, inclusive interação em tempo real com usuários do sistema.

1.4 CONTEXTO

Há aproximadamente uma década, a popularização de smartphones e tablets, aliada ao surgimento do mercado de aplicativos móveis, que facilitou o desenvolvimento e a publicação de *softwares* capazes de auxiliar as pessoas a resolverem tarefas cotidianas com seus celulares, ampliou muito a oferta de aplicações de consumo diversas e mudou a forma como as pessoas se relacionam com estes dispositivos. O contexto em que este trabalho se insere conecta-se com a existência destes dispositivos móveis, em particular smartphones e tablets capazes de adquirir imagens RGB-D (com cor e profundidade).

Atualmente, acessórios como o *Structure Sensor* [6] possibilitam a aquisição de dados geométricos de maneira móvel, acoplando-se uma câmera de profundidade a um *iPad*. Por outro lado, a plataforma *Tango* [7], desenvolvida pela *Google* almeja disponibilizar

no mercado smartphones com câmeras de profundidade já integradas. Ou seja, verifica-se um movimento no sentido de tornar esses sensores ainda mais populares. Dessa forma, se faz necessário o desenvolvimento de métodos e soluções que possibilitem a exploração destes dados em aplicações de consumo, tais como a modelagem de ambientes de forma não controlada ou experiências de realidade aumentada e realidade mista. O termo "não controlada" especifica a situação em que o usuário faz um passeio com seu dispositivo pela cena para adquirir dados de forma intuitiva e natural, sem preparação do ambiente ou planejamento do trajeto. Já os termos realidade aumentada ou mista referem-se às aplicações que mesclam dados visuais no ambiente real com objetos virtuais, que podem ter sido adquiridos previamente do mundo real, inclusive com esses próprios sensores. Em [8], há uma introdução às características das diferentes plataformas móveis para captura de imagens RGB-D.

Assim como o surgimento do primeiro Kinect trouxe novas variáveis ao cenário de sensoriamento geométrico e enriqueceu os problemas e soluções existentes na área, existe uma série de resultados e técnicas de geometria computacional, modelagem geométrica e visão computacional que foram aplicadas no contexto de reconstrução geométrica a partir de dados de imagens de profundidade de uma forma clássica e, atualmente, com este novo cenário de dispositivos móveis, torna-se possível operar dentro de um novo paradigma. Nestas circunstâncias, podemos aproveitar as vantagens destes dispositivos, tais como a existência de diversos sensores de atitude (acelerômetro, giroscópio, magnetômetro), multiplas câmeras, acesso à internet e outras formas de conexão sem fio como o padrão *bluetooth,* e minimizar os efeitos das desvantagens de se operar em um sistema com restrição de recursos, levando em consideração a maneira peculiar com que os usuários o utilizam.

Resultados clássicos geralmente não levam em conta a existência de sensores que possam ajudar a resolver problemas ou melhorar a solução dos problemas já existentes a partir de seus novos dados. Dessa forma, os sistemas e soluções computacionais elaborados acabam por se tornar mais complexos e mais computacionalmente intensivos pela tentativa de resolver um problema com menos informações. Ou seja, os sensores de um dispositivo móvel nos permitiria, por exemplo, assumir que temos um bom mecanismo de *tracking* e, então, construir uma solução que não precise calcular transformações entre câmeras de referenciais diferentes para registrar os dados.

Muitos resultados clássicos também trabalham com grades de amostragem uniforme, tendo a necessidade de armazenar uma enorme quantidade de dados, consequentemente, onerando a memória enquanto recurso computacional. Nesse sentido, métodos adaptativos, isto é, que produzam modelos adaptados à geometria da cena/objeto constituem soluções menos caras e, portanto, mais adequadas a um dispositivo móvel.

As tecnologias de comunicação sem fio e a possibilidade de conexão à rede mundial de

computadores a partir de um *smartphone* ou *tablet* também viabilizam a exploração de técnicas que levem em consideração a existência de um servidor operando em conjunto com um dispositivo cliente, seja recebendo e armazenando dados (geométricos, visuais, computacionais, etc), seja refinando computacionalmente soluções. Este é um fato bastante propício para o projeto de aplicações com interações em tempo real, pois é possível, por exemplo, apresentar ao usuário imagens em resoluções menores, enquanto o servidor finaliza cálculos mais pesados em uma resolução maior e transmite o resultado para o cliente. No passado, também por restrições de *hardware*, vários métodos out-of-core foram desenvolvidos para diferentes propósitos tais como aquisição, simplificação ou tratamento de dados geométricos. Este cenário de computação em nuvem permite a exploração de ainda mais recursos nas soluções, gerindo não só o armazenamento como também a computação em si.

1.5 CONTRIBUIÇÕES

As contribuições deste trabalho se concentram na extensão e adaptação de métodos clássicos para a construção de métodos que operem dentro desse novo paradigma de computação móvel integrada a sensores e que leve em consideração questões como a forma que esses dispositivos são utilizados e como esses métodos poderiam se integrar em aplicações para esses usuários.

Neste trabalho, os experimentos foram conduzidos com aquisição de dados por um Kinect e realização de cálculos em um computador convencional. Contudo cada etapa foi pensada de modo a constituir um teste de viabilidade do método de reconstrução apresentado, a fim que ele possa ser implementado e executado completamente em um dispositivo móvel. Propõe-se um *pipeline* que contemple a aquisição de imagens de profundidade, utilize dados de calibração obtidos externamente, construa uma representação extrínseca da cena/objeto adquirido, a partir de uma formulação implícita da superfície de interesse, e extraia uma representação intrínseca com uma resolução adaptada à geometria da cena/objeto ao final. Por hipótese, a calibração das câmeras seria obtida pelos sensores de um dispositivo móvel.

As contribuições principais estão em como partir de dados brutos como as imagens de profundidade e informações de calibração das câmeras e gerar uma representação extrínseca da região da cena/objeto adquirido; como lidar com um cenário em que os dados são adquiridos incrementalmente, isto é, uma imagem de cada vez, possibilitando a atualização desta representação extrínseca, juntando as informações em um modelo único; como preservar a topologia da superfície e calcular uma geometria reconstruída que preserve a forma do ente adquirido e; como construir essa representação de maneira semi-adaptada à geometria, economizando recursos e possibilitando a extração de um modelo intrínseco a cada etapa de final de ciclo do pipeline.

2 TRABALHOS RELACIONADOS

Para a concretização deste trabalho foi necessário o estudo da teoria e das técnicas apresentadas em trabalhos sobre imagens RGB-D, simplificação de malhas poligonais, reconstrução de dados implícitos, reconstrução geométrica a partir de dados de imagens de profundidade, poligonização de superfícies e poligonização adaptativa. Nesta seção apresentaremos alguns trabalhos relacionados para a compreensão dos termos e métodos que serão utilizados ou adaptados para o nosso propósito.

2.1 TRABALHOS GERAIS SOBRE RGB-D

Recentemente, imagens RGB-D têm sido utilizadas para trabalhos de naturezas diversas tais como segmentação semântica, reconhecimento e classificação de objetos, compreensão de cenas e reconstrução geométrica. O avanço na facilidade de se obter dados desta natureza e os esforços de pesquisa enveredados na área contribuíram com o aumento da quantidade de base de dados disponibilizadas para fins de pesquisa. Algumas dessas bases, como as [9, 10, 11], foram catalogadas por Firman no endereço em [12], classificadas pela finalidade da pesquisa em que foram produzidas. Para testes de algoritmos em condições menos adversas, há também bases de imagens RGB-D sintéticas como a do Vladen [13], geradas por programas de modelagem 3D.

Ao lidar com imagens RGB-D adquiridas do mundo real, temos que lidar com o ruído de aquisição, que pode variar com a qualidade do sensor e o nível de controle e preparação na captura. O ruído pode ser tratado no domínio dos pixels da imagem com filtros 2D específicos, como por exemplo o filtro bilateral, ou no domínio espacial da nuvem de pontos 3D. Com informações topológicas, de conectividade dos pontos, podem ser aplicados métodos voltados para redução de ruídos em malhas [14].

2.2 RECONSTRUÇÃO GEOMÉTRICA COM IMAGENS DE PROFUNDIDADE

Especificamente na área de reconstrução, temos como referência um trabalho relativamente recente e bastante conhecido por suas contribuições, o *Kinect Fusion* [15]. O Kinect Fusion é um projeto que utiliza os dados de profundidade adquiridos a partir de um Kinect de primeira geração para construir um modelo geométrico da cena capturada. O algoritmo realiza a fusão de dados de vários quadros capturados com um Kinect em movimento e gera um modelo virtual que pode ser visualizado em tempo real. Embora algumas das aplicações apresentadas como exemplos neste trabalho sejam similares ao que gostaríamos de obter, podemos destacar duas diferenças em termos de conceitos entre o Kinect Fusion e o nosso trabalho, ambas relacionadas ao fato de um cenário utilizar um sensor estático, sem dados adicionais, e o outro supor o uso de recursos de um dispositivo móvel.

A primeira diferença é que, por operar em um cenário com menos informações, o Kinect Fusion precisou criar um método que fosse capaz de computar dados de *tracking* para alinhar os referenciais dos quadros capturados. Estes cálculos eram feitos com base nos mapas de profundidade e na reconstrução parcial da geometria do modelo. Essa é uma importante contribuição do projeto, citada tanto nas publicações iniciais [15, 16] quanto em duas outras publicações sobre métodos de relocalização de câmeras [17, 18]. Em nosso trabalho, queremos verificar a viabilidade de se utilizar os sensores de um dispositivo móvel para obter transformações que nos permitam registrar os dados geométricos em um mesmo referencial, sem a necessidade de aplicar métodos sofisticados e custosos para esse fim. Sabemos que apenas com os sensores de atitude de um dispositivo móvel, como acelerômetro, giroscópio e magnetômetro, não é possível fazer este alinhamento de forma precisa, pois não se tem informações quanto à translação do dispositivo no espaço, apenas rotação e orientação. No entanto, dispositivos que utilizam o Structure Sensor ou a plataforma Tango são capazes de realizar um tracking preciso com seis graus de liberdade em áreas de pequenas dimensões, a partir da integração dos dados dos sensores de atitude, câmeras de cor e sensor de profundidade. Para grandes distâncias, uma abordagem baseada nesses sensores pode ter que lidar com o problema de *drift*, situação em que pequenos erros de medição se acumulam ao longo do tempo levando a uma situação de erro maior.

A segunda diferença que podemos destacar é que dentro deste cenário de computação fixa, o método apresentado utilizou-se de abordagens clássicas de reconstrução uniforme, trabalhando com volumes bastante refinados e produzindo malhas densas. Os dados geométricos obtidos de sensores RGB-D como o Kinect estão na mesma resolução ou em uma resolução próxima à da imagem, o que provê bastante redundância. Apesar de ser computacionalmente intensivo e requerer uma quantidade grande de memória, o método apresentado tinha soluções bastante eficientes que, utilizando o poder computacional e o paralelismo de placas gráficas, possibilitaram a execução do programa em tempo real. Queremos propor um método que tenha um consumo de memória reduzido para um dispositivo móvel, buscando fazer com que a resolução geométrica seja adaptada à precisão do objeto e não à da imagem.

2.3 SIMPLIFICAÇÃO DE MALHAS POLIGONAIS

O problema de simplificação de malhas poligonais é um problema de otimização em que se busca obter uma aproximação para uma determinada superfície segundo alguns critérios. Dois critérios comumente utilizados na definição do problema são o tamanho da malha e o erro geométrico obtido. No primeiro caso, dado um valor N, procura-se calcular uma malha de tamanho N que seja a melhor aproximação geométrica da superfície representada pela malha original. O valor N pode referir-se à quantidade de vértices ou faces da malha final, por exemplo. No segundo caso, dado ε, objetiva-se a

menor malha possível que seja uma aproximação geométrica da superfície dentro de uma tolerância de erro ε. A escolha dos critérios geralmente depende da aplicação.

Com a evolução das tecnologias para aquisição de dados geométricos, o tamanho dos modelos geométricos adquiridos cresceu muito, dificultando ou impossibilitando o uso direto desses dados, pois não era possível operar com eles na memória principal do computador. Dessa forma, para se realizar uma operação de visualização, por exemplo, poderia ser necessário reduzir a quantidade de dados, tentando comprometer ao mínimo a qualidade e detalhes do modelo. Este problema trouxe bastante atividade para a área de simplificação de malhas, resultando no desenvolvimento de diversos métodos e soluções tanto para operações *in-core* quanto para *out-of-core*.

Antes de adentramos nos detalhes dos métodos, contudo, levantaremos duas reflexões. Primeiramente, deve-se ter ciência que esse problema já foi provado ser *NP-completo* [19] e, portanto, as soluções computacionais propostas baseiam-se em heurísticas e hipóteses de simplificação que possibilitem o cálculo de soluções sub-ótimas, aproximações razoáveis para o problema. Uma etapa importante, portanto, é a definição de boas métricas para medir o erro cometido e computar funções objetivo a serem minimizadas ou maximizadas conforme o caso. De modo geral, o que temos como entrada em problemas de simplificação é uma malha poligonal, um conjunto de elementos (pontos, arestas, faces) que podemos considerar imersos em um espaço Euclideano de dimensão 3. Supondo que esta malha de entrada representa uma aproximação de uma superfície de interesse, queremos produzir como saída uma malha poligonal que seja próxima da primeira. Pelas características do problema, é adequado utilizar a *distância de Hausdorff* como métrica de erro. A distância de Hausdorff fornece uma medida do quão longe dois subconjuntos de um mesmo espaço métrico estão um do outro, deste modo, podemos calcular a distância entre a malha de saída e a malha de entrada. Uma outra métrica bastante utilizada por ser simples de ser calculada é a distância quadrática média como pode ser visto em [20].

Sejam M e M' duas malhas poligonais, podemos calcular a *distância de Hausdorff direcionada* (d_H) entre M e M' segundo a equação 2.1, em que p e q representam vértices [21]. A distância de Hausdorff não é uma métrica simétrica, por isso utilizamos o termo *direcionada*. Uma definição simétrica e mais abrangente é dada pela equação 2.2. Estas equações apontam um caminho teórico para o cálculo dessa medida entre malhas, sendo a implementação prática objeto de estudo de trabalhos como [21, 22, 23].

$$d_H(M, M') = \max_{p \epsilon M} \min_{q \epsilon M'} ||p - q|| \tag{2.1}$$

$$D_H(M, M') = max\{d_h(M, M'), d_h(M', M)\} \tag{2.2}$$

A segunda reflexão que queremos apresentar consiste no fato de que toda a informa-

ção que temos sobre a superfície de interesse vem de amostras capturadas por sensores e, portanto, estão sujeitas a imprecisões inerentes ao processo de aquisição. Sendo assim, é questionável se durante a simplificação devemos ser tão rígidos quanto à conformidade da malha final com a malha original. Detalhes geométricos e até mesmo topológicos nas amostras podem não ser tão próximos da superfície original. Evidentemente, nós, enquanto observadores e operadores das etapas de aquisição e simplificação dos dados, podemos exprimir uma avaliação qualitativa sobre as malhas e nos orientar em conjunto por medidas quantitativas de erro, mas esse é um fato interessante de se levar em consideração na escolha de métodos e construção de hipóteses para heurísticas.

As abordagens mais comuns consistem em métodos iterativos que operam a partir da aplicação sucessiva de um operador de simplificação em pontos da malha. A decisão de onde o operador será aplicado em cada iteração é orientada pela medida de erro utilizada, podendo-se considerar também outros critérios como, por exemplo, propriedades intrínsecas da malha como distorção e curvatura. A estratégia pode ser gulosa - tentar, em cada iteração, aplicar o operador sempre onde causará o menor erro - ou não, valendo-se de heurísticas mais sofisticadas. Podemos enquadrar estes métodos de simplificação nas classes de *clusterização de vértices*, *decimação de vértices* ou de *contração de arestas*.

O estudo de técnicas de simplificação desempenha dois papéis importantes neste trabalho. O primeiro é a realização da simplificação propriamente dita, como uma etapa de pré-processamento da malha poligonal densa, obtida a partir da imagem de profundidade, antes que a utilizemos como base para a reconstrução. Após a captura de uma imagem de profundidade, construímos uma nuvem de pontos conectada na resolução da imagem e depois formamos triângulos com os pontos próximos segundo um limiar de distância determinado. É necessário segmentar essa estrutura por componente conexa e aplicar um algoritmo de simplificação para obter uma malha na resolução do objeto a ser utilizada nas etapas seguintes do método. O segundo papel dos estudos sobre simplificação neste trabalho é o de prover uma métrica para a estimação do erro de aproximação. A métrica baseada em quádricas, além de prover um meio de calcularmos o erro de aproximação cometido também desempenha um papel central na etapa de geração de vértices para a poligonização da superfície, uma vez que as quádricas codificam a geometria da superfície de uma maneira eficiente, permitindo a preservação de pontos característicos e a utilização em cenários de pouca memória.

2.3.1 SIMPLIFICAÇÃO DE SUPERFÍCIES USANDO QUÁDRICAS COMO MÉTRICAS DE ERRO

A métrica quádrica é uma das mais utilizadas para medir o erro cometido em trabalhos recentes sobre simplificação de malhas poligonais. Isto porque as quádricas possibilitam uma codificação compacta da geometria de uma superfície utilizando apenas uma matriz

simétrica 4×4 para cada vértice. Em uma malha de triângulos, cada vértice é associado ao conjunto de planos base dos triângulos que o contém. Seja $p = \begin{bmatrix} a & b & c & d \end{bmatrix}^T$ o plano definido pela equação $ax + by + cz + d = 1$ em que $a^2 + b^2 + c^2 = 1$, e $v = \begin{bmatrix} v_x & v_y & v_z & 1 \end{bmatrix}^T$ um vértice em coordenadas homogêneas, o erro em v é definido a partir da forma quadrática $\triangle(v) = v^T Q v$ em que Q é uma matriz 4×4. Segundo [24], temos:

$$\triangle(v) = \sum (p^T v)^2$$

$$\triangle(v) = \sum (v^T p)(p^T v) = \sum v^T (pp^T) v \tag{2.3}$$

$$(pp^T) = \begin{bmatrix} a^2 & ab & ac & ad \\ ab & b^2 & bc & bd \\ ac & bc & c^2 & cd \\ ad & bd & cd & d^2 \end{bmatrix} = K_p \tag{2.4}$$

A matriz pp^T , referida como K_p, define a *quádrica fundamental de erro* para cada plano p considerado e pode ser usada para calcular o quadrado da distância de qualquer ponto do espaço ao plano p. O erro no vértice v é dado pela soma das quádricas fundamentais de todos os planos em consideração, ou seja, o erro é calculado como o quadrado da soma das distâncias do vértice v a cada um destes planos. Repare que, como v pertence a todos os planos calculados na inicialização da quádrica, o erro neste ponto inicialmente é nulo. A contribuição de cada plano é sintetizada em uma única matriz Q.

$$\triangle(v) = v^T (\sum K_p) v = v^T Q v \tag{2.5}$$

Originalmente, a métrica quádrica foi introduzida em [24] junto com a proposta de um método iterativo de simplificação baseado na contração de pares de vértices, uma generalização da contração de arestas. Uma contração de par de vértices, representada em [24] por $(v_1, v_2) \rightarrow v$ é uma operação que move o vértice v_1 para uma nova posição v, conecta todas as arestas incidentes no vértice v_2 ao vértice v_1 e exclui o vértice v_2. Quaisquer arestas ou faces que tenham sido degeneradas no processo são removidas da representação da malha. Quando o par (v_1, v_2) não é uma aresta, o algoritmo também considera casos em que a topologia do modelo não é preservada. Isso não necessariamente é um problema, pois dependendo da aplicação que se deseja, a topologia da superfície pode ser um fator secundário; além disso, considerando a reflexão que apresentamos na seção anterior, a topologia aferida pode não corresponder exatamente à topologia ideal do ente representado.

É razoável pensar que boas aproximações da malha original são obtidas se os vértices não se afastarem muito de suas posições iniciais, por isso, é possível limitar os pares de vértices válidos para a contração por meio de um limiar de distância aceitável entre eles. Se quisermos restringir as ações apenas às arestas, por exemplo, basta especificar uma distância topológica nula.

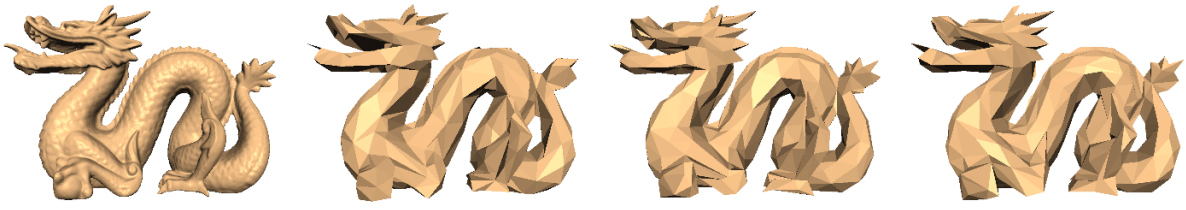
Para cada contração $(v_1, v_2) \rightarrow v$, calcula-se uma nova matriz Q dada pela soma $Q_1 + Q_2$ em que Q_1 é a quádrica correspondente a v_1 e Q_2, a correspondente a v_2. Idealmente, o vértice v é posicionado no resultado da otimização da quádrica Q, isto é, v é o ponto que minimiza a soma dos quadrados das distâncias aos planos dos triângulos que contém os vértices v_1 e v_2. Se a matriz Q não for inversível, tenta-se determinar uma posição para v em uma otimização restrita ao segmento v_1v_2. Se ainda assim não for possível calcular uma posição para v, escolhe-se v dentre o ponto médio e as extremidades do segmento v_1v_2. O erro associado à realização dessa operação é dado por $v^T(Q_1 + Q_2)v$ ou $v^T(Q)v$.

Esta propriedade aditiva das quádricas torna esta métrica muito eficiente e prática de ser utilizada, pois se quiséssemos considerar mais vértices em uma operação, seria possível somar as quádricas de todos esses vértices e obter uma nova quádrica que codificasse a distância a todos estes planos. Nos casos em que os vértices estão contidos em alguns triângulos em comum, temos uma redundância na representação, pois os planos que contém estes triângulos são considerados mais de uma vez. Em uma malha de triângulos, porém, cada plano pode ser computado no máximo 3 vezes. Considerando o fato de que dados adquiridos do mundo real sempre estão sujeitos a diversas formas de ruídos, parece razoável aceitar um certo grau de imprecisão pelo benefício de manter a operação de adição de quádricas simples e computacionalmente eficiente.

Com uma forma de calcular o erro cometido em uma contração, podemos compreender o funcionamento iterativo do algoritmo. Primeiramente, utiliza-se o limiar de distância entre os vértices para selecionar todos os pares válidos para contração. Depois disso, calcula-se o vértice otimizador para cada par e utiliza-se o erro $v^T(Q)v$ como custo da operação. Todos os pares são colocados em uma fila de prioridade baseada no custo da contração. Finalmente, remove-se iterativamente o par de menor custo da fila, executa-se a contração e atualiza-se os custos dos pares em que for necessário.

2.3.2 SIMPLIFICAÇÃO DE SUPERFÍCIES COM DUAS FASES ACOPLADAS

Embora o método de simplificação de superfícies apresentado em [24] produza resultados muito bons, em um cenário de recursos computacionais mais limitados e necessidade de interação em tempo real, podemos precisar reduzir a quantidade de operações realizadas. Métodos baseados em clusterização de vértices geralmente são computacionalmente menos custosos, porém produzem resultados mais grosseiros, principalmente quando o nível de simplificação desejado é elevado.



(a) Original model (b) Separate passes (c) Coupled passes (d) QSlim

Figura 2.1: Comparação entre simplificação com fases acopladas e seguidas [20]

O trabalho de Garland e Shaffer em [20] apresentou uma abordagem híbrida, em que realiza-se uma fase de clusterização de vértices por subdivisão espacial uniforme acoplada a uma fase de contração de pares de vértices. É importante compreender que a proposta do trabalho não é realizar a clusterização de vértices sobre a malha de entrada e, então, realizar a contração de pares de vértices de maneira independente. A etapa inicial de clusterização de vértices, ao mesmo tempo que reduz a quantidade de dados do modelo de entrada, calcula quádricas para cada um dos vértices de um modelo intermediário de saída. Esse modelo intermediário é utilizado como meio de comunicação entre as duas fases do processo, de modo que durante a realização da contração de pares de vértices, as quádricas consideradas para a função de custo são aquelas calculadas em relação à geometria original ao invés da geometria obtida após a clusterização.

Essa é uma diferença que possui impacto significativo sobre a qualidade dos resultados obtidos, aproximando-os do resultado obtido quando utiliza-se diretamente a contração de pares de vértices sobre o modelo, porém reduzindo o tempo computacional em até 80% [20]. A Figura 2.1 ilustra uma comparação qualitativa dos resultados. Em modelos muito densos, a clusterização de vértices pode reduzir drasticamente a quantidade de dados disponíveis e a contração de pares de arestas, além de continuar a simplificação, possibilita a relocação dos vértices em posições mais adequadas a uma boa aproximação da superfície.

Por ser computacionalmente eficiente e com pouco prejuízo à qualidade dos modelos produzidos, este método baseado em uma abordagem de duas etapas de simplificação acopladas parece mais adequado para aplicações em tempo real. Uma outra característica interessante deste método é que na verdade ele fornece um *framework* para o acoplamento de quaisquer dois métodos de simplificação que possam utilizar quádricas como métricas de erro. Poderíamos, por exemplo, realizar uma clusterização de vértices por segmentação espacial uniforme e, em seguida, realizar uma simplificação por operadores estelares [25] utilizando as quádricas em relação à superfície original para guiar os custos de cada iteração do algoritmo.

2.4 POLIGONIZAÇÃO DE SUPERFÍCIES IMPLÍCITAS

Dados um campo escalar $\phi : \mathbb{R}^d \to \mathbb{R}$ e uma constante $\sigma \in \mathbb{R}$, o conjunto $\{x : \phi(x) = \sigma\}$ é chamado um conjunto de nível de ϕ [26]. Sendo σ um valor da função ϕ, temos que o conjunto de nível σ de ϕ é dado por $\phi^{-1}(\sigma)$; quando $d = 3$ e σ é um valor regular de ϕ, podemos chamar este conjunto $\phi^{-1}(\sigma)$ de superfície de nível. Se a função ϕ for contínua, e σ for um valor regular, temos que $\phi^{-1}(\sigma)$ divide o espaço $\mathbb{R}^d$ em três classes de pontos: aqueles que têm valor σ e, consequentemente, estão sobre a superfície; aqueles que tem valor superior a σ e estão do lado exterior à superfície e aqueles que têm valor inferior a σ e estão do lado interior à superfície.

Diversos métodos de poligonização de superfícies implícitas se apoiam sobre essa segmentação que a superfície realiza no espaço em que está imersa e propõem uma estratégia de subdivisão sistemática desse espaço em células, geralmente com a geometria de um cubo ou de um simplexo. Com esta estrutura, é possível amostrar os valores da função implícita nos vértices de cada célula e procurar por aquelas arestas em que um dos vértices possui valor superior ao nível da superfície que desejamos reconstruir enquanto o outro vértice possui valor inferior. Geralmente, adequa-se a função implícita de modo que a superfície de interesse seja a superfície de nível 0; assim, os vértices da grade de amostragem podem ser classificados em positivos ou negativos conforme sua posição relativa à superfície. Convencionando que vértices positivos se encontram no interior da superfície enquanto os negativos, no exterior, podemos concluir que as arestas da grade de amostragem cujos vértices possuem sinal diferente são aquelas arestas que cruzam a superfície. Chamaremo-nas de arestas bipolares.

Esta estratégia é uma forma bastante intuitiva de se obter uma estimativa grosseira da localização da superfície implícita. O desafio do problema de poligonização de superfícies implícitas está em como determinar a estrutura topológica e geométrica da superfície a partir destes dados extrínsecos e, então, gerar uma representação intrínseca dela. Valendo-se de algumas hipóteses sobre o comportamento local da superfície, tenta-se reconstruí-la por partes analisando cada célula. Os procedimentos executados nesta etapa de poligonização diferenciam os métodos atualmente existentes, que podem operar conforme uma grade primal - coincidente com a grade de amostragem - ou uma grade dual.

Um problema inerente às aplicações de reconstrução geométrica a partir de imagens de profundidade consiste em como integrar os dados provenientes das diferentes câmeras em um único modelo. Tentar trabalhar com uma representação paramétrica dos dados, normalmente usada para renderização em pipelines como *OpenGL*, seria bastante difícil e ineficiente. Por conta disso, trabalhos como [15, 16, 28] adotam uma representação implícita da superfície e realizam sua poligonização por meio de algoritmos de poligonização de superfícies implícitas.

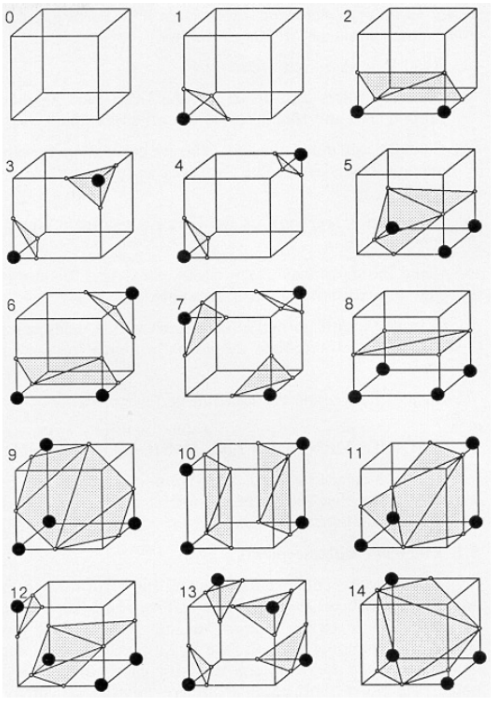


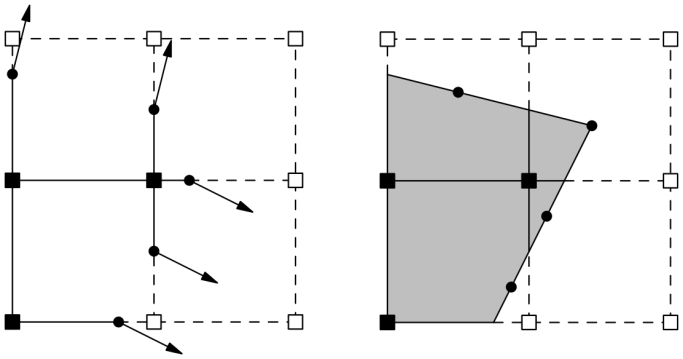
Figura 2.2: Possíveis configurações de poligonização para uma célula [29]

2.4.1 GRADE PRIMAL E GRADE DUAL

Os métodos que calculam vértices posicionados sobre a grade de subdivisão do espaço são classificados como métodos primais. O algoritmo *Marching Cubes* [29], um dos mais populares na área de poligonização de superfícies, é um exemplo de método primal que opera em uma grade regular de cubos.

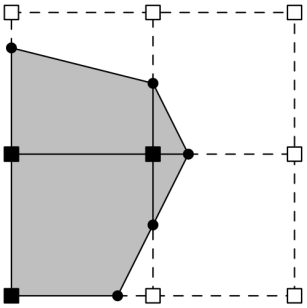
O Marching Cubes propõe que a estrutura topológica da aproximação poligonal de uma superfície implícita seja determinada por uma tabela, explorando o limite das possíveis formas que podemos classificar os vértices de uma célula. Como um cubo tem 8 vértices, a priori teríamos 256 configurações possíveis para a topologia local da superfície em relação à célula cúbica. Entretanto, levando-se em consideração as simetrias e rotações das células, é possível reduzir essa quantidade para 15, conforme ilustra a tabela exibida na Figura 2.2. Embora essa tabela nos permita identificar a topologia local da superfície, ainda é necessário determinar sua geometria, isto é, a posição de cada vértice que compõe os polígonos. Sabendo que a posição dos vértices está restrita às arestas da grade e conhecendo os valores da função implícita em cada um dos vértices da célula, podemos determinar a posição dos vértices dos polígonos por meio de uma interpolação destes valores nos extremos de cada aresta bipolar.

De outra parte, temos os métodos duais, que calculam posições para os vértices dentro das células e ligam vértices de células que compartilham arestas bipolares, constituindo uma grade dual à grade de amostragem. Métodos duais possuem maior flexibilidade no



(a) Célula marcada com dados de Hermite

(b) Reconstrução com Dual Contouring



(c) Reconstrução com Marching Cubes

Figura 2.3: Reconstrução de quinas

posicionamento dos vértices, pois eles não ficam restritos às arestas da célula como na grade primal, podendo ser posicionados em qualquer ponto no interior da célula. Um dos métodos duais mais antigos é o Surface Nets [30]. Por apresentar melhores resultados quanto à qualidade da malha gerada e por utilizar a métrica de matrizes quádricas apresentada no trabalho de simplificação de superfícies, escolhemos o método *Dual Contouring of Hermite Data* [31] como base para este trabalho.

Nos métodos primais, geralmente não temos conhecimento sobre o comportamento da superfície no interior da célula e, portanto, estruturas como quinas podem ser suavizadas como ilustrado na Figura 2.3c. Aumentar o refinamento da grade de amostragem ajuda a obter uma melhor aproximação da superfície nesses casos, mas aumenta quantidade de memória e esforço computacional necessários.

2.4.2 DUAL CONTOURING DE DADOS DE HERMITE

Em uma abordagem de segmentação espacial uniforme, há muitas células que não cruzam a superfície, mas precisam ser alocadas em memória e processadas. A superfície, neste caso, é reconstruída com uma malha uniforme, utilizando-se a mesma quantidade de

polígonos para regiões mais planas e para regiões mais detalhadas. O uso de uma estrutura espacial adaptativa como uma octree permite que a alocação de memória do algoritmo seja muito mais eficiente e que a malha reconstruída seja mais adaptada à geometria da superfície. No entanto, quando uma estrutura de dados adaptativa é utilizada, as regiões da grade de amostragem onde há contato entre células de níveis de detalhes diferentes podem gerar buracos nas malhas. É necessário tratar esses casos com particular atenção para gerar os polígonos e integrá-los adequadamente à malha em construção.

O trabalho [31] apresenta uma solução de poligonização de superfícies implícitas em grade de amostragem não uniforme cujos vértices estejam classificados como interiores ou exteriores à superfície. A posição dos vértices no interior das células é calculada a partir de dados de Hermite da função implícita (valor da função e sua derivada de primeira ordem), representados pelos pontos de interseção entre a superfície e as arestas da grade e os vetores normais à superfície nestes pontos.

Por não precisar dos valores da função implícita nos vértices da grade de amostragem, mas apenas a classificação destes vértices em interiores ou exteriores à superfície, este método é bastante favorável aos nossos experimentos, pois não dispomos de uma maneira direta de amostrar valores da função. Como iremos operar com malhas poligonais como dados de entrada, seria necessário propor um modelo de função implícita tal como o modelo de função de distância com sinal tal como em [16, 28] e, então, estimar os valores da função indiretamente com uma técnica de traçados de raios, por exemplo.

Destacamos duas contribuições deste método que são bastante relevantes para os experimentos deste trabalho:

1. Uma maneira de calcular a posição dos vértices no interior da célula utilizando dados de Hermite a partir da otimização de *Quadratic Error Functions* (QEF), funções quadráticas similares às apresentadas em [24].

2. Uma maneira eficiente de poligonizar uma superfície a partir de uma estrutura espacial adaptativa, uma *octree*, evitando buracos na malha gerada.

O trabalho de Dual Contouring de Dados de Hermite propõe uma maneira de tratar a poligonização de superfícies em uma octree notando que apenas as arestas bipolares que não contém propriamente outra aresta são as responsáveis por gerar polígonos. Estas arestas são chamadas de *arestas mínimas* pelos autores. Os detalhes quanto à ordem de processamento das células e associação dos vértices estão descritos em [31] e [32].

A menor restrição sobre os vértices de uma grade dual e a informação do vetor normal à superfície possibilitam que características de detalhes finos da superfície como, por exemplo, quinas e outras saliências geométricas da cena sejam preservadas mesmo em uma grade menos refinada. A posição do vértice no interior da célula é obtida pela otimização (minimização) da função de erro quadrático calculada a partir das normais e dos pontos de interseção nas arestas.

3 O MÉTODO

Nosso método é composto por 5 etapas principais que podem se repetir como um ciclo:

1. Aquisição de dados geométricos
2. Construção de nuvem de pontos conectada
3. Simplificação de malha
4. Geração de estrutura intermediária
5. Poligonização

Nesta seção, descreveremos o funcionamento de cada uma dessas etapas, explicando como elas se encaixam em um ciclo de captura e reconstrução incremental de objetos e cenas estáticos. A Figura 3.1 representa uma visão geral do pipeline. Embora os experimentos realizados neste trabalho tenham sido conduzidos com um Kinect e um computador convencional, cada etapa do método será descrita nesta seção levando-se em consideração os recursos de um dispositivo móvel. Adotamos esta abordagem, pois o nosso objetivo é propor um método que possa ser executado completamente em um dispositivo móvel. Devido a limitações de tempo nesta pesquisa, não implementamos esta solução em um dispositivo móvel, porém utilizamos os resultados dos experimentos para nos orientar na formulação desta proposta.

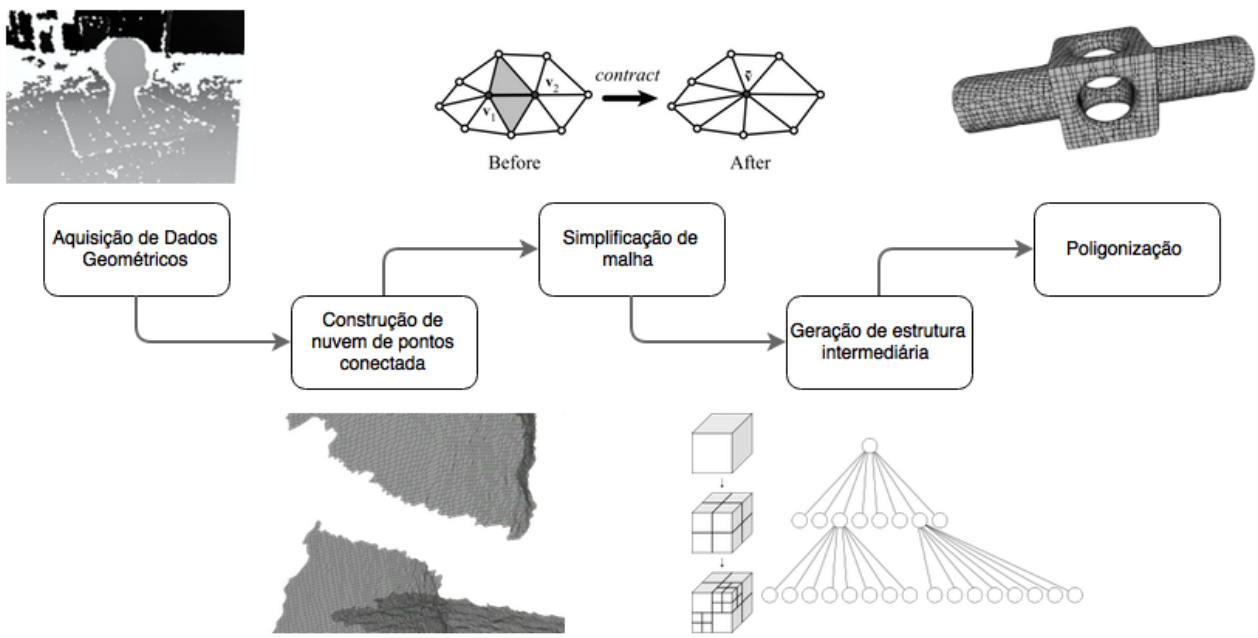
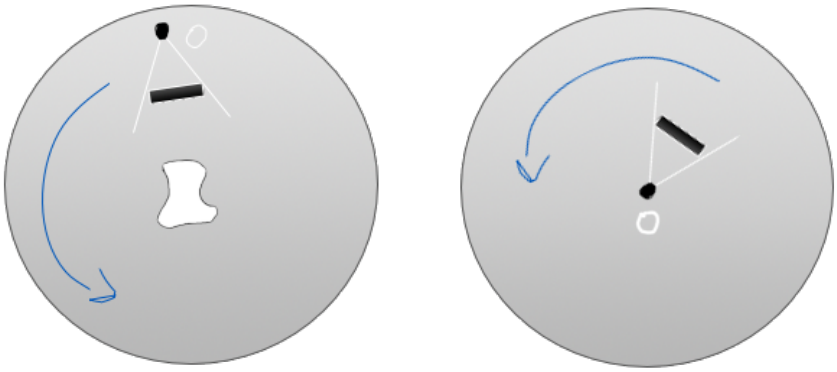


Figura 3.1: Ilustração do pipeline

3.1 AQUISIÇÃO DE DADOS GEOMÉTRICOS

Inicialmente, utiliza-se um dispositivo móvel para capturar uma imagem de profundidade do objeto ou cena de interesse. As vezes fazemos distinção entre os termos *objeto de interesse* e *cena de interesse*, embora pudéssemos tratar todo o conteúdo da imagem como parte de uma cena. Essa distinção, contudo, guarda uma relação com o posicionamento do usuário no momento de capturar as imagens e os dados geométricos. Quando se está interessado em um objeto específico, é razoável capturar imagens de pontos de vista em torno do objeto, em um movimento similar a uma rotação (Figura 3.2a), podendo aproximar-se ou afastar-se segundo o nível de detalhes. Por outro lado, quando se está interessado no espaço da cena como um todo, o movimento de captura tende a ser centrado próximo ao usuário, que move sua câmera em várias direções (Figura 3.2b).



(a) Captura em torno de objeto (b) Captura da cena

Figura 3.2: Formas de captura da imagem de profundidade

Por hipótese, estamos interessados em reconstruir um objeto incrementalmente, por isso basta capturar um único quadro de um ponto vista nesta etapa. Novos quadros são adicionados em iterações seguintes do ciclo. É nesse momento também que armazenamos os dados de calibração da câmera de profundidade, obtidos pelos sensores de atitude e outros recursos do dispositivo tal qual uma câmera com lente olho de peixe.

3.2 CONSTRUÇÃO DE NUVEM DE PONTOS CONECTADA

Após a aquisição da imagem, utiliza-se os dados no mapa de profundidade para construir uma nuvem de pontos conectada. Cada pixel da imagem é convertido em um vértice em um espaço tridimensional. Os dados de calibração obtidos na etapa anterior são utilizados para registrar estes vértices no referencial de reconstrução. O registro dos pontos no referencial correto é um passo importante para obtermos bons resultados, pois estamos reconstruindo um objeto ou cena por partes, a partir de informações de pontos de vista diferentes e que se complementam.

Como temos uma nuvem de pontos organizada, isto é, uma nuvem de pontos em que as coordenadas espaciais X e Y podem ser indexadas em termos das linhas e colunas do mapa de profundidade em uma ordem lógica [33], podemos explorar essa propriedade para obter uma estrutura conectada. Para isso, adotamos uma estratégia bastante simples, embora ineficiente em relação ao custo de armazenamento. Primeiramente, delimitamos um limiar de distância máximo aceitável como tamanho das arestas de triângulos. Depois disso, percorremos a imagem de profundidade pixel a pixel, verificando, para cada pixel, se a distância entre seu ponto correspondente no espaço da cena e o de seus vizinhos é inferior ao limiar estabelecido; caso seja, forma-se um triângulo. A vizinhança considerada é 4-conectada. Ao final desta etapa, produzimos uma malha triângular densa com aspecto uniforme. A Figura 3.3 ilustra o resultado este procedimento.

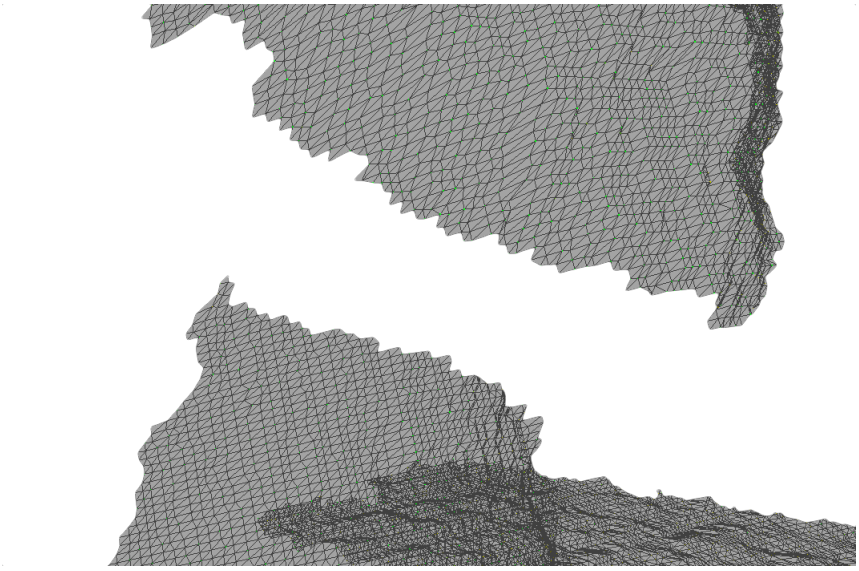


Figura 3.3: Construção da Nuvem de Pontos conectada

3.3 SIMPLIFICAÇÃO DE MALHA

Neste momento, temos dois problemas que podem prejudicar o funcionamento do método. O primeiro, é que a estrutura produzida na etapa anterior é bastante densa, apresentando muitos vértices e faces para serem processados. Trabalhar com a quantidade de dados de uma estrutura densa como esta iria requerer muito esforço computacional para cada quadro capturado. Além disso, a natureza uniforme da malha faz com que ela não leve em consideração a geometria dos objetos capturados e nós temos interesse em reconstruir malhas adaptadas à essa geometria. Se, ao final desta etapa, tivermos uma malha menos uniforme, mais adaptada, isso irá facilitar a nossa reconstrução.

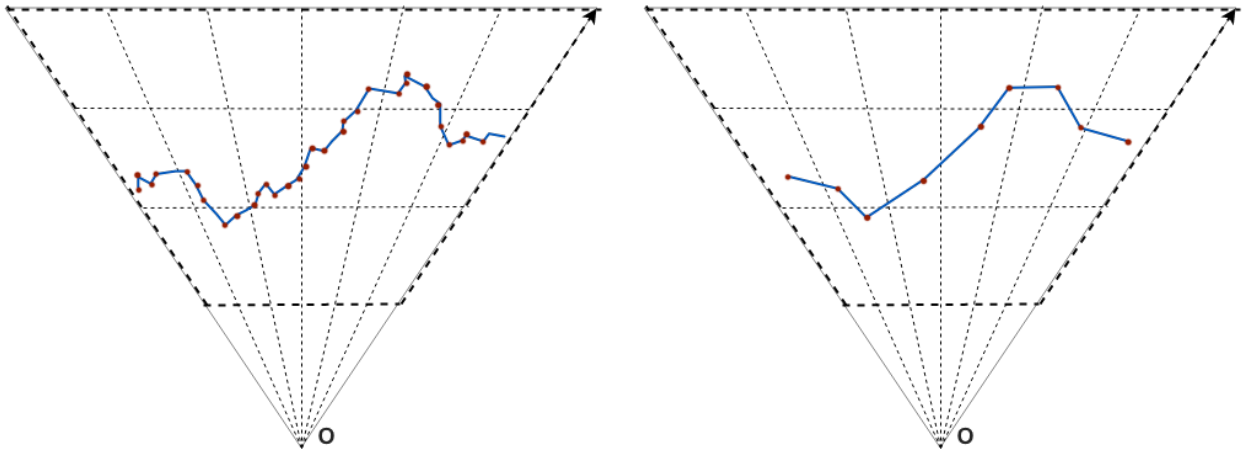
O segundo problema é que tratando de dados reais, adquiridos com um aparelho de uso comercial, como estamos interessados, há bastante ruído na aquisição. Para minimizar estes dois problemas e também calcular dados úteis para serem utilizados em etapas

posteriores, vamos recorrer a técnicas de simplificação de malhas poligonais.

O processo de simplificação é realizado em duas etapas acopladas de maneira similar ao que é feito em [20]. A primeira etapa consiste em uma clusterização dos vértices da malha, ou seja, segmenta-se o espaço em uma determinada resolução de células e, então, substitui-se todos os vértices que estão contidos em uma mesma célula por um único vértice. Para cada vértice, calcula-se uma quádrica utilizando-se o próprio vértice e a normal de cada face da malha que o contém. Assim, consideramos todos os planos das faces que contém o vértice, sendo ele localizado na interseção desses planos. Para cada célula, calculamos uma quádrica somando as quádricas de cada vértice contido nela. A quádrica da célula é associada ao seu vértice representativo, cuja posição pode ser calculada pela otimização dessa quádrica.

Uma consideração importante que fazemos sobre o processo de clusterização é quanto à forma da célula. Geralmente, algoritmos de clusterização de vértices utilizam células cúbicas, podendo fazer uma segmentação uniforme ou adaptativa do espaço. A julgar pela natureza dos dados que estamos trabalhando, é razoável utilizar uma segmentação em células considerando um volume diferente. Como os dados geométricos são adquiridos a partir de uma câmera e guardam uma relação de indexação com os pixels da imagem de profundidade, podemos utilizar uma região em formato de tronco de pirâmide, considerando o campo de visão da câmera (*view frustrum)*, e então segmentar esta região em células. A Figura 3.4 ilustra essa técnica em duas dimensões.

Esta abordagem parece melhor por duas razões: a primeira é que a precisão geométrica decai com a distância ao sensor, então os dados adquiridos de pontos mais próximos são mais precisos do que os daqueles mais distantes; a segunda, é que, em relação à câmera, os vértices que estão mais próximos na direção de um raio são os que determinam o que é visível, e este modelo de aglutinação leva isso em conta. Há, ainda, o fato de que devido à projeção cônica de captura, objetos mais afastados em relação à câmera terão pontos mais esparsos, com maiores distâncias, embora isso não corresponda às suas dimensões reais. Dessa forma, uma segmentação de volume em forma de tronco de pirâmide, dividindo uniformemente os ângulos dos raios em relação à câmera pode prover melhores resultados.



(a) Superfície Amostrada (b) Superfície simplificada

Figura 3.4: Segmentação do Espaço em Forma de Tronco de Pirâmide

Após essa redução inicial dos dados por clusterização dos vértices, continuamos o processo de simplificação realizando o colapso de pares de vértices tal como em [24]. Um detalhe importante a ser observado é que esta etapa utiliza as quádricas calculadas na etapa anterior, em relação à malha original, para determinar quais pares serão colapsados e qual será a posição do novo vértice representativo. Como descrito em [20] esse processo de acoplamento entre as etapas apresenta um resultado muito superior à realização de uma etapa de clusterização seguida do colapso de pares de vértices em que as quádricas da segunda etapa são calculadas em relação à malha produzida ao final da primeira.

Como em [24], a posição dos vértices é determinada pela otimização da quádrica resultante do par a ser colapsado. Esse processo reduz significativamente a quantidade de dados a ser processada, produzindo uma malha poligonal com um número reduzido de vértices e faces, além de filtrar um pouco do ruído de aquisição. A malha obtida ao final desta etapa possui quádricas associadas a cada vértice e que levaram em consideração dados desde a geometria original adquirida.

3.4 REPRESENTAÇÃO EXTRÍNSECA

Nesta etapa, utilizamos a malha e as quádricas produzidas na etapa anterior para construir ou atualizar uma representação extrínseca da superfície de interesse. Mantemos a representação da superfície em uma *Octree* [27], uma estrutura de dados espacial que nos permite segmentar o espaço de maneira adaptada. Cada célula da octree representa um volume em forma de cubo no espaço, podendo ser subdivido em 8 cubos menores, células filhos, interiores, pelo uso de 3 planos ortogonais passando pelo centro do cubo. Os dados de interesse ficam armazenados nas folhas. As palavras *célula* e *nó* serão utilizadas alternadamente para se referir ao mesmo objeto neste contexto; usaremos *célula* quando a ênfase estiver no volume do espaço ocupado; e *nó*, quando a ênfase estiver na organização

da estrutura de dados.

Como visto na seção 2.4, trabalhos que utilizam métodos implícitos para reconstrução de superfícies geralmente se valem de uma segmentação do espaço em células. Essa segmentação pode ser uniforme ou adaptada a algum critério. Com essa estrutura espacial, podemos amostrar valores da função implícita nos vértices de cada célula e verificar se a superfície de nível que estamos interessados passa por aquela célula ou não. Esta verificação pode ser feita observando-se as arestas em que em um dos vértices a função possua valor superior ao nível de interesse e no outro, inferior.

Nosso objetivo é construir uma estrutura similar à apresentada em [31], pois então poderemos aplicar o mesmo algoritmo descrito naquele trabalho para obter uma malha poligonal. O algoritmo apresentado em [31] é interessante para nós, pois ele permite a extração de uma malha poligonal fechada a partir de uma octree sem restrições sobre a disposição de seus nós. Para isso, basta que a octree possua dados de Hermite sobre a função implícita nas suas folhas. Dados de Hermite são representados pelo valor de uma função e o de todas as suas derivadas até uma determinada ordem. Neste trabalho, utiliza-se apenas a derivada de 1^a ordem, representada pelo vetor normal à superfíce no ponto de interseção com a aresta da célula. Assim, para cada célula da octree, precisamos conseguir classificar seus vértices em interior ou exterior à superfície e, para as arestas que intersectam a superfície, precisamos calcular o ponto de interseção e a normal à superfície neste ponto.

De posse de uma forma analítica da função, estas tarefas ficam bastante simplificadas. Contudo, estamos aplicando este modelo para uma situação em que o nosso dado de entrada é uma malha poligonal. Uma abordagem que costuma ser adotada nesse caso, é a definição de uma função de distância com sinal à superfície de interesse, apresentada em [28] e seguida por diversos trabalhos, dentre eles [15, 16]. Esta função representa um campo escalar no espaço em que a superfície está mergulhada, tal que para cada ponto de amostragem calcula-se a distância à superfície. Caso o ponto esteja no exterior da superfície, o valor da função é positivo; caso esteja no interior, negativo. A escolha do lado positivo ou negativo é arbitrária, o importante é que interior e exterior tenham sinais opostos. Dessa forma, podemos realizar os cálculos com essa função de distância, em que a superfície de nível 0 é a nossa superfície de interesse. Para tornar o processo computacionalmente mais eficiente, é possível definir esta função implícita apenas em uma *vizinhança tubular* da superfície; neste caso, chamamos de função de distância com sinal truncada (*Truncated Signal Distance Function - TSDF* [15, 16]).

Neste trabalho, pela natureza dos dados que precisamos calcular, não iremos trabalhar com uma função de distância com sinal diretamente. Para cada aresta de uma célula, tentamos calcular 3 dados: o ponto de interseção entre a aresta da célula e a superfície, o vetor normal à superfície neste ponto e a classificação dos vértices extremidades da

aresta em interior ou exterior à superfície. Vamos descrever cada passo desse processo nas subseções seguintes.

3.4.1 SELEÇÃO DO VOLUME

Primeiramente, define-se o centro e as dimensões da área a ser segmentada. O volume é escolhido como cúbico de modo a tentar manter os cálculos isotrópicos em relação aos três eixos coordenados. Estes valores são baseados na superfície de interesse. Com os dados do primeiro quadro capturado, podemos estimar o tamanho da área de interesse tomando, por exemplo, como centro o centro da caixa envolvente da malha de entrada e definindo a dimensão deste volume como um múltiplo da maior dimensão desta malha. Todos os referenciais de medida - caixa envolvente, maior dimensão da malha etc - são calculados em relação aos eixos coordenados. A Figura 3.5 ilustra esse procedimento em 2 dimensões.

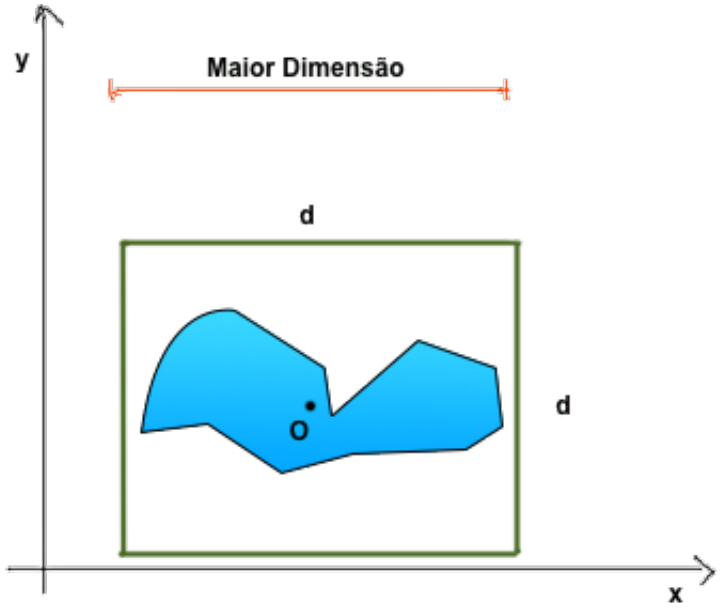


Figura 3.5: Cálculo do Volume de Interesse [34]

Uma alternativa à alocação de todo o volume de uma única vez por meio de estimativas, é calcular um volume mais justo à caixa envolvente da primeira malha e, então, caso seja necessário, agregar novas células à estrutura. Nesse caso, a arquitetura não representaria uma única octree, mas sim um nó geral como raiz, que pode ter uma quantidade arbitrária de filhos. Cada um desses nós filhos seria raíz de uma octree, podendo haver interseções entre elas. A Figura 3.6 ilustra esta arquitetura em 2 dimensões, em que cada cor representa uma *quadtree* segmentando o espaço.

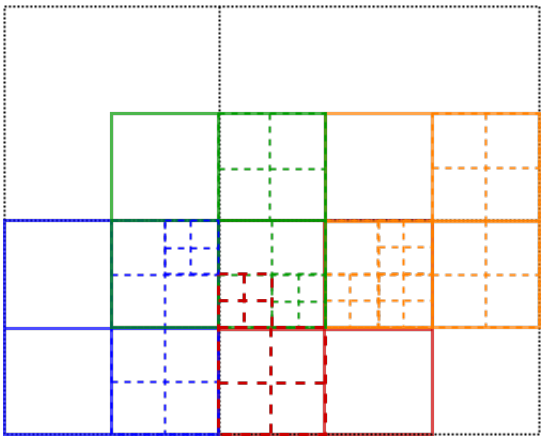


Figura 3.6: Arquitetura espacial com múltiplas Octrees

3.4.2 SEGMENTAÇÃO E ORGANIZAÇÃO DOS DADOS

Com o volume de interesse definido, prossegue-se ao processamento dos dados. Na célula raiz do volume, única existente no começo do procedimento, cria-se uma referência para todas as faces da malha de entrada. Após isso, subdivide-se uniformemente esta célula em 8 células-filho. Idealmente, gostaríamos de identificar os triângulos da célula-pai que são interiores ou intersectantes (intersectam alguma aresta) em relação às células-filho. Classificar em interior é uma tarefa simples, visto que requer somente algumas operações de comparação: um vértice está no interior de uma célula se suas coordenadas encontram-se entre as coordenadas mínima e máxima de cada plano da célula. Por sua vez, um triângulo está no interior de uma célula se os seus 3 vértices estão no interior da célula. Por outro lado, para classificar um triângulo como intersectante à célula, precisaríamos realizar mais operações de cálculo e isto deixaria o algoritmo consideravelmente mais lento, uma vez que a grande maioria dos triângulos não intersecta a célula. Dessa forma, o que fazemos, caso o triângulo não esteja no interior da célula, é verificar se ele se encontra no exterior dela, também usando apenas comparações. Caso ele não esteja nem completamente no interior, nem completamente no exterior, então ele é classificado como potencial intersectante da célula.

Observe que, embora um triângulo possa estar contido em apenas uma única célula para cada nível da octree, ele pode ser classificado como potencialmente intersectante para diversas células deste mesmo nível. Este processo repete-se recursivamente até que uma das seguintes condições seja satisfeita:

1. A lista de triângulos interiores e a lista de potenciais intersectantes estão ambas vazias

2. A lista de triângulos interiores da célula corrente e da sua célula pai são ambas vazias, mas a lista de potenciais intersectantes é não vazia.

3. A profundidade da célula corrente é igual ao limite máximo de profundidade deter-

minado.

Se o primeiro caso ocorre, significa que não há dados da superfície na região representada pela célula corrente. Neste caso, os dados que houverem nesta célula são destruídos e à celula-pai é atribuido o valor nulo para a posição correspondente a esta célula filho. No segundo caso, subdividir a célula não traria acréscimo de informação, uma vez que não há triângulos em seu interior. Contudo, esta célula pode ter dados para a reconstrução e, portanto, ela marcada como uma possível folha da octree e é passada para uma outra função que realiza o cálculo dos dados que são necessários. O terceiro caso é, na verdade, um critério de parada para evitar que o algoritmo desça níveis demais e acabe consumindo muita memória ou até mesmo rodando por períodos indefinidos de tempo.

A destruição de células que não possuem informação para a reconstrução é realizada ao final da recursão. Caso uma célula não possua nenhuma referência não nula em seu vetor de filhos, ela é destruída e em sua célula-pai atribui-se o valor nulo à referência correspondente. Este processo é repetido recursivamente.

3.4.3 CÁLCULO DOS DADOS

Ao identificarmos uma potencial célula folha, realizamos os cálculos propriamente ditos. Cada célula tem 8 vértices a serem classificados e 12 arestas que podem ou não intersectar a superfície. Criamos uma lista que contém a classificação de cada vértice da aresta em interior, exterior ou desconhecido – a lista é inicializada com o valor desconhecido para todos.

Para cada aresta, percorre-se a lista de potenciais triângulos intersectantes da célula e verifica-se a existência de interseção entre a aresta e algum dos polígonos. Este teste é feito por um algoritmo de interseção entre segmentos e triângulos. Não basta apenas descobrir se há interseção ou não, é necessário calcular o ponto exato da interseção. Criamos uma lista responsável por armazenar todas as interseções na aresta corrente, bem como uma lista para armazenar todas os vetores normais correspondentes a estas interseções. Se houver interseção, armazena-se o ponto de interseção na lista correspondente. O vetor normal à face intersectada é armazenado como a normal à superfície na respectiva interseção. Este vetor é calculado como o resultado normalizado do produto vetorial entre duas arestas do triângulo. Se não houver interseção em nenhuma das 12 arestas da célula, seus dados são deletados e ela é descartada, atribuindo-se o valor nulo à posição correspondente em sua célula-pai.

Após percorrermos a lista de potenciais triângulos intersectantes, verificamos os dados obtidos. Caso haja apenas uma interseção entre a aresta corrente e a superfície, podemos classificar os vértices da aresta sem problemas. Sejam A e B os pontos extremidades da aresta corrente, C o ponto de interseção entre a aresta e a superfície e N a normal ao triângulo intersectado, relizamos o seguinte cálculo:

$$S = (C - A) \cdot N \tag{3.1}$$

Como o vetor normal tem a orientação no sentido do interior para o exterior da superfície, caso o valor S seja positivo, temos que o ponto A encontra-se no exterior da superfície; caso contrário, no interior. O ponto B, por sua vez, deve ter sinal contrário ao de A. A Figura 3.7 ilustra esse processo em 2 dimensões.

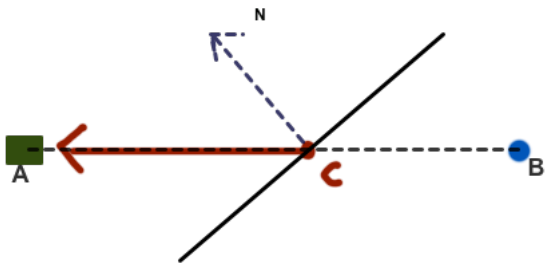
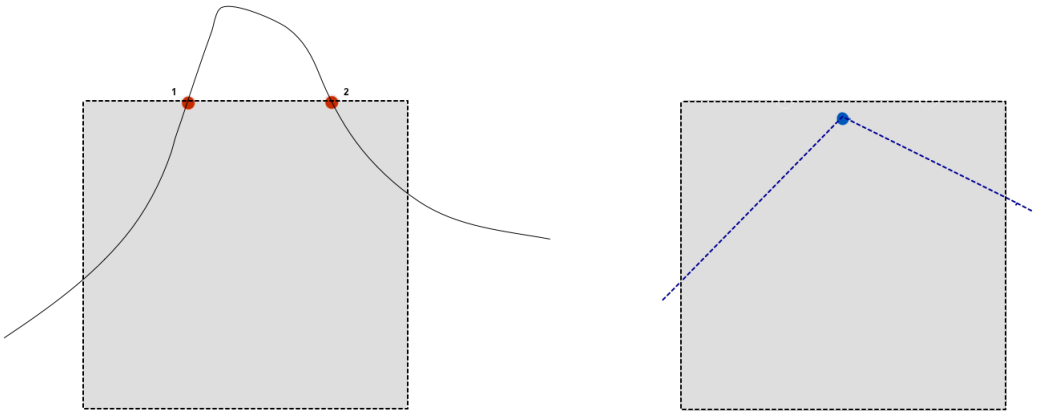


Figura 3.7: Cálculo do sinal das extremidades da aresta AB

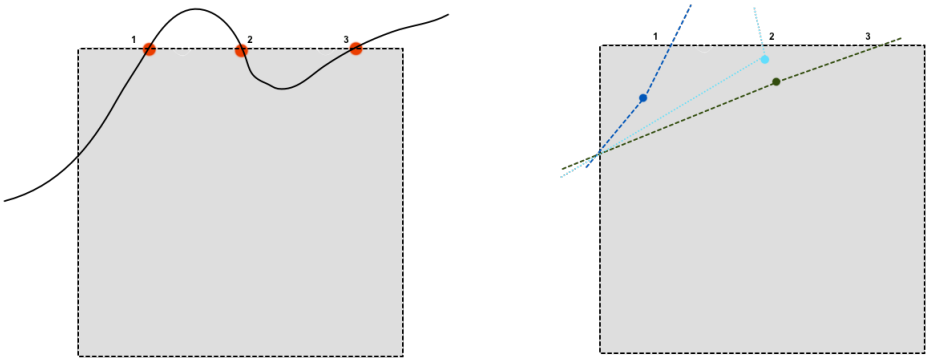
Se houver mais de uma interseção entre a aresta e a superfície, a classificação dos vértices fica um pouco mais difícil. Em um primeiro momento, tentamos evitar essa situação, pois ela é um sinal de que o grau de discretização do espaço nesta região pode não estar adequado à resolução de detalhes do objeto. Portanto, verificamos se a profundidade da célula corrente é inferior ao limite máximo de profundidade. Caso seja, esta outrora possível folha é classificada como nó interior e descemos mais um nível de subdivisão na árvore, tentando calcular estes dados em cada uma de suas células-filho.

Caso a profundidade limite tenha sido atingida, verifica-se a paridade do número de interseções na aresta. Se o número de interseções for par, então ambos os vértices da aresta têm o mesmo sinal, pois a aresta "entrou e saiu" da superfície. Se esse número for ímpar, eles têm sinais diferentes. Para calcular o sinal correto, escolhe-se, arbitrariamente, um dos pontos da aresta, por exemplo, o ponto A. Após isso, determina-se qual o ponto de interseção mais próximo do ponto A, percorrendo-se a lista de pontos de interseção. Calcula-se, então, o sinal de A, de acordo com a 3.1, considerando C como o ponto de interseção mais próximo de A e N como a respectiva normal nesse ponto. Se o número de interseções na aresta for par, desconsideraremos essas interseções e B terá o mesmo sinal de A; caso contrário, terá sinal oposto. Na Figura 3.8, temos uma representação da situação em que ocorre uma dupla interseção na aresta de uma célula e uma possível reconstrução geométrica. A Figura 3.9 ilustra o caso de uma tripla interseção.



(a) Número par de interseções (b) Possível reconstrução

Figura 3.8: Dupla interseção na mesma aresta



(a) Tripla interseção (b) Possíveis Reconstruções

Figura 3.9: Multiplas Interseções na Mesma Aresta

A classificação dos vértices das células em exterior ou interior à superfície é um dado importante para reconstruir a topologia da superfície. Para a geometria, utilizaremos quádricas, pois a quádrica é uma forma de codificar o plano normal à superfície de nível de interesse no ponto de interseção. Na solução que implementamos em nossos experimetos, para cada célula, os pontos de interseção de cada aresta, bem como o vetor normal correspondente, eram acumulados em uma quádrica representativa desta célula. Esta abordagem apresentou resultados razoáveis, com algumas falhas na geometria devido a problemas de otimização das quádricas. Por conta disso, acreditamos que uma solução melhor seria somar as quádricas provenientes da simplificação para cada vértice no interior da célula, pois assim teríamos uma codificação da geometria referente aos dados originais, desde a aquisição. Esta é a nossa proposta para esta etapa do método, embora não tenha sido possível validá-la por experimentos no tempo destinado a este trabalho.

Um ponto de discussão a se ponderar quando há mais de uma interseção sobre uma aresta é se devemos utilizar apenas uma delas, por exemplo, aquela escolhida para calcular o sinal de uma extremidade, ou se devemos considerar todos os pontos de interseção no processo de calcular a quádrica. Qualquer uma dessas opções é imprecisa no retrato da situação. Contudo, visto que a topologia da malha será baseada na hipótese de haver uma única interseção sobre a aresta, optamos por considerar apenas o ponto A.

Nesse processo, nem todos os vértices das célula são classificados. Aqueles vértices que não fazem parte de nenhuma aresta intersectada, por exemplo, permanecerão com o valor *desconhecido* de sinal. Se a célula estiver próximo de uma região bem comportada da superfície, isto é, uma região em que a malha for localmente uma variedade sem bordos, a célula apresentará uma configuração de sinais em que será possível definir o sinal dos vértices *desconhecidos* a partir de seus vizinhos na célula. A figura Figura 3.10 ilustra um exemplo de configuração de sinais coerente. Os vértices marcados por um círculo azul foram classificados no exterior da superfície, enquanto o vértice marcado com um quadrado verde, no interior. As arestas que não possuem interseção não geram informações de classificação para os vértices e, portanto, há alguns vértices que possuem o valor desconhecido – representado na figura por uma interrogação. Contudo, é possível inferir o valor desses vértices a partir do valor de seus vértices vizinhos de modo que não há conflitos, dúvidas, na classificação final.

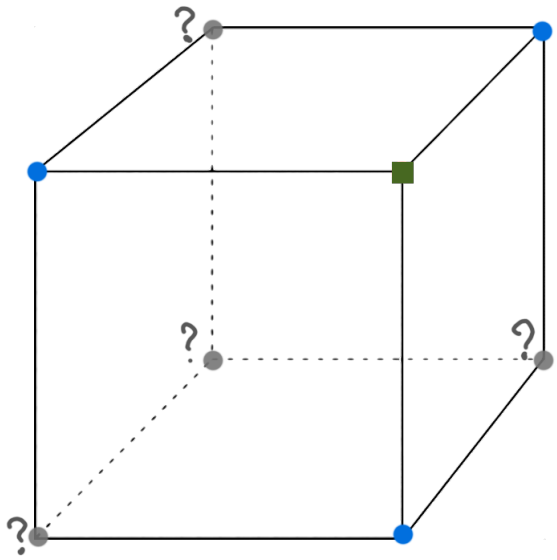


Figura 3.10: Célula com configuração de sinais consistente

No entanto, se a malha tiver bordo, por exemplo, podemos ter uma configuração ambígua de classificação. Neste caso, não realizamos a classificação dos vértices desconhecidos neste momento. Este passo é postergado como o último passo antes de iniciar o processo

de poligonização dessa estrutura.

Na Figura 3.11, os vértices com valor desconhecido, marcados com um triângulo vermelho ao lado de um sinal de interrogação não podem ser classificados a partir do valor de seus vizinhos sem que haja uma incoerência, isto é, sem que haja uma incompatibilidade entre as interseções detectadas e a classificação proposta. No contexto deste trabalho, assumimos que a cena capturada é estática e que cada imagem nos fornece um fragmento de um mesmo objeto. Assim, postergando a classificação destes vértices, é possível que seja capturado um novo quadro com informações complementares que permitam classificar os demais vértices tornando a célula como um todo em uma configuração consistente.

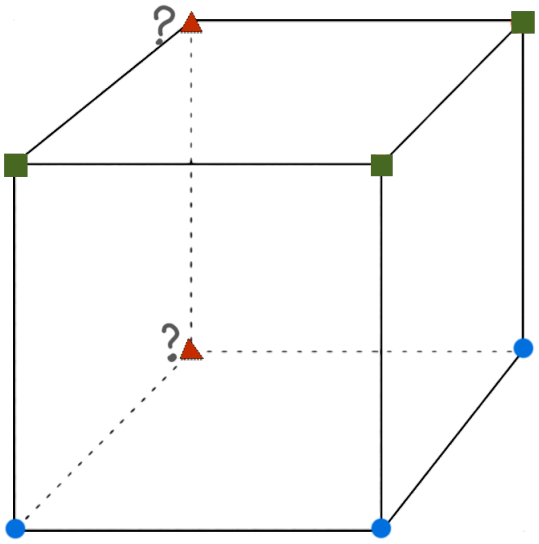


Figura 3.11: Célula com configuração de sinais ambígua

3.5 ATUALIZAÇÃO INCREMENTAL

Descrevemos o processo de aquisição até a construção de uma representação extrínseca da superfície para um único quadro. Agora vamos descrever as adaptações necessárias para a atualização da estrutura apresentada para congregar novos quadros adquiridos pelos sensores de captura. A etapa de aquisição é idêntica, não muda para nenhum quadro em particular, apenas captura-se a imagem de profundidade com o dispositivo adequado e armazena-se os dados de calibração.

A etapa de simplificação também é completamente independente das demais. Como ressaltado anteriormente, é necessário utilizar os dados de calibração para se realizar o registro adequado dos pontos em relação ao referencial do mundo. Da mesma forma, durante o processo de simplificação, armazena-se as quádricas para cada vértice remanescente, calculadas em relação à superfície original pelo mesmo processo de clusterização de

vértices acoplado ao colapso de pares de vértices.

A etapa que lida com a estrutura extrínseca da superfície é o ponto em que é necessário realizar as alterações adequadamente. O processo inicia-se de maneira similar ao que é feito para o primeiro quadro, isto é, colocamos na célula raiz uma referência para cada face da malha de entrada e vamos subdividindo as células recursivamente, separando, em suas células-filhos, quais triângulos estão no interior e quais são potenciais intersectantes. Quando tentamos descer por um ramo da árvore em que, por não haver informação nos quadros anteriores, a célula filho era nula, criamos uma nova célula e tratamos a computação deste caso da mesma forma como é tratado o primeiro quadro.

Quando descemos por um ramo onde estamos lendo células criadas em passos anteriores, primeiro verificamos se a lista de triângulos interiores e a lista de potenciais intersectantes estão ambas vazias, pois neste caso não temos informações a acrescentar na região e, portanto, retornamos a mesma célula inalterada. Depois, verificamos se o nó em que estamos é interior ou é uma folha da árvore. Se for um nó interior, continuamos descendo recursivamente. Note que, nesse caso, podemos descer por um ramo com células já existentes ou por um ramo nulo. Se o nó em questão for uma folha, temos 2 opções de tratamento: atualização do nó ou refinamento da estrutura.

Se a lista de triângulos interiores na célula pai do nó corrente for não vazia e a profundidade da célula corrente for inferior ao limite máximo estabelecido, significa que podemos refinar a estrutura, tornando o nó atual um nó interior e descendo até folhas mais profundas. As folhas serão criadas pelo mesmo processo adotado na construção do primeiro quadro.

Caso a condição de refinamento não seja satisfeita, atualizamos a célula folha. Para atualizá-la, calculamos exatamente os mesmos dados que precisaríamos calcular se fôssemos construí-la da primeira vez: pontos de interseções com as arestas, vetor normal para cada ponto de interseção e classificação dos vértices em interior ou exterior à superfície. A atualização da quádrica da célula é feita simplesmente pela adição da quádrica calculada nesse passo com a quádrica acumulada já existente na célula. A atualização dos sinais é feita comparando-se o sinal já presente com o novo sinal calculado:

1. Ambos os sinais são iguais. Neste caso, não é preciso fazer nada
2. O sinal anterior é desconhecido. Neste caso, adota-se o sinal calculado na atualização
3. Os sinais são diferentes. Neste caso, marcamos este vértice como ambíguo e deixamos para ajustar o seu sinal na última etapa antes da poligonização

Dessa forma, temos um processo bem definido para atualização da nossa representação extrínseca da superfície. Ao final da etapa de construção/atualização da representação extrínseca, dependendo da aplicação, podemos retornar para a etapa 1 e capturar um

novo quadro ou prosseguir para a etapa 5 e gerar uma malha. Em um caso de uso em tempo real, em que deseja-se permitir ao usuário visualizar a progressão da atualização do modelo, faz sentido seguir para a poligonização para só então retornar ao passo 1.

3.6 POLIGONIZAÇÃO

A última etapa do ciclo consiste na poligonização da representação extrínseca e obtenção de uma malha poligonal para podermos visualizar a superfície reconstruída. Antes de iniciar a poligonização, contudo, procedemos ao ajuste da classificação dos vértices da grade de amostragem em interior ou exterior à superfície. Como foi dito na seção 3.4, quando as arestas da célula intersectam a superfície em uma região em que localmente a geometria corresponde à um variedade sem bordo, temos uma configuração de sinais que nos permite classificar os vértices com sinais desconhecidos sem ambiguidade.

Nesse caso, percorremos todos vértices da célula e quando encontramos um vértice com sinal desconhecido, olhamos para os seus 3 vizinhos. Se ao menos um deles tiver sinal positivo ou negativo, atribuímos o mesmo sinal ao vértice. Se todos os vizinhos tiverem sinal desconhecido, passamos para o vértice seguinte e depois realizamos mais uma passagem em todos os vértices, quando, então, algum vizinho terá tido valor atribuído.

A configuração apresentada na Figura 3.10 é uma configuração consistente, pois não há vértices em que um dos vizinhos possua sinal positivo e um outro vizinho possua sinal negativo, gerando ambiguidade. Outro detalhe importante é que nessa configuração sempre é possível escolher uma ordem para analisar os vértices com sinal desconhecido de modo que todos possam ter um valor atribuído. Computacionalmente, para não ter que determinar previamente esta ordem particular, simplesmente realizamos mais uma iteração sobre todos os vértices e saltamos aqueles que não possuem vizinhos com sinal atribuído no momento.

Em se tratando de uma superfície suave e fechada, se ela corta determinadas arestas de uma célula para entrar no espaço da célula, ela precisa cortar outras arestas (ou essas mesmas arestas) para sair. Dessa forma, não é possível criar situações de ambiguidade. Por outro lado, se a superfície tiver bordo, células próximas ao bordo podem apresentar configurações de sinal estranhas uma vez que a superfície pode cortar apenas algumas arestas para entrar e nunca sair (Figura 3.11). Em particular, na situação de interesse para este trabalho, estamos reconstruíndo uma superfície a partir de fragmentos, de dados que representam amostras de partes da superfície. Individualmente, cada fragmento deste é uma superfície com bordo. O processo de atualização das células, no entanto, se encarrega de preencher adequadamente a configuração de sinais da célula, utilizando os dados de vários quadros para que a topologia da malha gerada seja à de uma superfície fechada.

Nas regiões em que tivermos, de fato, bordo na superfície, podemos ter dificuldade em definir a classificação dos vértices. Uma forma de resolver este problema seria traçar raios

a partir da posição da câmera que capturou os dados da região em questão em direção ao vértice que deseja-se classificar. Caso o raio intersecte a superfície dentro da distância entre a câmera e o vértice, então ele se encontra no interior da superfície; caso contrário, ele se encontra no exterior. Esse método baseado em traçado de raios, no entanto, é consideravelmente mais lento, pois requer muito mais cálculos além da manutenção na memória de uma estrutura hierárquica da malha subdivida no espaço para que possamos realizar o teste de interseção entre o raio e a malha sem ter que testar todas as faces da malha.

Por falta de um mecanismo para lidar com essa situação, convencionou-se classificar como exterior os pontos em que não for possível dirimir a ambiguidade. Esta decisão baseia-se no fato de que se um ponto está no interior da superfície, então ele tem que ser classificado como interior a partir de qualquer ponto de vista.

As vezes, pode ocorrer de um vértice ser classificado adequadamente em uma célula e não o ser em outra, por haver uma intereseção em uma aresta não compartilhada. Para fazer essa troca de informações, mantém-se uma tabela de hash com uma chave que indica o vértice e o valor associado indica o sinal atribuído, que pode ser, positivo, negativo, desconhecido ou ambíguo. Esta tabela é consultada durante o algoritmo para se realizar a atualização de valores nas células e na própria tabela. Dessa forma, apenas as células em que algum dos vértices não conseguimos classificar por nenhuma dessas vias é que são submetidas a este procedimento.

O processo de poligonização é essencialmente o mesmo apresentado em [31]. Primeiramente, faz-se uma visita em profundidade aos nós da árvore de modo a atribuir-lhes uma numeração. Neste momento, fazemos uma modificação em relação ao algoritmo original, mas que não altera o processo de poligonização. Na implementação disponibilizada em [35], a otimização da quádrica de cada célula é realizada durante a construção da célula folha. Isto é possível e razoável de ser feito, pois o caso original não considera a construção de uma superfície incrementalmente, portanto neste ponto todas as informações necessárias para se determinar a posição do vértice representativo da célula já está disponível. No nosso caso, como construímos a malha incrementalmente, deixamos este cálculo de otimização para o momento em que requisita-se a malha da superfície. Note que podemos requisitar uma malha da superfície após cada adição de dados e manter uma visualização incremental dos resultados. Porém, se esse não for o caso, estaremos evitando computações desnecessárias.

Em uma segunda passada, visita-se novamente a árvore em profunidade mas desta vez com a utilização de procedimentos recursivos específicos discutidos em [31, 32]. A ideia consiste basicamente em procurar pelas *arestas mínimas* que possuem extremos com sinais diferentes e, então, gerar um polígono conectando o vértice representativo de cada célula que contém esta aresta. Uma aresta é dita mínima se ela é uma aresta de uma célula

folha e ela não contém propriamente uma aresta de uma célula folha vizinha. Ou seja, se duas folhas vizinhas têm o mesmo nível de subdivisão, as arestas nas interseções delas são mínimas. Contudo, se elas tiverem níves diferentes de subdivisão, considera-se-á mínima a aresta pertencente ao cubo com maior grau de subdivisão.

Esta técnica de percorrer e poligonizar a octree possibilita a geração de malhas fechadas e sem redundância de polígonos, algo delicado de ser feito quando há células com níveis diferentes de subdivisão. Repare que se tivermos apenas folhas de mesma profundidade em torno de uma aresta mínima, teremos 4 células, enquanto se tivermos uma folha com profundidade menor contendo esta aresta, teremos 3 células em seu entorno. A priori, ao conectar os vértices representativos de cada célula, poderíamos gerar quadriláteros para o primeiro caso e triângulos para o segundo. Nós optamos por ter uma malha só com triângulos, por isso nos casos em que há 4 células no entorno da aresta mínima, geramos 2 triângulos utilizando uma diagonal do quadrilátero.

Opcionalmente, antes de realizar a poligonização da superfície, podemos fazer uma simplificação da octree, colapsando nós folha em nós com menor profundidade. Esse procedimento é similar a uma simplificação por clusterização de vértices em que leva-se em consideração uma medida do erro quadrático médio cometido calculado a partir da otimização da soma das quádricas da região a ser simplificada.

Especificando-se uma tolerância de erro ϵ, realiza-se uma visita em profundidade à octree acumulando, em cada nó, a soma das quádricas de seus nós filhos. Se o erro calculado para o vértice otimizador da quádrica em alguma célula for menor que ϵ, então podemos remover todos os nós filhos desta célula, tornando-a em uma nova folha. Esta operação não necessariamente preserva a topologia da malha.

4 EXPERIMENTOS E RESULTADOS

Para chegar na proposta de método apresentada na seção 3, foi necessário, além do estudo das técnicas no estado da arte existentes, a realização de experimentos para validá-la. As etapas do método foram desempenhadas separadamente para que pudéssemos ter maior controle sobre cada uma e estudá-las individualmente. Além disso, a condução dos experimentos dessa maneira nos permitiu focar no estudo e desenvolvimento das nossas próprias adaptações e contribuições. Apresentaremos uma demonstração de conceito, abordando cada uma das etapas desde a aquisição até a poligonização. Resultados preliminares obtidos durante a pesquisa podem ser encontrados em [34].

No experimentos, utilizamos um aparelho Kinect de 1^a geração, um tabuleiro de xadrez, uma base giratória e uma cabeça de manequin como nosso objeto de interesse. A implementação foi feita em um computador *desktop* utilizando a linguagem de programação *C++*. Para a etapa de poligonização da superfície, utilizamos as implementações de *Dual Contouring* [31] disponíveis em [35] e [36] como base para a nossa própria. Nossa implementação encontra-se disponível em [37]. Para a visualização dos modelos gerados, utilizamos o programa *Meshlab* [38], que também foi utilizado em alguns procedimentos descritos na próxima seção.

4.1 AQUISIÇÃO DE DADOS GEOMÉTRICOS

Em nosso ambiente de captura, as imagens de profundidade foram adquiridas a partir de um Kinect. A cabeça de manequin, nosso objeto de interesse, foi posicionada no centro do tabuleiro de xadrez, que por sua vez, foi centralizado em relação à base giratória apoiada no chão, conforme mostra a figura Figura 4.1. O Kinect permaneceu fixo durante todo o experimento, sobre um plano a alguns centímetros do chão.

O caso de uso em uma aplicação prática seria com objetos fixos e uma câmera móvel, pois imagina-se um usuário que se move em torno de um objeto e utiliza seu aparelho para capturar imagens de diferentes pontos de vista. Essa diferença, todavia, não afeta o ponto que queremos validar com esse experimento, haja vista que o dado que precisamos obter é similar: imagens de um mesmo objeto (manequin e tabuleiro) capturadas de ponto de vistas diferentes e dados de calibração da câmera. Por conta disso, montamos o experimento da forma mais conveniente para minimizar a chance de erro de aquisição.

A informação de calibração será essencial para o registro de todos os dados geométricos na etapa seguinte. Como nesse experimento não estamos utilizando um aparelho com sensores de atitude para capturar as imagens, precisaremos realizar a calibração da câmera separadamente. O tabuleiro de xadrez foi introduzido na cena para que tivéssemos um padrão conhecido e um referencial para realizar a calibração da câmera a partir das imagens capturadas. Marcamos o quadrado próximo ao canto superior esquerdo, em

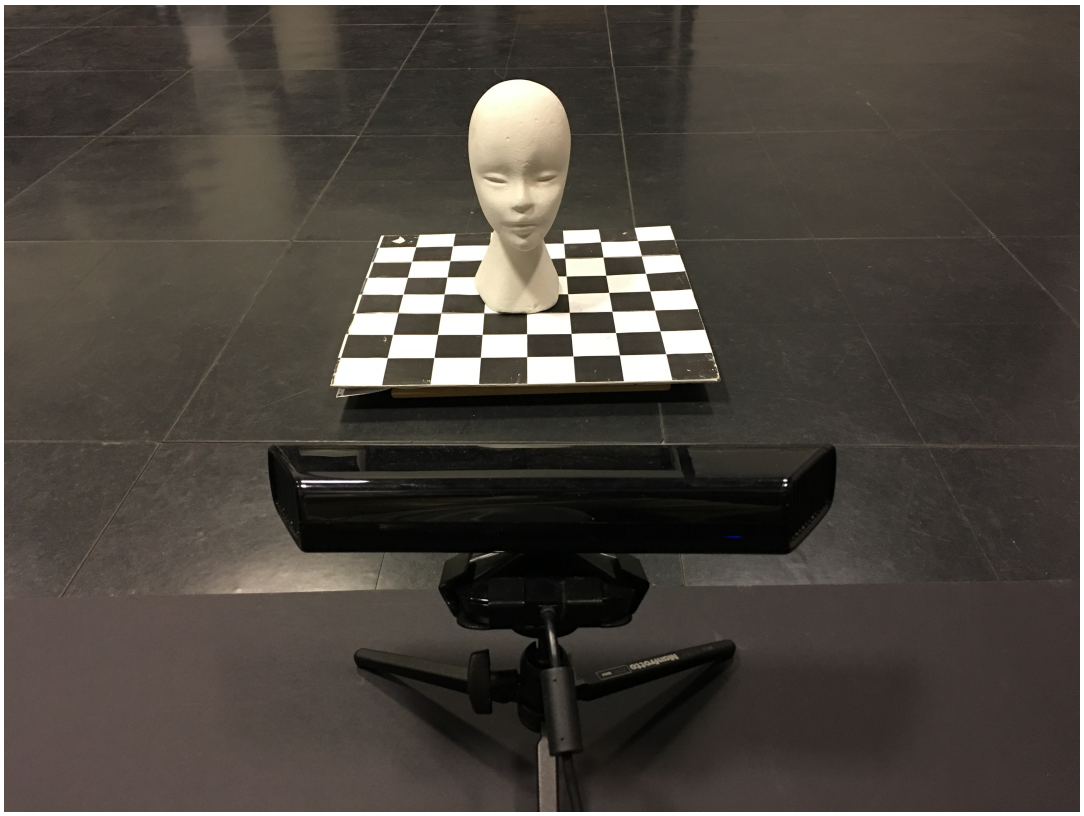


Figura 4.1: Ambiente de Captura

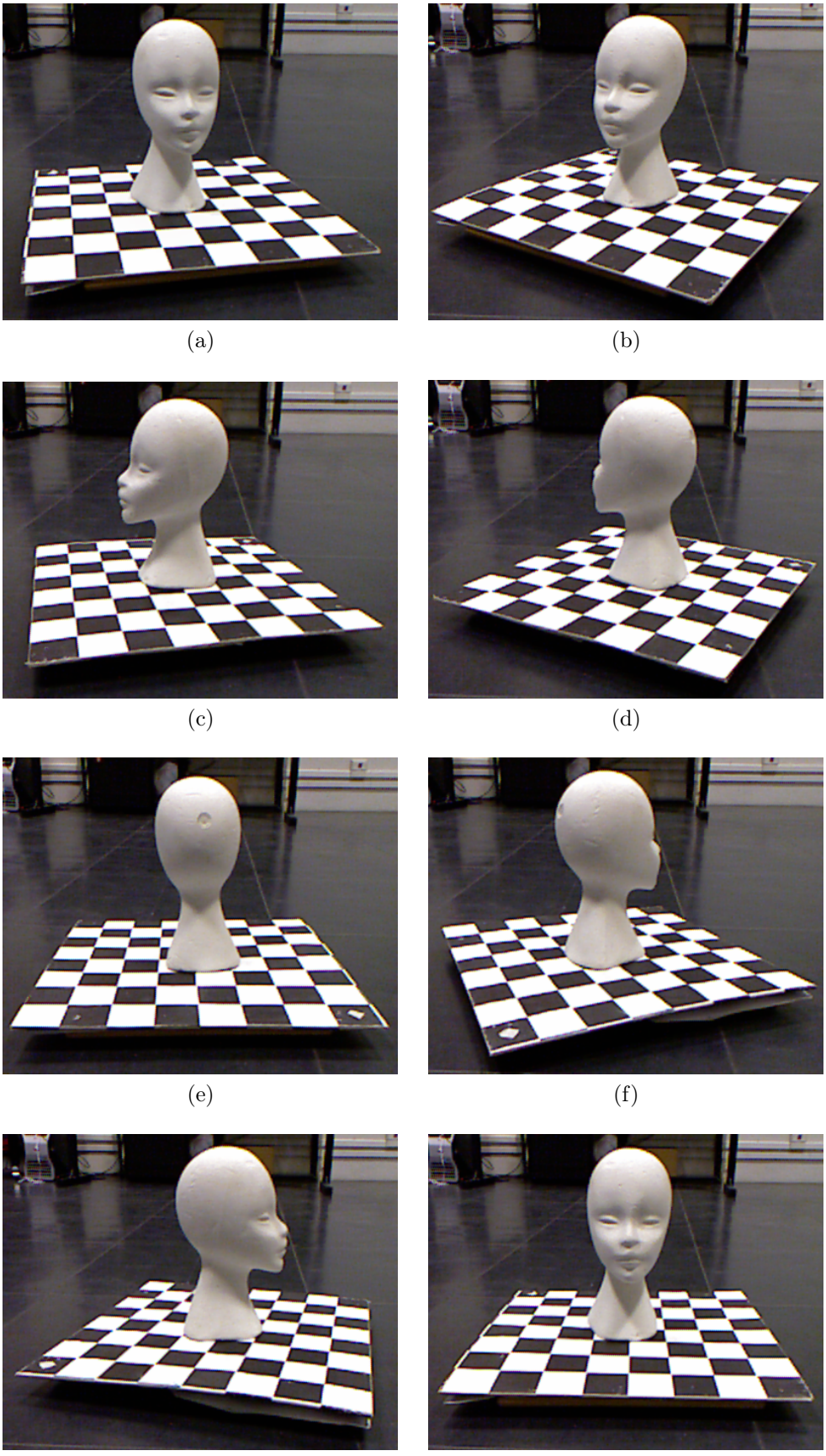
relação à Figura 4.1 para facilitar a identificação do ponto referenciado como origem do sistema de coordenadas. A calibração foi calculada com o programa *QtCalib* [41], uma ferramenta de calibração baseada no algoritmo de Tsai [42].

A Figura 4.2 apresenta os 8 quadros capturados durante a fase de aquisição. A Figura 4.3 mostra as imagens de profundidade após uma acentuação de contraste para fins de visualização. Pode-se observar bastante ruído na aquisição dos dados de profundidade.

4.2 NUVEM DE PONTOS CONECTADA

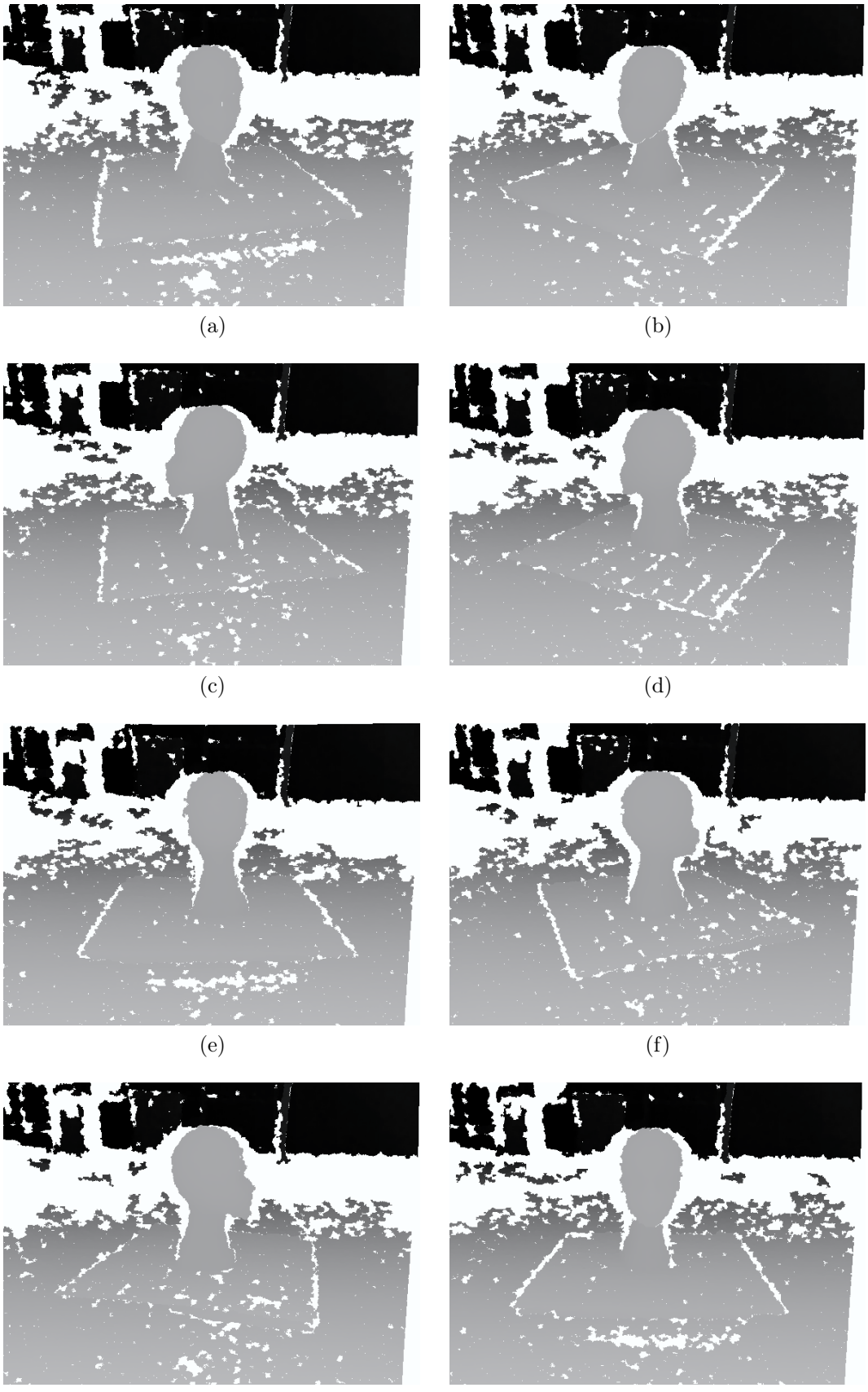
Após a captura das imagens, precisamos construir uma nuvem de pontos conectada e registrar os pontos em relação a um referencial do mundo. Com os dados dos parâmetros intrínsecos da câmera, pegamos o valor de profundidade de cada pixel e calculamos uma posição no espaço 3D para ele. Essa posição está referenciada em relação ao Kinect, como se a origem do sistema de coordenadas fosse a localização do sensor de profundidade. Os pontos para os quais não temos informação são ignorados, eles são identificados no mapa de profundidade com o valor 0. Aplicando a transformação inversa da matriz de calibração formada pelos parâmetros extrínsecos da câmera, levamos os pontos ao referencial desejado. Por conta do deslocamento da câmera do kinect, é necessário fazer uma correção de translação nos pontos.

Os pontos vizinhos são conectados para formarem triângulos se a distância entre eles



(a) (b) (c) (d) (e) (f) (g) (h)

Figura 4.2: Quadros capturados em cores



(a) (b) (c) (d) (e) (f) (g) (h)

Figura 4.3: Quadros capturados (profundidade)

for menor do que a média das distâncias entre todos os pontos vizinhos. O critério de conectividade é importante, pois por conta do ruído, usando um limiar de distância muito baixo, podemos acabar com muitos buracos em regiões afastadas de bordos, visto que há falta de informação de profundidade em vários pontos. Para nós, é melhor ter uma malha que seja razoavelmente densa, com poucos buracos fora da borda, e deixar a responsabilidade de correções geométricas com os efeitos da simplificação e acumulação de quádricas. A Figura 4.4 apresenta o resultado de levar para o espaço 3D e conectar os pontos de cada um dos quadros capturados. É notória a quantidade de ruído decorrente da aquisição, principalmente em um modelo como o que escolhemos para este experimento, pois o tabuleiro de xadrez é um objeto plano e a cabeça de manequin é um objeto com uma superfície bastante suave, sem variações bruscas de curvatura. A Figura 4.5, por sua vez, ilustra com mais detalhe a malha resultante do quadro 1. Na Figura 4.6 podemos analisar de perto a formação dos polígonos criados e verificar que a estrutura é bastante densa e possui algumas regiões bem homogêneas.

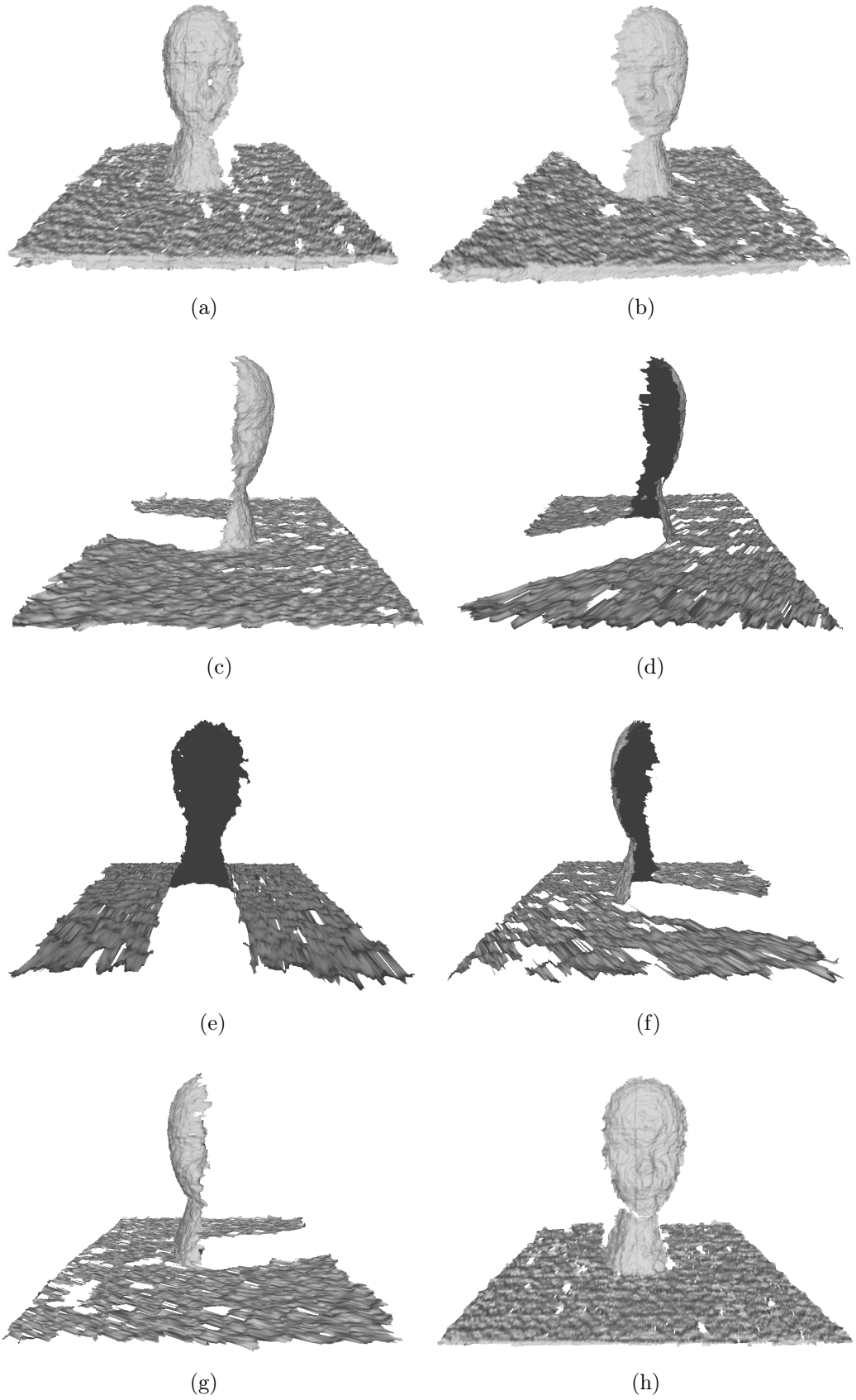
4.3 SIMPLIFICAÇÃO DAS MALHAS

A etapa de simplificação é uma das menos alteradas em relação ao métodos já existentes. À exceção da proposta de modificação da forma das células para a clusterização de vértices, os algoritmos de simplificação seguem inalterados. Dentro do tempo destinado a este trabalho, não foi possível concluir uma implementação própria desta estapa e, por isso, ela foi conduzida com o auxílio do programa *Meshlab*. Cada malha foi simplificada diretamente pelo algoritmo de colapso de pares de arestas descrito em [24]. Como a amostragem de pontos foi consideravelmente densa, a simplificação foi realizada de modo a reduzir o número de faces em aproximadamente 95%, resultando nas malhas exibidas na Figura 4.7.

4.4 REPRESENTAÇÃO EXTRÍNSECA E POLIGONIZAÇÃO

Embora tenhamos realizado as operações das etapas anteriores – construção da nuvem de pontos conectada e simplificação da malha densa – já em todas as malhas como um pré-processamento para esta etapa, a construção da representação extrínseca que usaremos na poligonização é feita de maneira incremental, tal como proposto no seção 3 e como se espera que seja o caso de uso natural em uma aplicação móvel. O pré-processamento foi descrito de uma vez apenas por conveniência, visto que as etapas não estão integradas em um único programa, mas separadas em aplicações fechadas e até mesmo com auxílio de programas como o Meshlab.

Iniciando com o primeiro quadro, calculamos o centro e o tamanho do volume cúbico que será subdivido. A subdivisão ocorre segundo os critérios de parada descritos na seção 3.2, de forma a poupar memória e recursos computacionais em relação a uma divisão



(a) (b) (c) (d) (e) (f) (g) (h)

Figura 4.4: Malhas densas

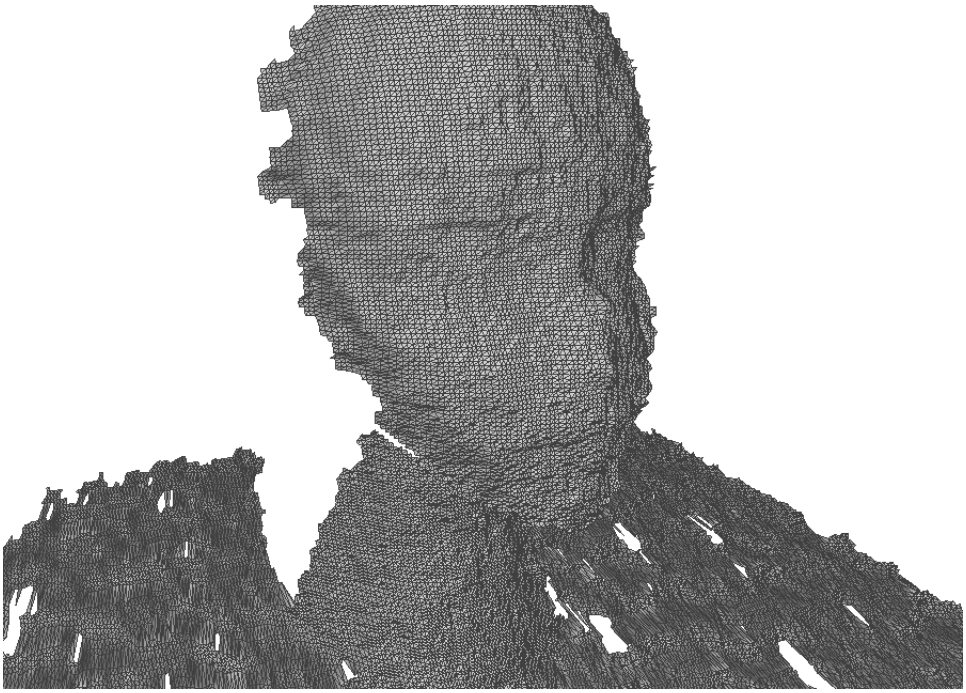


Figura 4.5: Quadro 1 em detalhe

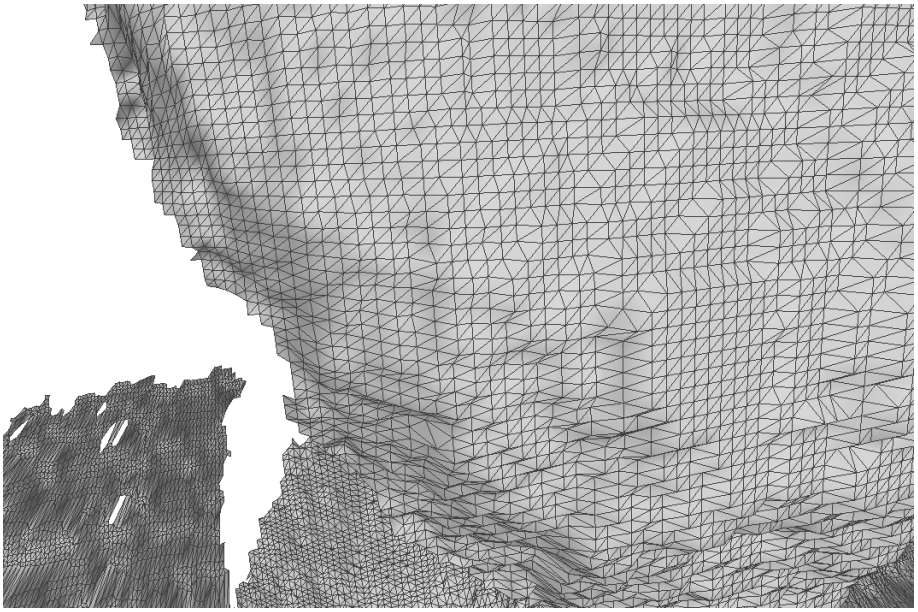
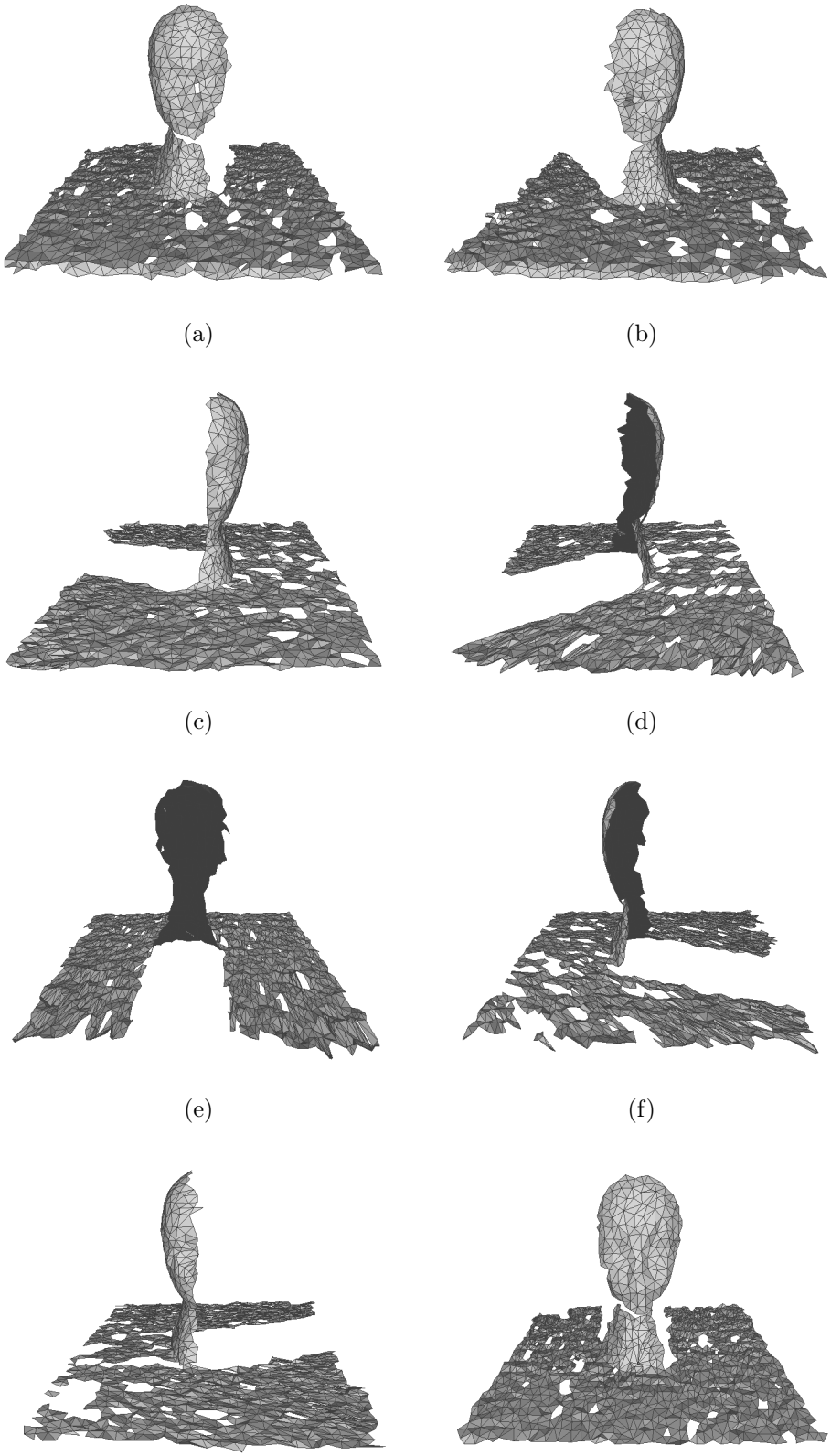


Figura 4.6: Conectividade da malha



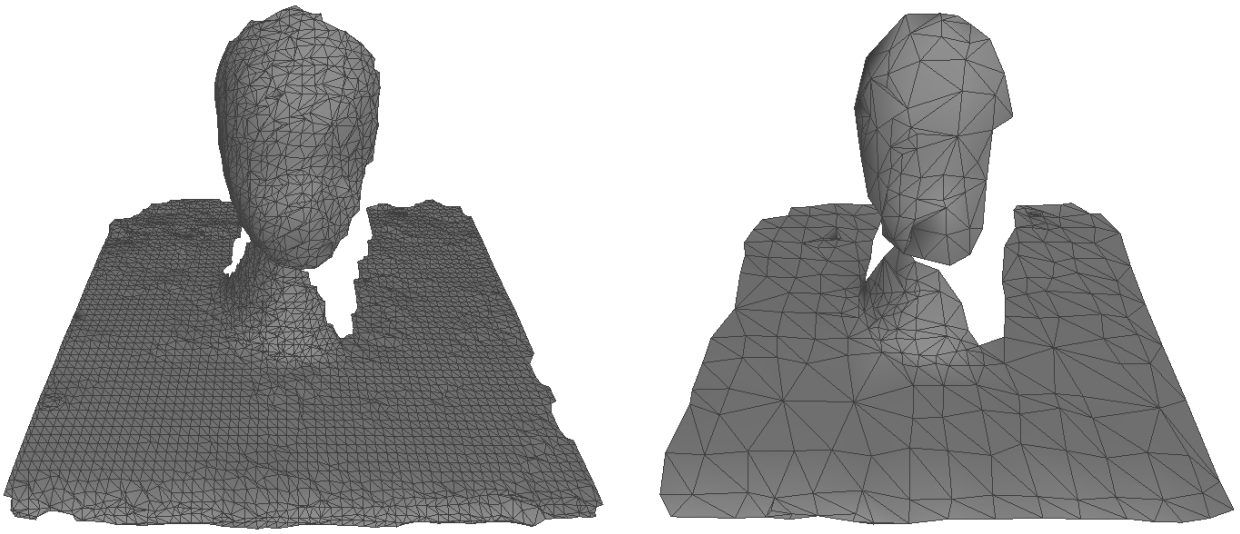
(a) (b)

(c) (d)

(e) (f)

(g) (h)

Figura 4.7: Malhas Simplificadas em 95%



(a) Sem simplificação (b) Com simplificação

Figura 4.8: Reconstrução do quadro 1

uniforme. Como a simplificação foi realizada em separado, não integrada, não temos os dados das quádricas para esta superfície. Estes dados foram, então, construídos a partir do cálculo da interseção entre as arestas da grade de amostragem e a malha poligonal de entrada. No método, já era necessário calcular estas interseções para que tivéssemos informações para reconstruir a topologia da superfície, classificando os vértices em interior ou exterior à malha. Contudo, agora utilizamos os pontos de interseção e as normais das faces calculadas nesses pontos – dados de Hermite – como base para as quádricas, dados mais próximos daqueles propostos por [31] para a poligonização. Durante o processamento de uma célula, para cada interseção calculada, adicionamos o ponto e o vetor normal na quádrica, de modo a considerar o plano tangente local à superfície e permitir uma estimativa da tendência de seu movimento. Esta abordagem também está coerente com a ideia de tentar preservar pontos característicos da superfície.

Utilizando apenas o quadro 1, já podemos realizar a extração de uma malha poligonal dessa representação. A Figura 4.8 ilustra o resultado obtido para a malha do quadro 1. Na Figura 4.8a temos a malha extraída sem simplificação, e na Figura 4.8b, a mesma malha aplicando a simplificação com tolerância de erro de 1 décimo da ordem de grandeza do tamanho do modelo.

Agora, prosseguimos à atualização da representação extrínseca fazendo o processamento do quadro 2. A atualização ocorre da maneira descrita na seção 3.5, considerando a mesma modificação para o cálculo das quádricas que fizemos para o quadro 1, isto é, as quádricas são geradas pela interseção entre a grade de amostragem e a malha poligonal do quadro 2. Observe que a grade de amostragem pode sofrer modificações em relação àquela construída a partir do quadro 1, visto que ela deve contemplar as novas informações adi-

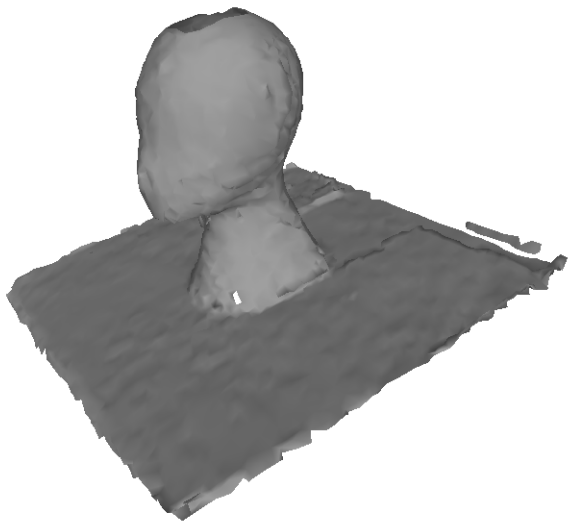
cionadas pelo quadro 2, preenchendo espaços vazios e, possivelmente, refinando detalhes em células antigas. Novamente, poderíamos extrair uma malha poligonal neste ponto do processo. Essa é uma das vantagens de termos um método incremental.

Na nossa implementação, a menos da modificação dos dados de reconstrução geométrica (cálculo das quádricas), seguimos o fluxo do método, em que após a construção/atualização da octree, podemos retornar no ciclo pegando os dados do próximo quadro, que passou pela aquisição, registro da nuvem de pontos conectada e simplificação. Se a octree não for simplificada no meio do processo, a poligonização da superfície implícita em todas ou algumas etapas do ciclo não interefere com o resultado que seria obtido se fizéssemos apenas o processamento de cada quadro incrementalmente e deixássemos para extrair a malha somente ao final. No entanto, se a octree for simplificada em alguma iteração do ciclo, pode ocorrer divergências entre o resultado final obtido dessa maneira e o resultado que seria obtido continuamente.

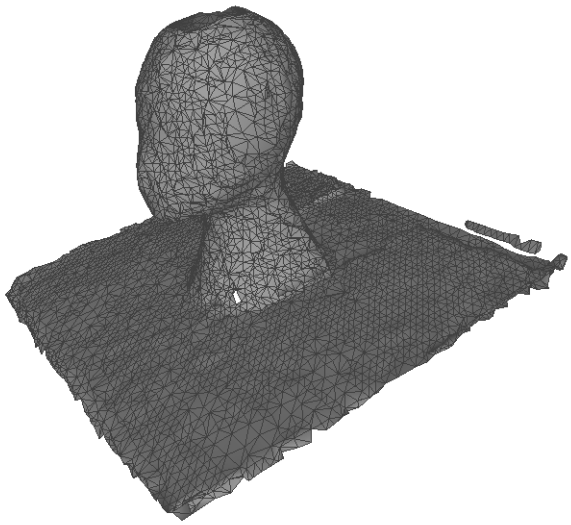
Os demais quadros seguem a mesma lógica para atualização. Ao final do 8° e último quadro, realizamos a poligonização da superfície de interesse. As Figuras 4.9, 4.10 e 4.11 apresentam o resultado obtido.

A finalidade do modelo obtido como resultado do processo pode ser bem diversa, desde o uso em renderizações à impressão 3D. As vezes, o usuário pode estar interessado na captura de um objeto específico sobre uma mesa, chão ou outra superfície plana dentro da cena. Por conta disso, pode-se trabalhar na remoção de elementos indesejáveis da cena capturada, visando à reconstrução de um único objeto. Em particular, para planos, há os trabalhos [43, 44] que propuseram técnicas de identificação de planos em imagens RGB-D que permitiriam a segmentação e remoção de estruturas planares. Para exemplificar os resultados dessa proposta, removemos o tabuleiro de xadrez da imagem e reconstruímos somente o modelo da manequin (Figura 4.12 e Figura 4.13).

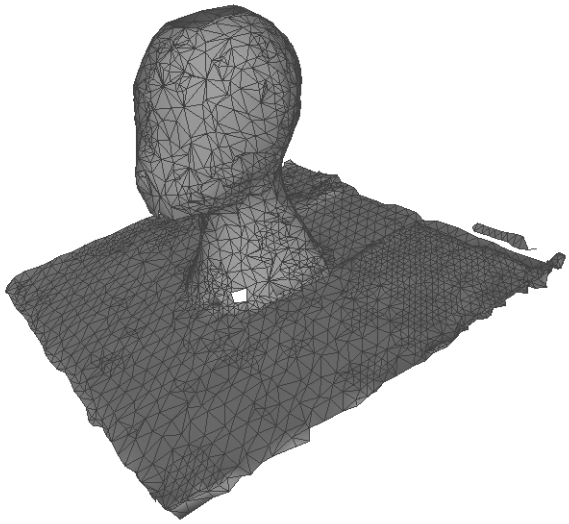
Embora o ruído não comprometa muito a estrutura topológica da malha reconstruída, é perceptivel o seu efeito sobre a geometria da malha. Observando-se as figuras de 4.8 a 4.13, vemos que a forma do manequin reconstruído não é tão suave quanto à forma observável nas imagens capturadas. No plano do tabuleiro é possível ver pequenas rugosidades devido ao ruído além de um grande desnível por conta de imprecisões no registro das malhas em relação ao mesmo referencial.



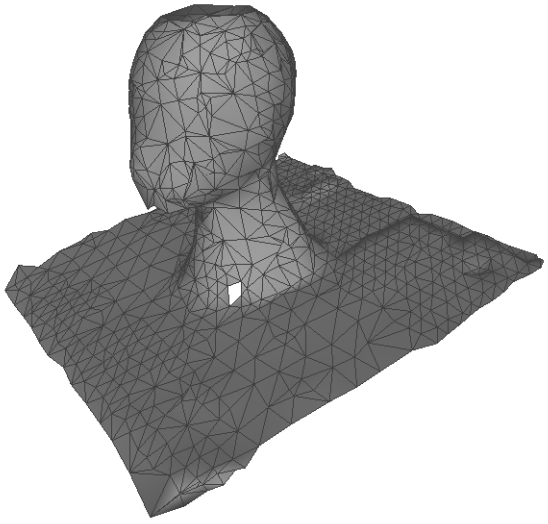
(a) Sem simplificação



(b) Sem simplificação

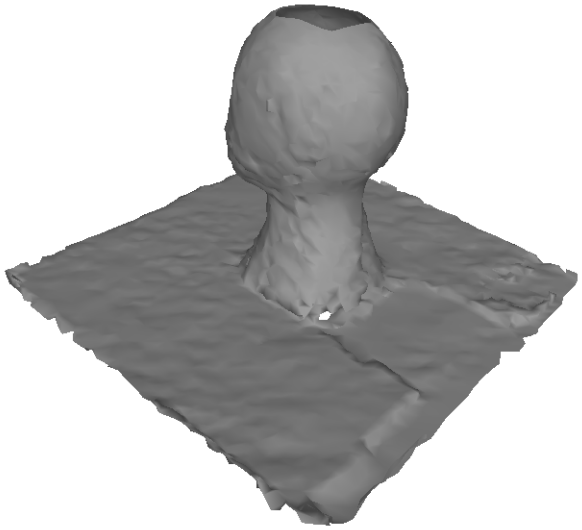


(c) Simplificação com erro de 1/100

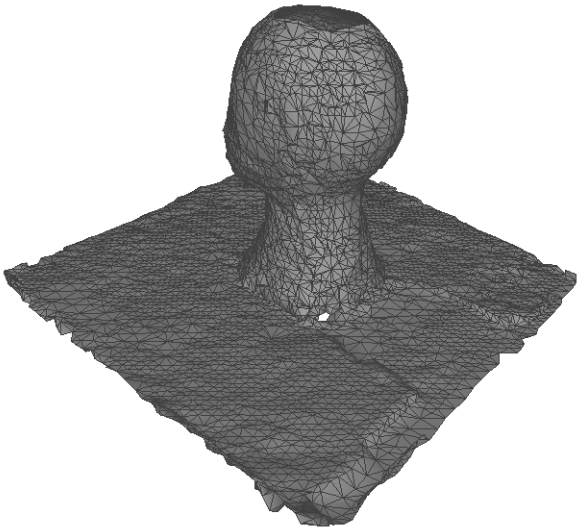


(d) Simplificação com erro de 1/10

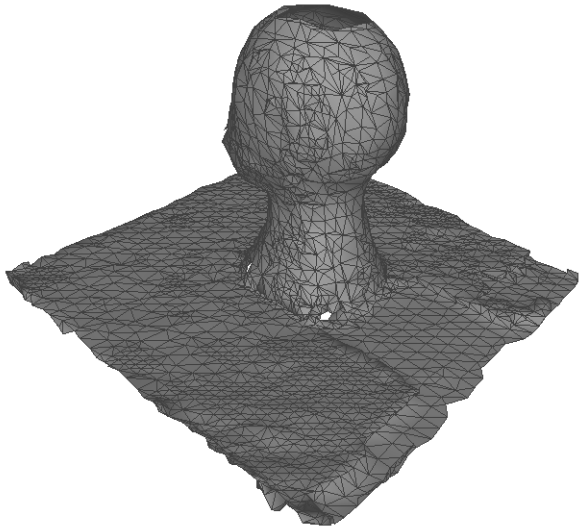
Figura 4.9: Reconstrução utilizando os 8 quadros (vista 1)



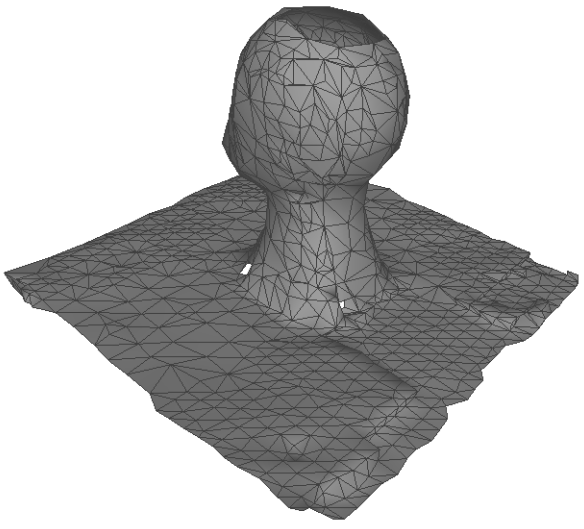
(a) Sem simplificação



(b) Sem simplificação

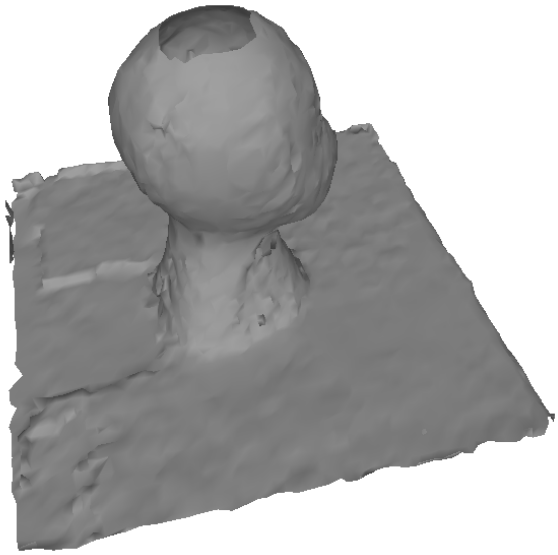


(c) Simplificação com erro de 1/100

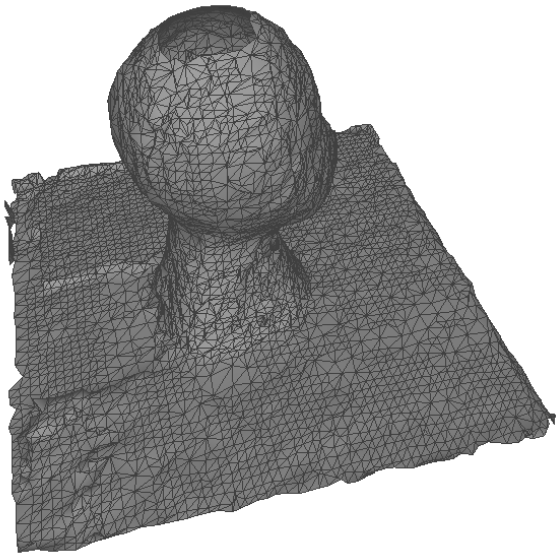


(d) Simplificação com erro de 1/10

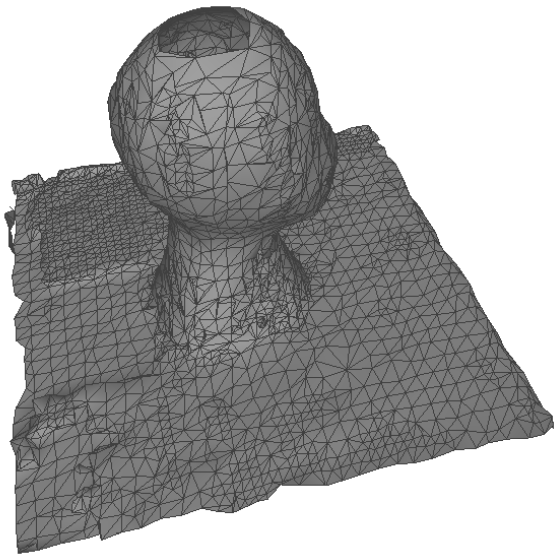
Figura 4.10: Reconstrução utilizando os 8 quadros (vista 2)



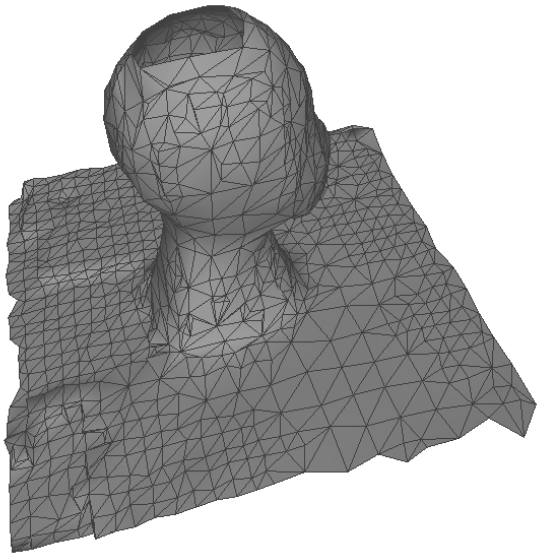
(a) Sem simplificação



(b) Sem simplificação



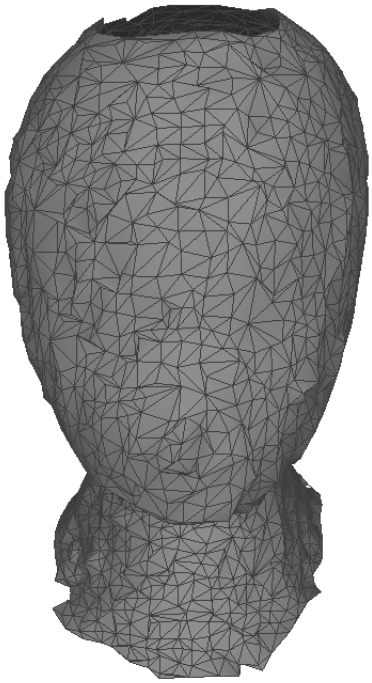
(c) Simplificação com erro de 1/100



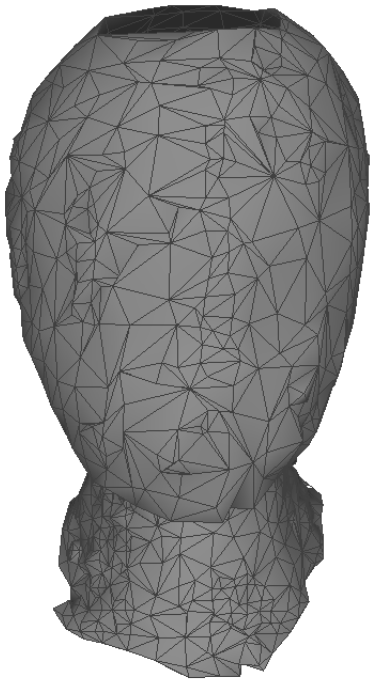
(d) Simplificação com erro de 1/10

Figura 4.11: Reconstrução utilizando os 8 quadros (vista 3)

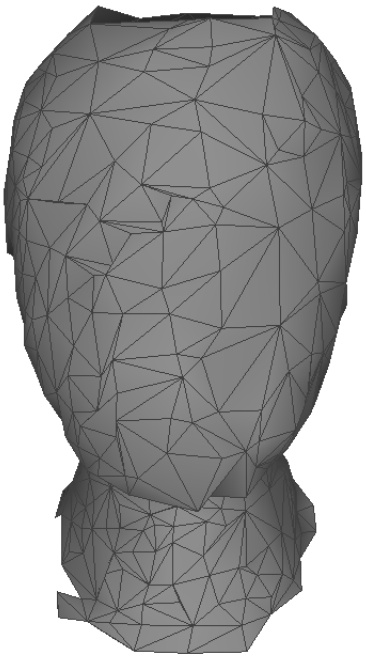
(a) Sem simplificação



(b) Sem simplificação

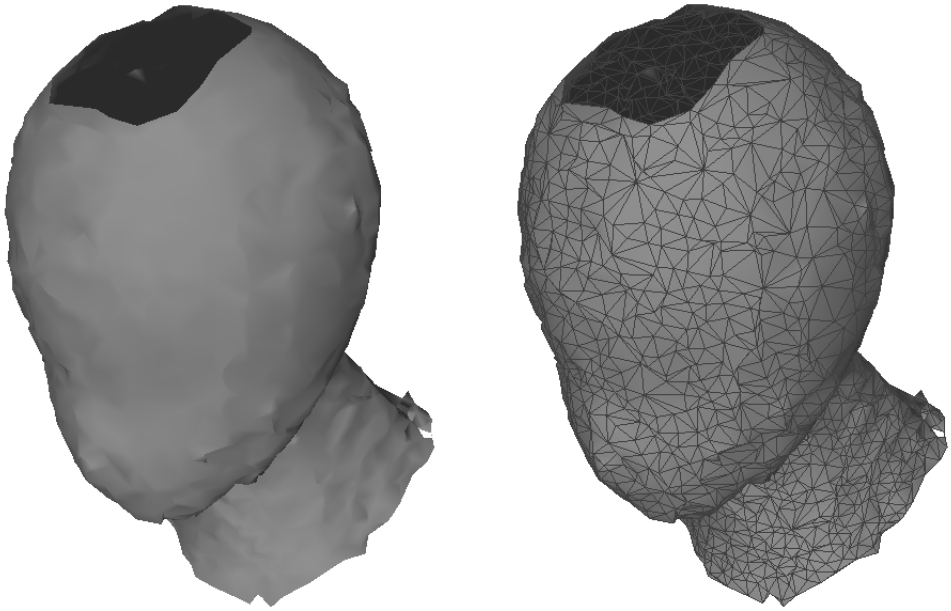


(c) Simplificação com erro de 1/100



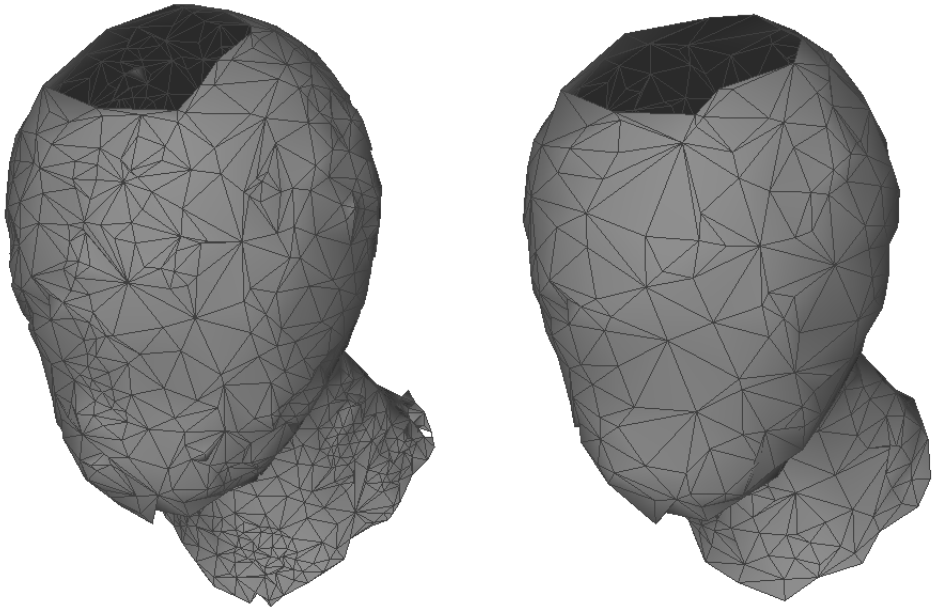
(d) Simplificação com erro de 1/10

Figura 4.12: Reconstrução sem tabuleiro de xadrez (vista 1)



(a) Sem simplificação

(b) Sem simplificação



(c) Simplificação com erro de 1/100

(d) Simplificação com erro de 1/10

Figura 4.13: Reconstrução sem tabuleiro de xadrez (vista 2)

5 CONCLUSÃO E TRABALHOS FUTUROS

O surgimento e a popularização de câmeras RGB-D ao ponto de serem utilizadas fora de ambientes controlados como a academia e a indústria aumentam a importância da existência de métodos eficazes, robustos e eficientes para soluções de mapeamento e interação com o ambiente que nos cerca. Os experimentos realizados nos guiaram para a concepção de um pipeline de reconstrução que fizesse bom uso das tecnologias existentes e adaptações de soluções propostas para problemas próximos.

Há diversos experimentos encorajadores que não puderam ser realizados dentro do prazo deste trabalho, mas que seriam relevantes para se melhorar a qualidade da malha reconstruída e robustez do método. Por exemplo, embora os experimentos tenham mostrado que o cálculo das quádricas a partir das interseções das malhas com as arestas da grade de amostragem não gere os melhores resultados em termos de qualidade da malha, não pudemos investigar a razão por trás disso. Verificamos que o ruído de aquisição, mesmo após a etapa de simplificação, impacta consideravelmente a qualidade da malha final produzida, porém também não pudemos avaliar as consequências de se adicionar uma etapa de filtragem dos mapas de profundidade ao método.

O método de dual contouring [31] utilizado para poligonizar a superfície a partir da octree construída foi projetado para malhas fechadas, isto é, malhas que representem superfícies sem bordo. Se poligonizarmos uma superfície a partir de uma única imagem de profundidade, naturalmente teremos bordas. Se conseguirmos imagens que cubram um objeto por completo, podemos obter uma configuração de octree que será poligonizada em uma malha fechada. Contudo, em diversas situações, é possível que ainda tenhamos bordo e, nesses casos, podemos observar imperfeições na reconstrução próximo a regiões de bordo. Uma maneira de minimizar este problema é adicionar às quádricas de células de bordo um termo de correção que leve este fato em consideração, por exemplo, inserindo planos ortogonais às bordas como em [45]. Uma estratégia semelhante poderia ser adotada em células próximo de regiões de alta curvatura, mas as características e impactos do termo a ser adicionado teriam que ser estudadas.

Para melhor formalização do método, seria adequado compará-lo com outras abordagens de solução existentes de modo a estabelecer uma base com critérios objetivos e identificar os pontos fortes e fracos de cada um. Uma comparação em termos de velocidade de execução e qualidade da malha produzida entre os critérios que utilizamos para o cálculo das quádricas a partir das malhas e uma abordagem baseada na amostragem de funções de distância com sinal seria bastante válida.

Uma proposta de trabalho futuro decorrente de nossos experimentos é o desenvolvimento da prova de conceito apresentada em sua forma polida, integrando todos os passos do pipeline em uma implementação para dispositivo móvel que utilize bem os recursos dis-

poníveis para interação em tempo real. A integração de métodos de detecção de estruturas planares como o [43] e outras formas geométricas notáveis, possibilitaria a segmentação e reconstrução de objetos pelos métodos mais adequados a cada forma. Esta pode ser uma linha de pesquisa em busca de uma representação intermediária que retrate esta segmentação.

Neste trabalho, nos restringimos à reconstrução de topologia e geometria da superfície, utilizando apenas a informação de profundidade das imagens RGB-D. Como trabalho futuro, propõe-se o estudo da reconstrução da informação de cor e textura sobre o modelo usando a informação RGB das imagens.

Por fim, aproveitando-se as capacidades dos dispositivos móveis de conexão a redes de computadores, sugere-se o estudo de como constituir uma arquitetura eficiente que utilize recursos remotos para armazenar e processar dados para a reconstrução. É possível, por exemplo, guardar os dados geométricos junto com metadados de geolocalização, permitindo a digitalização de estabelecimentos completos, tais como uma escola, a partir da captura de imagens de cada sala e agregação dos dados no servidor.

6 REFERÊNCIAS BIBLIOGRÁFICAS

[1] Marc Levoy, Kari Pulli, Brian Curless, Szymon Rusinkiewicz, David Koller, Lucas Pereira, Matt Ginzton, Sean Anderson, James Davis, Jeremy Ginsberg, Jonathan Shade, and Duane Fulk. **The Digital Michelangelo Project: 3D Scanning of Large Statues**. SIGGRAPH 2000, New Orleans, LA, 24-28 July, 2000.

[2] A. S. Huang, A. Bachrach, P. Henry, M. Krainin, D. Maturana, D. Fox, N. Roy. **Visual odometry and mapping for autonomous flight using an RGB-D camera**. Proc. Int. Symp. Robot. Res., pp. 1-16, 2011.

[3] Henry, P.; Krainin, M.; Herbst, E.; Ren, X.; Fox, D. **RGB-D mapping: using depth cameras for dense 3D modeling of indoor environments**. In: Khatib, O., Kumar, V., Sukhatme, G. (eds.) Experimental Robotics, pp. 477–491. Springer, Heidelberg (2014).

[4] Christian Feinen, Joanna Czajkowska, Marcin Grzegorzek, Longin Jan Latecki. **Matching of 3D Objects Based on 3D Curves**. Computer Vision and Machine Learning with RGB-D Sensors, 2014, pp 137-155.

[5] Jungong Han, Junwei Han. **RGB-D Human Identification and Tracking in a Smart Environment.** Computer Vision and Machine Learning with RGB-D Sensors, 2014, pp 137-155.

[6] **Structure Sensor Documentation**. [internet] Disponível em: <http://structure.io/developers>. Acesso em 15 de janeiro de 2016.

[7] **Project Tango**. [internet] Disponível em: <https://developers.google.com/project-tango/developer-overview> . Acesso em 15 de janeiro de 2016.

[8] Paz, Hallison; Velho, Luiz. **Imagens RGB-D em plataformas móveis**. Relatório Técnico: Laboratório VISGRAF, Instituto de Matemática Pura e Aplicada, 2016.

[9] F. Tombari, L. Di Stefano, S. Giardino, **Online Learning for Automatic Segmentation of 3D Data**. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS '11), 2011 [RGB-D Semantic Segmentation Dataset - http://vision.deis.unibo.it/fede/kinectDataset.html]

[10] Kevin Lai; Liefeng Bo; Xiaofeng Ren; Dieter Fox. **A Large-Scale Hierarchical Multi-View RGB-D Object Dataset**. IEEE International Conference on Robotics and Automation (ICRA), May 2011.

[11] Quanshi Zhang; Xuan Song; Xiaowei Shao; Huijing Zhao; Ryosuke. **Category Modeling from just a Single Labeling: Use Depth Information to Guide the Learning of 2D Models**. Shibasaki Proc. of IEEE International Conference on Computer Vision and Pattern Rcognition (CVPR), 2013.

[12] Firman, Michael. **List of RGB-D Datasets**. [Internet] Disponível em <http://www0.cs.ucl.ac.uk/staff/M.Firman/RGBDdatasets/>. Acesso em 26 de fevereiro de 2017.

[13] Choi, Sungjoon; Zhou, Qian-Yi; Koltun, Vladlen. **Robust Reconstruction of Indoor Scenes**. The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2015.

[14] Antonio W. Vieira; Armando A. Neto; Douglas G. Macharet; Mario F. M. Campos. **Mesh Denoising Using Quadric Error Metric**. Universidade Estadual de Montes Claros Montes.

[15] Shahram Izadi, David Kim, Otmar Hilliges, David Molyneaux, Richard Newcombe, Pushmeet Kohli, Jamie Shotton, Steve Hodges, Dustin Freeman, Andrew Davison, and Andrew Fitzgibbon, **KinectFusion: Real-time 3D Reconstruction and Interaction Using a Moving Depth Camera**, ACM Symposium on User Interface Software and Technology, October 2011.

[16] Richard A. Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J. Davison, Pushmeet Kohli, Jamie Shotton, Steve Hodges, and Andrew Fitzgibbon, **KinectFusion: Real-Time Dense Surface Mapping and Tracking**, in IEEE ISMAR, IEEE, October 2011.

[17] Ben Glocker; Shahram Izadi; Jamie Shotton; Antonio Criminisi. **Real-Time RGB-D Camera Relocalization**. International Symposium on Mixed and Augmented Reality (ISMAR), IEEE, October 1, 2013.

[18] Jamie Shotton; Ben Glocker; Christopher Zach; Shahram Izadi; Antonio Criminisi; Andrew Fitzgibbon. **Scene Coordinate Regression Forests for Camera Relocalization in RGB-D Images**. in Proc. Computer Vision and Pattern Recognition (CVPR), IEEE, June 1, 2013.

[19] Argawal, P.K., Suri, S.: **Surface approximation and geometric partitions**. SIAM Journal on Computing 27(4), 1016–1035 (1998).

[20] Garland, M., & Shaffer, E. (2002). **A multiphase approach to efficient surface simplification**. IEEE Visualization, 2002. VIS 2002., 117–124.

[21] R. Straub. **Exact Computation of the Hausdorff Distance between Triangular Meshes**. Proceedings of Eurographics 2007, pp. 17-20, 2007.

[22] Cignoni P., Rocchini C., Scopigno R.: **Metro: Measuring error on simplified surfaces**. Computer Graphics Forum 17, 2 (1998), 167–174.

[23] Guthe M., Borodin P., Klein R.: **Fast and accurate Hausdorff distance calculation between meshes**. In Proceedings of the 13th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision (WSCG) (2005), pp. 41–48.

[24] Garland, M., & Heckbert, P. S. (1997). **Surface simplification using quadric error metrics**. Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques - SIGGRAPH '97, 209–216.

[25] Velho, L. (2001). **Mesh simplification using four-face clusters**. Proceedings - International Conference on Shape Modeling and Applications, SMI 2001, 200–208.

[26] Wenger, Rephael. **Isosurfaces: Geometry, Topology & Algorithms**. A.K. Peters/CRC Press, 2013. (Capítulos 2 e 8)

[27] Jane Wilhelms , Allen Van Gelder. **Octrees for faster isosurface generation**. ACM Transactions on Graphics (TOG), v.11 n.3, p.201-227, July 1992 [doi>10.1145/130881.130882].

[28] B. Curless and M. Levoy. **A volumetric method for building complex models from range images**. ACM Trans. Graph., 1996.

[29] Lorensen, W. E.; Cline, Harvey E. (1987). "**Marching cubes: A high resolution 3d surface construction algorithm**". ACM Computer Graphics 21 (4): 163–169.

[30] Gibson, S. F. F. (1998a). **Constrained elastic surface nets: Generating smooth surfaces from binary segmented data**. In Proceedings of the First International Conference on Medical Image Computing and Computer-Assisted Intervention, MICCAI 1998, pages 888–898, Berlin. Springer-Verlag.

[31] Ju, T., Losasso, F., Schaefer, S., & Warren, J. (2002). **Dual contouring of Hermite data**. ACM Transactions on Graphics, 21(3), 339–346.

[32] Schaefer, Scott; Warren, Joe. **Dual Contouring:" The Secret Sauce"**. Citeseer, 5 (2002). Retrieved from http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.13.2631

[33] Achuta. Kadambi; Ayush. Bhandari; Ramesh Raskar, L. Shao, J. Han, P. Kohli, Z. Zhang, **3D depth cameras in vision: Benefits and limitations of the hardware**. In Computer Vision and Machine Learning With RGB-D Sensors, Berlin, Germany:Springer-Verlag, pp. 3-26, 2014.

[34] Paz, Hallison; Velho, Luiz. **Adaptive Polygonization Methods for RGB-D Images**. Relatório Técnico: Laboratório VISGRAF, Instituto de Matemática Pura e Aplicada, 2016.

[35] Tao Ju. **Dual Contouring Implementation in C++ by Tao Ju**. [internet] Disponível em: <https://github.com/aewallin/dualcontouring>. Acesso em 5 de julho de 2016.

[36] Nicholas Gildea. **Dual Contouring Implementation by Nicholas Gildea**. [internet] <https://github.com/nickgildea/DualContouringSample> . Acesso em 5 de julho de 2016.

[37] **Experiments with dual contouring**. [internet] <https://github.com/hallpaz/dual_contouring_experiments>. Acesso em 6 de março de 2017.

[38] **MeshLab**. [Internet] Disponível em: <http://meshlab.sourceforge.net/> . Acessado em 5 de julho de 2016.

[39] VELHO, Luiz Carlos; GOMES, Jonas de Miranda; FIGUEIREDO, Luiz Henrique de. **Implicit objects in computer graphics**. New York: Springer, 2002. (Capítulo 10)

[40] Sungjoon Choi and Qian-Yi Zhou and Vladlen Koltun. **Robust Reconstruction of Indoor Scenes**. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015.

[41] Zang, Aldo; Lucio, Djalma; Velho, Luiz. **QTCalib - Tsai Calibration Tool**. Disponível em <http://w3.impa.br/~zang/qtcalib/>. Acesso em 11 de Fevereiro de 2017.

[42] Tsai, Roger Y. **An Efficient and Accurate Camera Calibration Technique for 3D Machine Vision**. Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, Miami Beach, FL, pp. 364-374, 1986.

[43] Sousa, Eduardo Vera. **D-KHT: Detecção Em Tempo Real De Planos Em Imagens De Profundidade**. Master Thesis: Universidade Federal Fluminense, 2016.

[44] Silva, Djalma L. **Uso de Estruturas Planares Extraídas de Imagens RGB-D em Aplicações de Realidade Aumentada**. Master Thesis: Pontífica Universidade Católica do Rio de Janeiro, 2016.

[45] Lindstrom, P. ; Silva C. **A memory insensitive technique for large model simplification**. Proceedings of IEEE Visualization 2001, 2001.