Laboratório VISGRAF
Instituto de Matemática Pura e Aplicada

Imagens RGB-D em plataformas moveis

Hallison Paz
Luiz Velho (orientador)

Technical Report TR-16-02 Relatório Técnico

March - 2016 - Março

1 Introdução

Este relatório tem por finalidade documentar o curso de leitura *Imagens RGB-D em plataformas móveis,* que teve como objetivos o estudo das características de três das principais plataformas atuais de captura de imagens RGB-D para dispositivos móveis, a ZED câmera, o Google Project Tango e o Structure Sensor; o estudo de métodos de poligonização de superfícies implícitas, e a realização de experimentos de reconstrução de malhas tridimensionais.

2 Plataformas de captura de imagens RGB-D para aplicações móveis

2.1 Structure Sensor

O Structure Sensor é um sensor de captura de profundidade baseado em luz estruturada, ou seja o aparelho possui um emissor de luz em frequência infravermelho, que emite um padrão específico, conhecido pelo sistema, e um receptor de infravermelho que captura captura a cena e identifica distorções no padrão, permitindo-se que seja computado um mapa de profundidade. Este tipo de técnica para estimação de profundidade faz parte do que se conhece como estéreo ativo, pois é necessário empregar (ativamente) informação extra na cena. Esta tecnologia é similar à que foi utilizada no Kinect da 1ª geração.

Em uma estratégia de otimização de uso para plataformas iOS, a Occipital — fabricante do sensor — desenvolveu também uma estrutura que permite o acoplamento do sensor a um iPad (figura 1). Desta forma, pode-se trabalhar aplicações com imagens RGB-D em que a informação de radiância (RGB) provém da câmera nativa do iPad, enquanto a dimensão D (depth) é capturada pelo Structure Sensor. Para funcionar adequadamente, é necessário realizar uma calibração para registrar esses dados em um mesmo referencial.

Figura 1: Structure Sensor acoplado a um iPad

Há uma grande ênfase em mobilidade. Sendo projetado para funcionar em um cenário móvel e com aplicações que devem responder ao usuário em tempo real, o sensor sacrifica a precisão da informação em prol da rapidez de resposta. Por conta disso, o mapa de profundidade obtido normalmente apresenta bastante ruído e regiões onde não foi possível realizar uma estimação de profundidade. O alcance do sensor encontra-se entre 40 centímetros e 3,5 metros, o que permite tanto a captura de objetos quanto de cenas inteiras em ambientes limitados [4]. Objetos mais próximos que o limite mínimo e mais afastados que o limite máximo podem não ser capturados, gerando mais regiões sem leitura no mapa de profundidade.

Embora seja otimizado para uso com dispositivos iOS, principalmente iPads (devido ao tamanho), também é possível utilizar o Structure Sensor ligado diretamente em um computador ou até

mesmo em um dispositivo Android. Neste caso, deve-se recorrer a APIs de terceiros como o OpenNI, pois o Structure SDK é disponível apenas para iOS.

A tecnologia de luz estruturada permite que o Structure Sensor capture dados mesmo em situações de escuridão total, devido à radiação infravermelho lançada sobre a cena. No entanto, em contextos onde haja intensa luz solar como, por exemplo, ambientes ao ar livre durante o dia, haverá uma grande quantidade de ruído nos dados devido à interferência da radiação da luz solar. Este fato, aliado ao alcance relativamente pequeno do sensor fazem com que ele seja mais adequado a aplicações em ambientes fechados.

2.2 Google Project Tango

Ao contrário do Structure Sensor, que consiste em um dispositivo periférico para a captura de profundidade, que deve ser integrado à câmera de um dispositivo principal, a proposta do Project Tango é ser uma plataforma móvel com a captura de profundidade integrada com os demais recursos. A proposta da plataforma não é ser apenas um meio de captura de mapas de profundidade, mas sim ser um meio de prover às aplicações móveis a capacidade de lidar com informações sobre o espaço que cerca o usuário. A princípio, a plataforma Tango parece ser mais focada em conceitos de alto nível, com ênfase na combinação entre dados de tracking e profundidade para a geração de dados posicionais do dispositivo. No entanto, é possível adquirir os dados brutos do dispositivo para que sejam trabalhados em outras aplicações.

Figura 2: Project Tango development kit tablet

Dentro desta proposta, destacam-se três tipos de funcionalidades:

1. *Motion Tracking*: consiste na combinação de dados dos sensores de atitude (acelerômetro e giroscópio) com dados a partir da identificação de pontos característicos no ambiente, adquiridos por meio de uma câmera do tipo fisheye. Com motion tracking, é possível capturar o movimento do dispositivo em 6 graus de liberdade (3 de orientação e 3 de translação).

2. *Area Learning*: consiste no armazenamento de dados sobre o ambiente e o movimento do dispositivo, que podem ser compartilhados com outros usuários e melhorados pela adição de metadados.

3. *Depth perception*: consiste na detecção de superfícies, medição de distâncias e comprimentos dentro da cena.

No que concerne à captura RGB-D, a plataforma Tango, assim como o Structure Sensor, se baseia em luz estruturada para estimar um mapa de profundidade, apresentando um alcance entre 50 centímetros e 4 metros [5] .

2.3 ZED Câmera

Ao contrário do Structure Sensor e do Project Tango, a Zed câmera utiliza uma tecnologia de estéreo ativo, isto é, a informação de profundidade é recuperada sem a necessidade de projetar elementos adicionais sobre a cena. Basicamente, um par de imagens é capturado, com uma pequena distância entre eles, e seus dados são processados pelo sistema de modo a obter uma estimativa de profundidade por triangulação.

O processamento dos dados da Zed câmera não ocorrem no dispositivo em si, mas sim em uma máquina usada como servidor, pois o processo de estimar profundidade por meio de estéreo passivo é computacionalmente intensivo, requerindo um hardware com uma capacidade de processamento relativamente alta para que se possa operar em tempo real.

Figura 3: Zed Camera

Por não utilizar luz estruturada, a Zed câmera não sofre interferência de radiação infravermelho como os demais dispositivos analisados, sendo uma câmera que se propõe a ser utilizada tanto em ambientes indoor quanto outdoor. Seu alcance varia entre 1 metro e 20 metros, com captura de imagens em alta resolução.

3 Reconstrução de Superfícies

Uma aplicação bastante comum no uso de dados adquiridos por sensores de profundidade é a reconstrução de modelos geométricos. Estes modelos podem, posteriormente, serem utilizados em visualizações e interações em contextos diversos. Utilizaremos as imagens do dataset disponibilizado por [14] spara exemplificar este assunto.

3.1 Nuvem de Pontos

O tipo de reconstrução de informação tridimensional mais simples que podemos obter a partir de um mapa de profundidade é uma nuvem de pontos. Uma nuvem de pontos é um conjunto de pontos definidos em um sistema de coordenadas [6]. Conhecendo-se parâmetros intrínsecos da câmera, podemos construir uma nuvem de pontos em um referencial próprio a partir de um mapa de profundidade. Sejam u_0 e v_0 as coordenadas do centro da imagem, f_x e f_y a distância focal e γ o parâmetro que representa o desvio entre as colunas e linhas em relação ao ângulo reto, temos que:

$$\begin{bmatrix} f_x & \gamma & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} u \\ v \\ z_p \end{bmatrix} \tag{1}$$

Em que (x, y, z) são as coordenadas do ponto no espaço da cena, (u, v), no espaço da imagem e z_p, o valor de profundidade medido na posição (u, v). Considerando o parâmetro γ como nulo, isto é, considerando que as linhas e colunas da matriz do dispositivo de captura são perfeitamente ortogonais, temos que:

$$f_x * x + u_0 * z = u \tag{2}$$

$$f_y * y + v_0 * z = v \tag{3}$$

$$z = z_p \tag{4}$$

Com isso, podemos recuperar as coordenadas (x, y, z) no espaço da cena a partir das coordenadas (u, v) no espaço da imagem e do valor z_p correspondente ao pixel (u, v). Um mapa de profundidade com valores pixel a pixel como o que é retornado pelo Structure Sensor possibilita a construção de uma nuvem de pontos densa que, quando visualizada, nos permite ter uma impressão geral da cena capturada (figura 4). Evidentemente, havendo pixels sem informação de profundidade, verificaremos buracos nesta representação.

(a) vista afastada

(b) vista aproximada

Figura 4: Nuvem de pontos

Ainda com um mapa de profundidade dado por informações de profundidade pixel a pixel, podemos construir 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. Uma nuvem de pontos organizada é uma estrutura de dados mais poderosa na medida em que facilita algumas operações de visão computacional como, por exemplo, o registro.

3.2 Malhas 3D

A partir de uma nuvem de pontos, podemos reconstruir uma malha poligonal. O tipo mais

comum de malha poligonal para se trabalhar é uma malha triangular. Há diversos métodos diferentes para se calcular uma malha poligonal a partir de uma nuvem de pontos, podendo partir de uma abordagem paramétrica ou implícta. O trabalho [7] discute alguns destes métodos.

Como possuímos uma nuvem de pontos organizada, podemos adotar uma estratégia bastante simples, embora ineficiente em termos de armazenamento, para se obter uma malha poligonal como ponto de partida, utilizando a ordem das linhas e colunas da imagem de profundidade. Esta estratégia consiste em, primeiramente, delimitar um limiar de distância como tamanho máximo aceitável para as arestas dos triângulos e, então, percorrer a imagem de profundidade 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. A figura 5 ilustra o resultado desta abordagem sobre a imagem de teste.

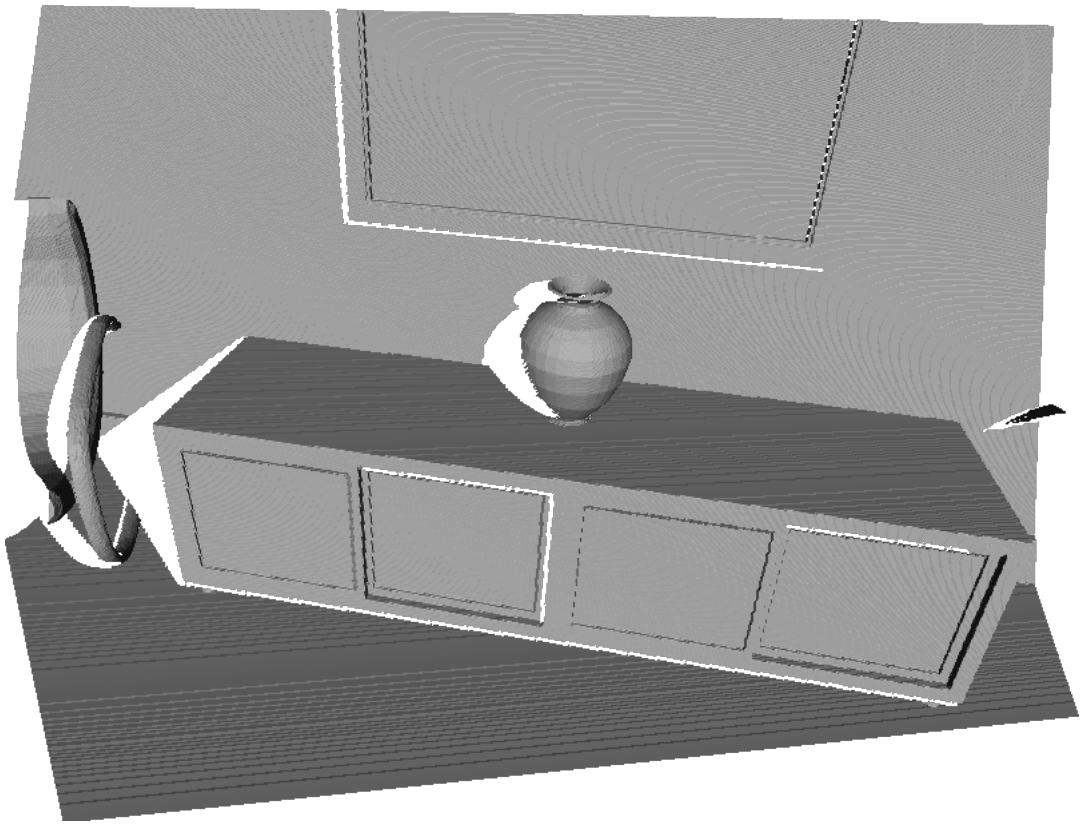


Figura 5: Malha poligonal calculada diretamente a partir de uma nuvem de pontos ordenada

3.3 Poligonização de Superfícies Implícitas

Dado 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 [12]. Sendo σ um valor regular da função ϕ, temos que o conjunto de nível σ de ϕ é dado por $\phi^{-1}(\sigma)$; quando $d = 3$, podemos chamr este conjunto de superfície de nível. Se a função ϕ for contínua, 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.

Um dos métodos de se reconstruir uma superfície definida implicitamente por um campo escalar consiste em subdivididir sistematicamente o espaço que contém a superfície em células, amostrar valores da função implícita nos vértices de cada célula e, então, procurar por arestas em que um dos vértices possua valor superior ao nível da superfície e o outro, inferior. Desta forma, sabemos que um

vértice deve estar no interior da superfície e o outro, no exterior; portanto, a superfície cruza a aresta bipolar.

Um algoritmo bastante clássico para realizar esse processo de poligonização de superfícies implícitas é o Marching Cubes [10]. Este algoritmo recebe como entrada uma grade que subdivide o espaço em cubos. Esta grade pode ser dividida de maneira uniforme ou não. Então, percorre-se cada cubo à procura de arestas bipolares[12], isto é, arestas em que um dos vértices possui valor superior ao nível da superfície e o outro, inferior. Baseado na quantidade de arestas intersectadas e na posição de cada aresta, o marching cubes determina uma estrutura poligonal que aproxima a superfície dentro da célula utilizando uma tabela que mapeia todas as 256 possibilidades de interseção. Explorando as simetrias das configurações de interseções, Lorensen e Cline concluíram que bastava representar 15 configurações na tabela (figura 6). Contudo, observou-se que com apenas 15 possibilidades era possível cair em casos de ambiguidade que podiam ser resolvidos com 7 configurações adicionais, totalizando 22 [12].

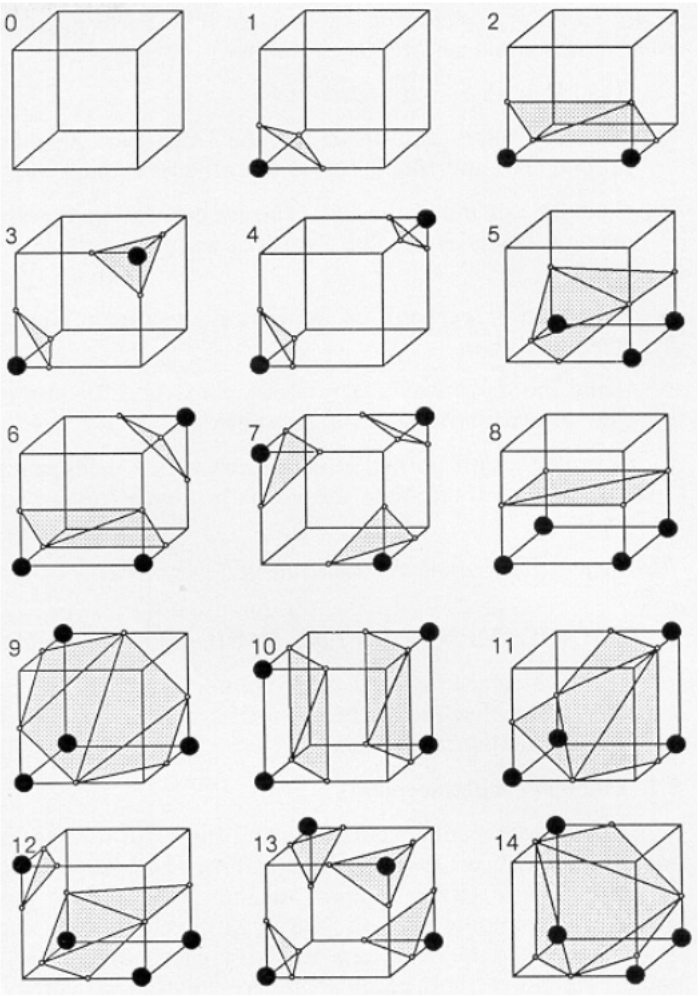


Figura 6: Possíveis configurações de poligonização para uma célula[10]

Com um processo bem definido de poligonização de uma superfície de nível, vamos pensar em quais pontos devemos amostrar a função e em como determinar uma função implícita para ser amostrada. A forma mais simples de se escolher os os pontos onde iremos amostrar o valor da função é optar por uma grade regular de cubos, representação que o marching cubes lida muito bem. É possível, também subdividir o espaço em tetraedros e aplicar uma variante do marching cubes, o marching tetrahedra, para a extração da superfície de nível. Alternativamente, pode-se optar por uma abordagem adaptativa

em que o espaço pode ser divido de forma bastante irregular de acordo com algum critério, utilizando estruturas de dados mais avançadas como, por exemplo, uma octree. Essa escolha tem impacto sobre a eficiência computacional do método e também sobre o nível de detalhe com que conseguiremos representar a aproximação da superfície. Em todo caso, o espaço a ser subdividido tem que ser limitado.

Quanto à obtenção da função implícita, há várias alternativas possíveis dependendo da aplicação. Se a superfície a ser reconstruída satisfaz, de fato, a alguma função conhecida, podemos simplesmente usar essa função e computar seus valores para os vértices da grade no espaço. Por exemplo, considere a superfície definida implicitamente pela função $((x^2 + y^2)^2 - x^2 + y^2)^2 + z^2 - 0.01 = 0$, podemos avaliar a função em cada vértice de uma grade regular e obtermos uma representação em lista de triângulos para uma superfície de nível (não degenerada) qualquer desta função. A figura 7 ilustra a isosuperfície de nível 0 obtida utilizando-se o algoritmo marching cubes.

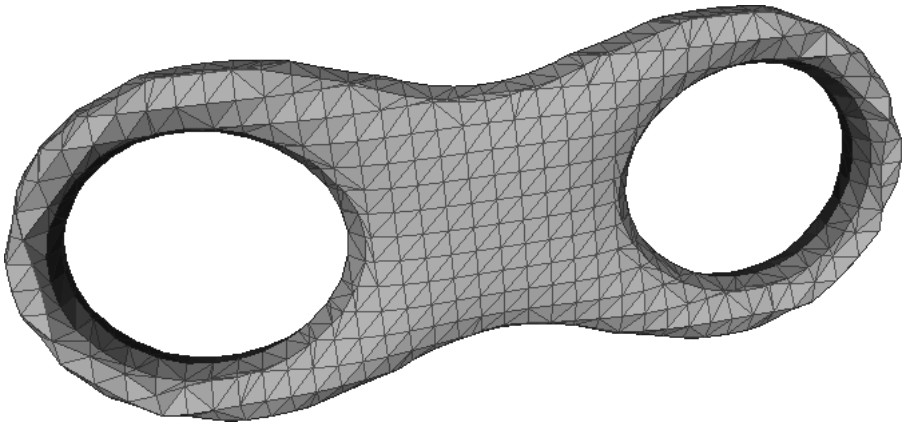


Figura 7: Bitoro

A forma analítica da função pode ser arbitrariamente complicada, gerando superfícies com formatos diversos. Um exemplo interessante é a superfície de nível zero dada implicitamente pela função $(x^2 + (9/4)y^2 + z^2 - 1)^3 - x^2z^3 - (9/80)y^2z^3 = 0$ (figura 8).

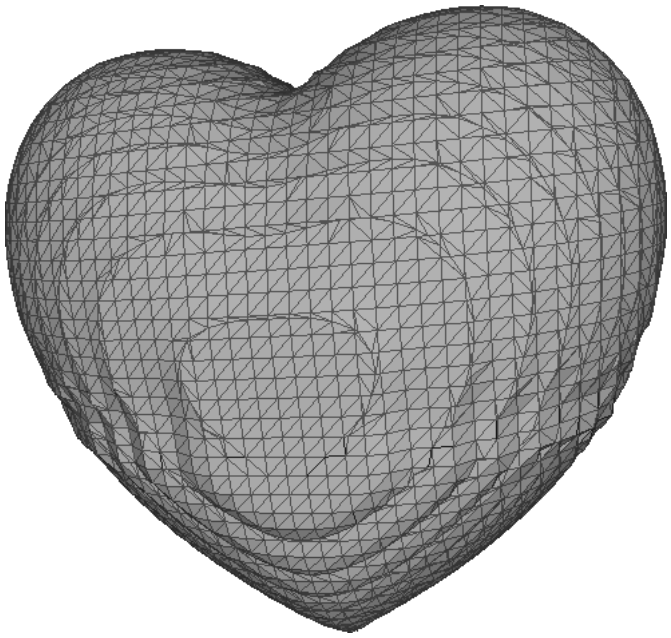


Figura 8: Superfície em forma de coração

No entanto, nem toda superfície pode ser definida por uma fórmula analítica como apresentamos. Muitas vezes estamos interessados em reconstruir superfícies em situações que apenas conhecemos amostras adquiridas por algum sensor como os mencionados na seção 2 deste relatório. Neste caso, é necessário formular alguma função que possa representar um modelo implícito para esta superfície. A solução pode passar pela união de vários modelos locais. Uma estratégia bastante adotada é a apresentada em [13], em que o autor sugere o cálculo de uma função de distância com sinal truncada. Considera-se a superfície como uma curva de nível 0 da função implícita que estamos procurando; então, para cada vértice da grade, calcula-se a (menor) distância entre este vértice e a superfície; caso o ponto esteja no interior da superfície, atribui-se um sinal positivo a essa distância, do contrário, ela é negativa (a escolha do lado positivo é arbitrária). O truncamento vem do fato de que a função não é calculada em todos os vértices da grade, mas apenas em uma vizinhança tubular da superfície.

3.4 Fusão de dados

Uma das aplicações que vem sendo bastante explorada no uso de sensores RGB-D é a de scanner 3D. Neste tipo de aplicação, objetiva-se reconstruir um modelo 3D de algum objeto ou até mesmo de uma cena inteira a partir da captura de várias imagens RGB-D de vários pontos de vista do objeto/cena.

Uma dificuldade inerente desta aplicação é 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 [8, 9, 13] adotam uma representação implícita da superfície e realizam poligonização da superfície por meio de algoritmos como o Marching Cubes ou uma de suas variantes.

A estratégia básica consiste em ter um bom sistema de tracking, que permita fazer o registro entre uma câmera e outra, e para cada vértice da grade de subdivisão do espaço, se o vértice já possuir algum valor de distância à superfície dado por algum ponto de vista, atualiza-se este valor com uma média que leve em consideração o novo ponto de vista. Esta média não precisa ser simples; pode-se, por exemplo, utilizar pesos baseados no ângulo entre a normal à superfície e a direção de observação, de modo a diminuir a influência de dados capturados de ângulos com menor visibilidade.

4 Experimento de reconstrução

Nesta seção, descreve-se um experimento de reconstrução de superfície por um método implícito utilizando o algoritmo marching cubes. O objetivo é partir da malha poligonal apresentada na figura 5 como dado de entrada, subdividir o espaço de maneira uniforme, definir e amostrar uma função implícita e, por fim, tentar obter uma superfície que se aproxime da superfície de entrada. O espaço da grade de amostragem foi limitado à caixa envolvente (*bounding box*) da superfície em questão, sendo dividido em células com arestas de mesmo comprimento.

4.1 Metodologia

Neste experimento, iremos contruir uma função que represente a distância com sinal entre os pontos da grade e a superfície que queremos reconstruir. Dessa forma, pontos que estiverem na parte exterior da superfície terão valores negativos, enquanto pontos no interior da superfície terão valores positivos. Este cálculo de distância, porém deve ser bastante cuidadoso, pois como veremos nos resultados, alguns detalhes podem levar a erros que geram resultados com falhas.

A primeira abordagem utilizada para cálculo desta distância consistiu em, para cada triângulo da face da malha de entrada, procurar todas as células que contém algum dos vértices ou que são

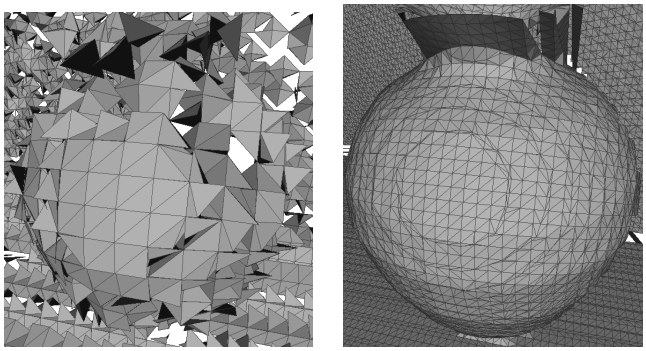
vizinhas de alguma célula que contém algum vértice do triângulo em questão e, então, calcular a interseção das arestas dessas células com o triângulo. No caso de haver interseção, computa-se o ponto de interseção e toma-se como valor do vértice da célula a distância entre esse vértice e o ponto de interseção. O sinal atribuído à distância é determinado pelo produto escalar entre a normal do triângulo intersectado e o segmento que une o vértice ao ponto de interseção. Como um mesmo vértice pode fazer parte de até três arestas diferentes, pode-se acabar calculando 3 valores diferentes para a distância com esta abordagem. A solução adotada para este problema foi realizar a média dos valores. Após a valoração dos vértices da grade, executou-se o algoritmo Marching Cubes sobre estes vértices para extrair uma superfície de nível.

A segunda abordagem utilizada para o cálculo de uma função implícita valeu-se do fato de estamos utilizando uma grade de amostragem uniforme para a definição de uma estratégia de ray casting a partir de um plano de referência. Como esta grade foi construída pela subdivisão da caixa envolvente da superfície (alinhada com os eixos de orientação do sistema de coordenadas), escolheu-se o plano que contêm a face com menor valor de coordenada Z, paralela ao plano XY, como o plano de referência. A partir de cada ponto da grade sobre esta face, lançamos um raio ortogonal ao plano de referência (paralelo ao eixo Z) no sentido da superfície. Nos casos em que este raio intersectou a superfície, calculamos a distância de cada ponto da grade ao longo do raio como a diferença entre a coordenada Z do ponto de interseção e a coordenada Z do próprio ponto. Desta forma, precisamos lançar raios apenas a partir dos pontos da grade sobre o plano de referência e o sinal da distância fica naturalmente determinado pelo resultado da subtração. Esta abordagem assemelha-se a uma projeção ortogonal da superfície no plano de projeção da câmera.

Nos casos em que o raio não intersectou a superfície, nenhum valor foi atribuído aos pontos da grade no caminho do raio. Por conta disso, realizou-se uma etapa de inicialização em que todos os pontos da grade receberam um "valor sentinela", de modo que durante a etapa de geração de polígonos, com o algoritmo *marching cubes*, apenas foram processadas as células cujos vértices não possuíam este valor sentinela.

4.2 Resultados

A figura 9 apresenta, lado a lado, um mesmo trecho da cena de teste reconstruído pelos dois métodos descritos na seção 4.1. Observando a figura 9a, vemos que a primeira abordagem utilizada não apresentou bons resultados, gerando várias falhas na malha. Este resultado decorre de uma formulação incorreta da função implícita computada. A interseção de mais de uma aresta da célula com a superfície influencia a estimação da função de distância neste ponto de uma forma diferente. A formulação correta do problema consiste em determinar quais valores deveriam ser atribuídos aos vértices das arestas intersectadas para que os pontos de interseção possam corresponder ao valor zero para uma interpolação linear. Como abordado em [11], a solução deste problema passa por um sistema linear.



(a) Primeira abordagem (b) Segunda abordagem

Figura 9: Reconstrução do vaso da cena

A segunda abordagem, porém, apresentou bons resultados, como pode ser visto nas figuras 9b e 10.

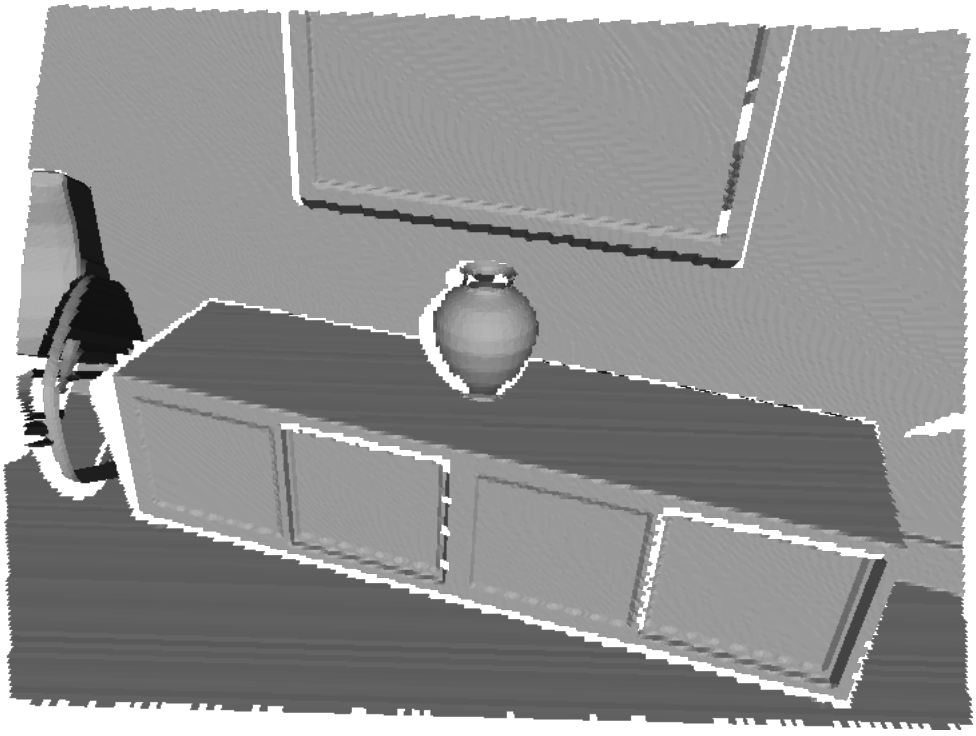


Figura 10: Reconstrução após ray casting ortogonal

Como um passo intermediário, após calcular as interseções entre os raios e a superfície, geramos também uma nuvem de pontos. Esta nuvem de pontos assemelha-se bastante à superfície original como mostra a figura 11.

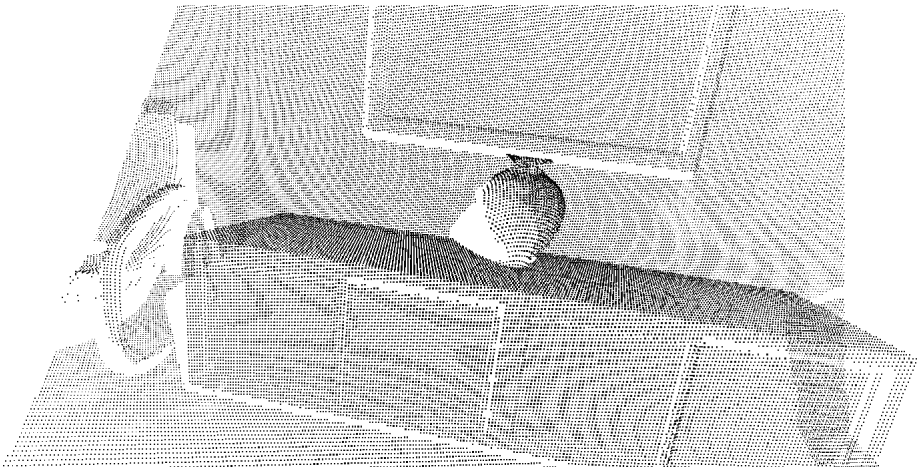


Figura 11: Nuvem de pontos obtida pela interseção de raios ortogonais ao plano de observação

A forma mais tradicional de tratar corretamente este problema de definição e amostragem da função implícita seria o lançamento de raios na cena a partir da posição do olho da câmera e a atribuição de valores aos vértices das células intersectadas pelo raio no caminho. No entanto, vimos que definir uma função de distância baseada em raios ao longo da direção de observação passando pelos vértices da grade apresentou bons resultados com um baixo custo computacional.

5 Conclusão

O surgimento de cada vez mais dispositivos de captura de informação de profundidade voltados para uso doméstico possibilita que sejam colocados em prática para usuários comuns aplicações que outrora eram restritas à academia e a projetos específicos na indústria. Hoje, muitas pessoas tiram fotografias comuns em seus smartphones, mas estamos caminhando em direção ao momento em que essas mesmas pessoas poderão capturar modelos tridimensionais e interagir com a cena por meio de interfaces e aplicações que explorem a geometria de seu entorno com a mesma facilidade. Com o poder computacional e os demais recursos disponíveis nos dispositvos de hoje em dia, é possível que essas interações sejam feitas em tempo real.

No experimento realizado, há vários pontos que podem ser melhorados. Primeiramente, a função de distância com sinal poderia ser calculada a partir de uma malha simplificada com bem menos polígonos. Em segundo lugar, há um grande espaço para otimizações no uso dos recursos computacionais, tanto em processamento quanto em memória. A saída obtida com o algoritmo Marching Cubes, por exemplo, consiste apenas de uma lista de triângulos (vértices e ordenação entre os vértices), com repetição de coordenadas de vértices que poderiam ter sido reaproveitados. Uma abordagem que tornaria este processo de reconstrução consideravelmente mais eficiente seria uma subdivisão adaptativa do espaço por meio de estruturas de dados mais apropriadas como, por exemplo, uma octree. Além disso, a utilização de algoritmos com soluções locais de poligonização da superfície, tal como o Marching Cubes, permite que a computação dos resultados seja realizada em paralelo com o auxílio da GPU, estratégia adotada em [8, 9], por exemplo. Essas otimizações são de extrema relevância em aplicações de interação em tempo real, cenário bastante comum no contexto de dispositivos móveis.

Referências

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

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

[3] **ZED Getting Started**. [internet] Disponível em: <https://www.stereolabs.com/developers/documentation/ZED_Developer_Guide.pdf>. Acesso em 15 de janeiro de 2016.

[4] **Structure Sensor Kickstarter Campaign**. [internet] Disponível em: <https://www.kickstarter.com/projects/occipital/structure-sensor-capture-the-world-in-3d/description>. Acesso em 7 de março de 2016.

[5] **Project Tango Depth Perception**. [internet] Disponível em: <https://developers.google.com/project-tango/overview/depth-perception>. Acesso em 7 de março de 2016.

[6] Shao, Ling Han, Jungong Kohli, Pushmeet Zhang, Zhengyou. **Computer Vision and Machine Learning with RGB-D Sensors.** Switzerland: Springer, 2014.

[7] Pawar, Navpreet Kaur; Macey, Jon **. Surface Reconstruction from Point Clouds**. Master Thesis: Bournemouth University, 2013.

[8] 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.

[9] 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.

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

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

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

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

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