JONG WAN SILVA

SISTEMA DE DIGITALIZAÇÃO 3D USANDO SUPER-RESOLUÇÃO EM IMAGENS RGBD

CURITIBA

2013

JONG WAN SILVA

SISTEMA DE DIGITALIZAÇÃO 3D USANDO SUPER-RESOLUÇÃO EM IMAGENS RGBD

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre. Programa de Pós-Graduação em Informática, Setor de Ciências Exatas, Universidade Federal do Paraná.
Orientador: Prof. Dr. Luciano Silva
Coorientadora: Profa. Dra. Olga R. P. Bellon

CURITIBA

2013

JONG WAN SILVA

SISTEMA DE DIGITALIZAÇÃO 3D USANDO SUPER-RESOLUÇÃO EM IMAGENS RGBD

Dissertação aprovada como requisito parcial à obtenção do grau de Mestre no Programa de Pós-Graduação em Informática da Universidade Federal do Paraná, pela Comissão formada pelos professores:

Orientador: Prof. Dr. Luciano Silva
Departamento de Informática, UFPR

Profª. Drª. Olga R. P. Bellon
Departamento de Informática, UFPR

Prof. Dr. André L. P. Guedes
Departamento de Informática, UFPR

Prof. Dr. Hugo Vieira Neto
Departamento de Eletrônica, UTFPR

Curitiba, 10 de setembro de 2013

Ministério da Educação
Universidade Federal do Paraná
Programa de Pós-Graduação em Informática

PARECER

Nós, abaixo assinados, membros da Banca Examinadora da defesa de Dissertação de Mestrado em Informática, do aluno Jong Wan Silva, avaliamos o trabalho intitulado, *"Sistema de digitalização 3D usando super-resolução em imagens RGBD"*, cuja defesa foi realizada no dia 10 de setembro de 2013, às 11:00 horas, no Departamento de Informática do Setor de Ciências Exatas da Universidade Federal do Paraná. Após a avaliação, decidimos pela (X)**aprovação** ( )**reprovação** do candidato.

Curitiba, 10 de setembro de 2013.

Prof. Dr. Luciano Silva
DINF/UFPR – Orientador

Prof. Dr. Hugo Vieira Neto
UTFPR – Membro Externo

Profa. Dra. Olga Regina Pereira Bellon
DINF/UFPR – Membro Interno

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DO PARANÁ
Setor de Ciências Exatas
Coordenação de Pós-Graduação em Informática

AGRADECIMENTOS

Gostaria de agradecer primeiramente a Deus por me abençoar com a sua presença e por estar sempre me guiando nesta longa jornada.

Agradeço aos professores Luciano Silva e Olga Regina Pereira Bellon pela orientação, apoio e oportunidade à pesquisa.

À CAPES pelo financiamento deste mestrado e pelo seu apoio à pesquisa no Brasil.

A todos os membros do grupo IMAGO pela amizade, idéias e disposição em ajudar.

Ao meu colega e amigo Leonardo Gomes que contribuiu na elaboração e desenvolvimento deste trabalho.

Aos amigos do grupo Hashi que tive a sorte de conhecer durante o mestrado.

E à minha família que sempre apoiou e incentivou a minha formação, e principalmente pelo amor e educação que sempre recebi.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE SIGLAS

3D	Tridimensional.
CPU	*Central Processing Unit.*
FPS	*Frames per second.*
GA	*Genetic algorithms.*
GPU	*Graphic Processing Unit.*
GPGPU	*General-Purpose Computing on Graphic Unit.*
ICP	*Iterative Closest Point.*
IR	*Infrared.*
IVIA	*IMAGO Volumetric Integration Algorithm.*
JBU	*Joint Bilateral Upsampling.*
MRF	*Markov Random Field.*
NAFDU	*Noise-Aware Filter for Depth Upsampling.*
PCL	*Point Cloud Library.*
POCS	*Projection Onto Convex Sets.*
RBC	*Random Ball Cover.*
RGB	Cor.
RGBD	Uma imagem de profundidade e outra colorida.
SDCF	*Slope Depth Compensation Filter.*
SDF	*Signed Distance Function.*
SIMD	*Single Instruction, Multiple Data.*
SURF	*Speeded Up Robust Features.*
TBB	*Threading Building Blocks.*
ToF	*Time-of-Flight.*
TSDF	*Truncated Signed Distance Function.*
VRIP	*Volumetric Range Image Processing.*
WJBF	*Weighted Joint Bilateral Filter.*

RESUMO

Com o advento de novos sensores de profundidade de baixo custo e com o aumento do poder de processamento paralelo das placas gráficas, houve um aumento significativo em pesquisas na área de reconstrução 3D em tempo real.

No grupo de pesquisa IMAGO, existe um sistema de reconstrução 3D para a preservação digital, adaptado aos *scanners* a laser de alta resolução. Visando aumentar a flexibilidade deste sistema, o objetivo deste trabalho é a ampliação do atual pipeline de reconstrução 3D do IMAGO para permitir a criação de modelos utilizando os novos sensores em tempo real. Outro objetivo é a aplicação de um método para o tratamento das imagens de baixa qualidade desses sensores, proporcionando modelos reconstruídos a partir das novas imagens de melhor resolução.

A principal meta da preservação digital é a fidelidade tanto na geometria quanto na textura do modelo final, o tempo e custo computacional são objetivos secundários. Portanto, o novo pipeline se resume a três etapas: a modelagem geométrica em tempo real, a super-resolução e a reconstrução 3D de alto custo. O objetivo da primeira é proporcionar a captura completa e o armazenamento de todas as imagens, ambos em tempo real, usando o modelo atualizado apenas para guiar o usuário. Na segunda etapa, aumentamos a qualidade e resolução das imagens capturadas para a criação de um modelo mais fidedigno na etapa final, a etapa de reconstrução 3D utilizando o atual sistema do IMAGO.

ABSTRACT

With the advent of new low-cost depth sensors and with the increasing parallel processing power of graphics cards, there was a significant increase in research involving the field of real-time 3D reconstruction.

In the IMAGO research group, there is a 3D reconstruction system for digital preservation, applied to high resolution laser scanners. To increase the flexibility of the mentioned system, our goal is to contributes to the expansion of IMAGO's current 3D reconstruction pipeline to enable the creation of models using new real-time depth sensors. Another objective is the employment of a method that process the sensor's low resolution images, providing reconstructed models using higher resolution images.

The aim of digital preservation is the accuracy in both geometry and texture for the final model, the computational time and cost are secondary goals. Therefore, the new pipeline is summarized in three steps: a real-time geometric modeling, a super-resolution technique, and high-cost geometric modeling. The goal of the first step is to provide a complete capture and image storage, using the real-time model to guide the user. In the second step, we increase the quality and resolution of the captured images to create smooth and accurate models in the 3D reconstruction step using IMAGO's current system.

CAPÍTULO 1

INTRODUÇÃO

A reconstrução digital tridimensional (3D) de objetos do mundo real se tornou um dos temas mais desafiadores nas áreas de visão computacional e computação gráfica. O seu uso pode ser observado em diferentes campos: reconhecimento facial [58], navegação robótica e realidade aumentada [42], engenharia médica [4], museus virtuais [38] e preservação digital [63]. Os algoritmos para o cálculo da pose da câmera e extração da geometria do objeto evoluem em ritmo acelerado, assim como a tecnologia de captura, proporcionando modelos cada vez mais realísticos.

Apesar disso, os equipamentos de digitalização existentes são limitados pelos seus respectivos campos de visão. Por este motivo, várias aquisições a partir de pontos de vistas diferentes são necessárias para que a superfície de um objeto seja completamente capturada. Após a aquisição destas vistas, um sistema de reconstrução 3D deve alinhá-las em um mesmo sistema de coordenadas, e então combiná-las em um modelo único [42,63].

O grupo de pesquisa IMAGO[1] vem desenvolvendo um sistema de reconstrução 3D desde 2004, e diversas contribuições foram realizadas neste período. A versão original (Figura 1.1), detalhada por Vrubel *et al.* [63], abrange uma etapa de aquisição das vistas, um método de pré-alinhamento manual de pares de vistas, o alinhamento automático destas vistas, a integração volumétrica, o preenchimento de buracos, a geração da malha poligonal, e a criação do atlas de textura para adicionar cor aos vértices do modelo 3D. Atualmente, o sistema conta com pré-alinhamento automático de vistas [22,57], calibração automática entre imagens do *scanner* e de uma câmera fotográfica de alta resolução [57], geração de texturas de alta resolução [1], e integração volumétrica aprimorada [29]. Este novo *pipeline* pode ser observado na figura 1.2.

[1] www.imago.ufpr.br/

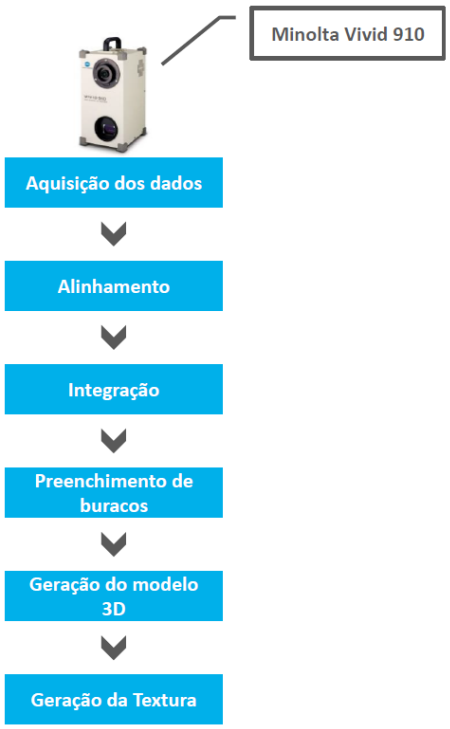


Figura 1.1: Antigo *pipeline* de reconstrução 3D do IMAGO.

O sistema desenvolvido produz modelos fidedignos, mas a captura está restringida a dispositivos de aquisição 3D de alta resolução, neste caso o *Minolta Vivid 910*[2]. Além disso, há muitos casos onde uma abordagem de alta resolução com scanners não é possível devido às restrições do ambiente de captura. Com o surgimento de equipamentos capazes de obter imagens RGBD (imagem de cor e profundidade) a 60 quadros por segundo (fps, *frames per second*), há uma demanda para métodos que trabalhem nessa mesma velocidade para aproveitar todo o potencial do equipamento.

[2]www.konicaminolta.com/instruments/products/3d/non-contact/vivid910/index

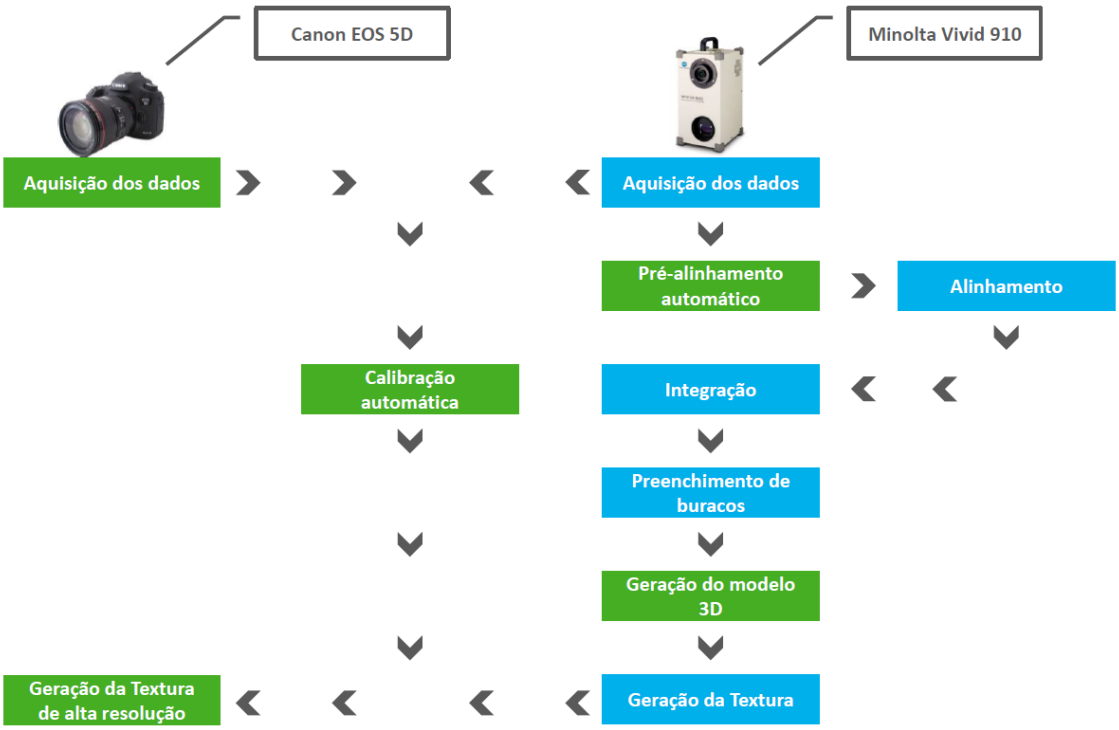


Figura 1.2: Atual *pipeline* de reconstrução 3D do IMAGO.

A contribuição deste trabalho é a ampliação do atual *pipeline* de reconstrução 3D do IMAGO para permitir a criação de modelos utilizando os novos sensores em tempo real. O dispositivo escolhido é o Kinect[3], um sensor de baixo custo utilizado em sistemas estado-da-arte para reconstrução de objetos em tempo real [27, 32, 41, 42]. Entretanto, o processo de reconstrução requer um elevado poder de processamento para manipular as informações capturadas em um curto período de tempo, e inicialmente a CPU era o único recurso disponível para a realização desta tarefa. Com o advento da GPU de Propósito Geral (GPGPU, ou *General-Purpose Computing on Graphic Unit*), a placa de vídeo passou a ser utilizada para tarefas antes exclusivas da CPU.

Enquanto as CPUs atuais possuem de quatro a oito núcleos, uma GPU[4] contém centenas de núcleos menos potentes, capazes de processar grandes blocos de informação em paralelo. A estrutura do método Única Instrução, Múltiplos Dados (SIMD, *Single Instruction, Multiple Data*) combina perfeitamente com este alto nível de paralelismo, proporcionando um alto ganho em desempenho. Embora a comunicação e acesso à memória

[3] www.xbox.com/kinect
[4] www.nvidia.com/object/What-is-GPU-Computing.html

sejam fatores limitantes, quando explorados corretamente demonstram um grande ganho em tempo de execução [27, 41, 42]. Quanto ao tipo de informação, os pontos 3D e *pixels* de cada vista são altamente independentes, fato que os torna ideais ao processamento em paralelo [7].

Pesquisas relacionadas à modelagem geométrica em tempo real evoluíram com este avanço tecnológico. Mas analisando o ponto de vista qualitativo, tais sistemas não são suficientes para a criação de modelos tridimensionais fidedignos. Um melhor resultado poderia ser alcançado melhorando a etapa de reconstrução volumétrica, mas isto não é possível sem comprometer a velocidade do sistema. Qualquer mudança diminuiria a taxa de captura que, consequentemente, aumentaria a probabilidade de falhas no rastreamento. Da mesma forma, não justifica apenas adicionar um sensor em tempo real em um sistema de reconstrução 3D de alto custo. Tal sistema é lento e limitaria a capacidade do sensor. Além disso, a informação durante a captura é insuficiente para determinar a sua conclusão. Logo, o trabalho torna-se repetitivo devido à necessidade de refazer a aquisição dos dados enquanto o modelo reconstruído está incompleto.

Este trabalho explora as vantagens de ambos os sistemas mencionados, como pode ser visto na Figura 1.3. Na primeira etapa, é utilizado o modelo reconstruído em tempo real para guiar o usuário durante o manuseio do Kinect no processo de captura. Assim, eliminamos o retrabalho da aquisição. Em seguida, aplicamos uma técnica de super-resolução para aumentar a qualidade e diminuir o ruído das imagens originais. Finalmente, reconstruímos o modelo com as novas imagens utilizando um processo de maior custo computacional para criar modelos mais fidedignos.

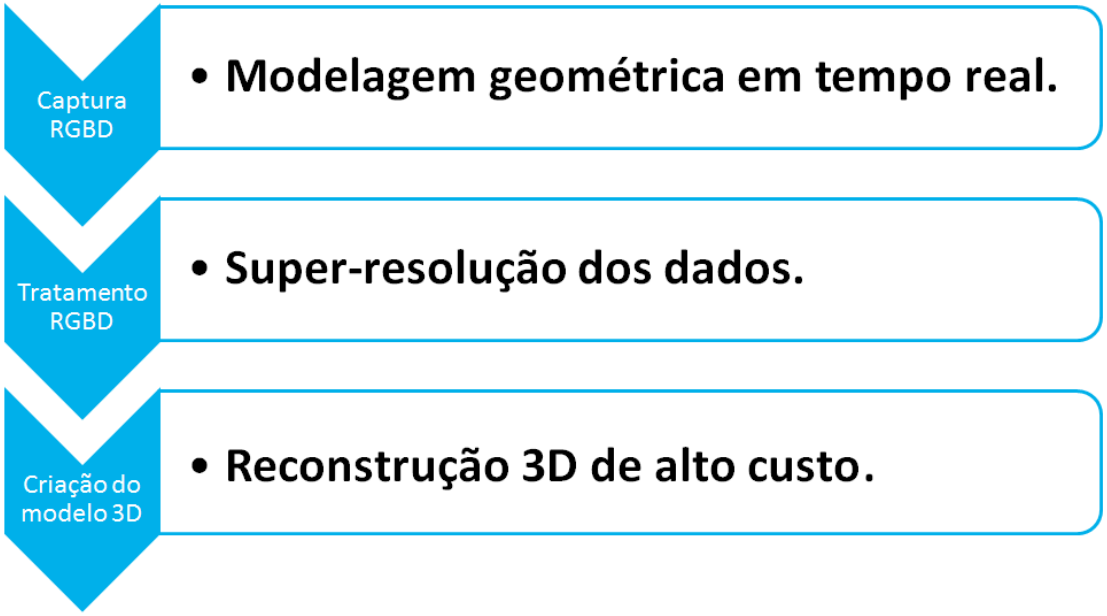


Figura 1.3: Estrutura do novo *pipeline* do IMAGO.

Esta dissertação está organizada conforme descrito a seguir. O capítulo 2 apresenta o dispositivo de captura escolhido para este trabalho. No capítulo 3 é apresentado o sistema de captura e modelagem em tempo real. Os métodos para obtenção da super-resolução das imagens do dispositivo são introduzidos no capítulo 4. No capítulo 5 é apresentado o sistema de reconstrução 3D, e os experimentos estão descritos no capítulo 6. Finalmente, no capítulo 7 temos a conclusão e proposta de trabalhos futuros.

CAPÍTULO 2

TECNOLOGIA DE CAPTURA

Estamos interessados neste trabalho somente em dispositivos de baixo custo que realizem a captura em tempo real da informação 3D, como os de tempo-de-voo (ToF, *time-of-Flight*) [4, 25] e de luz estruturada [51, 64]. No momento, os sensores mais interessantes para o projeto são aqueles que possuem um equilíbrio entre velocidade de aquisição, qualidade das imagens e custo financeiro. Os dispositivos que encontram-se nessa categoria são aqueles que utilizam a tecnologia de luz estruturada desenvolvida pela PrimeSense [61], entre eles temos o Kinect, o ASUS Xtion PRO LIVE[1] e o sensor PrimeSense.

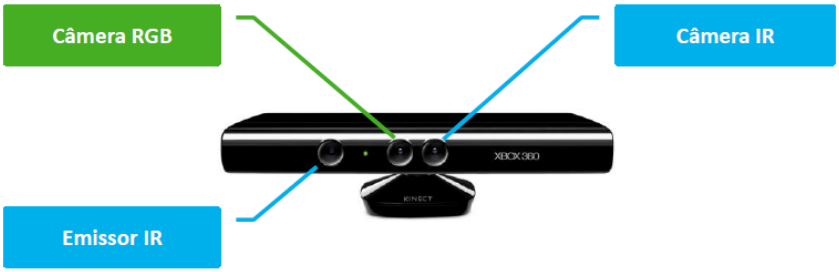


Figura 2.1: Componentes do sensor Kinect. *Fonte: Microsoft*

O dispositivo escolhido para este trabalho é o Kinect. Produzido pela Microsoft e voltado originalmente para a área de entretenimento, seu objetivo é a captura dos movimentos corporais que são interpretados como comandos. Entretanto, após o seu lançamento, este dispositivo foi adotado como um sensor de baixo custo em pesquisas que envolvem o uso da informação 3D para a reconstrução digital [32, 41, 42, 52].

O Kinect é equipado com uma câmera em cores(RGB), um emissor de laser infravermelho (IR, *Infrared*) e uma câmera IR, como ilustrado na Figura 2.1. De acordo com Barak *et al.* [3], a medição da profundidade é feita por triangulação. O emissor IR projeta um único raio que atinge duas grades de difração (Figura 2.2) e é dividido em múltiplos raios que formam um padrão constante de pontos luminosos [43]. O padrão captado pela

[1] http://www.asus.com/Multimedia/Motion_Sensor/Xtion_PRO_LIVE/

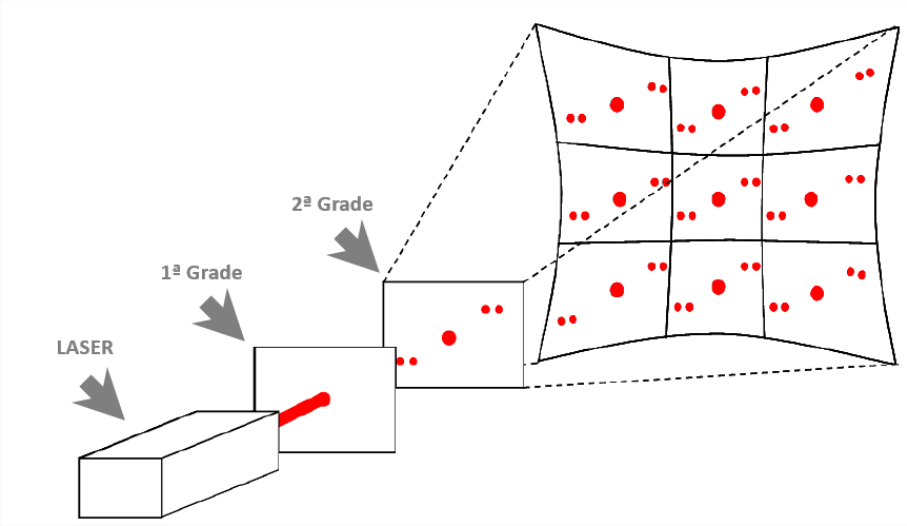


Figura 2.2: O raio é projetado na primeira grade de difração, e dividido em diferente padrões. Estes são duplicados na segunda grade e projetados no ambiente. Fonte: Nordmark [43]

câmera de infravermelho é comparado com o padrão de referência para calcular o mapa de profundidade [30]. Cada ponto desse mapa contém a distância entre o sensor e a respectiva posição na cena capturada. Baseado nesses valores é possível recuperar coordenadas 3D destes pontos.

Figura 2.3: Tipos de informação capturados pelo sensor Kinect.

2.1 Calibração

O Kinect captura por padrão as informações de cor e profundidade na resolução de 640×480. A distância das duas câmeras, a IR e RGB, causa uma pequena distorção (figura 2.4) que impossibilita que imagens de ambas as câmeras se combinem naturalmente [27,30,42]. Desta forma, a calibração se torna uma etapa indispensável para garantir o correto uso dessas imagens.

Adotamos a calibração estéreo [8] que provou ser suficiente para diminuir significativamente a disparidade entre a imagem de cor e profundidade. É importante lembrar que a calibração difere para cada aparelho, pois a distância das duas câmeras varia.

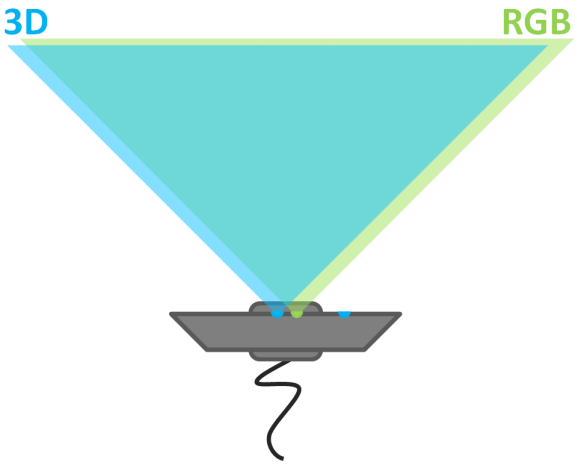


Figura 2.4: Demonstração da distorção entre o sensor infravermelho e a câmera.

Neste processo é utilizada a biblioteca de processamento de imagens OpenCV[2] e um objeto de superfície plana com padrões de um tabuleiro de xadrez, como observado na figura 2.5.

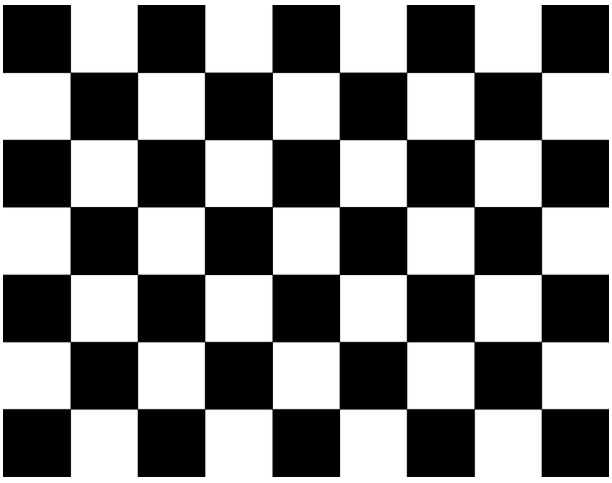


Figura 2.5: Imagem utilizada no processo de calibração. A imagem é colocada na superfície de um objeto plano para facilitar a manipulação.

Em cada imagem capturada, marcamos as quinas internas do tabuleiro utilizando a função *findChessboardCorners* [8] da biblioteca OpenCV, que retorna a posição com base no tamanho do tabuleiro que foi especificado. Encontradas as posições (figura 2.6), a

[2]http://opencv.org/

função *cornerSubPix* [8] refina ainda mais a localização das quinas a nível de *sub-pixel*.

Figura 2.6: Resultado da busca pelas quinas do tabuleiro.

Estas imagens são utilizadas para o cálculo da calibração intrínseca da câmera. A funçao *calibrateCamera* [8] retorna a matriz com os parâmetros de calibração íntrinseca da câmera que descrevem a sua geometria interna:

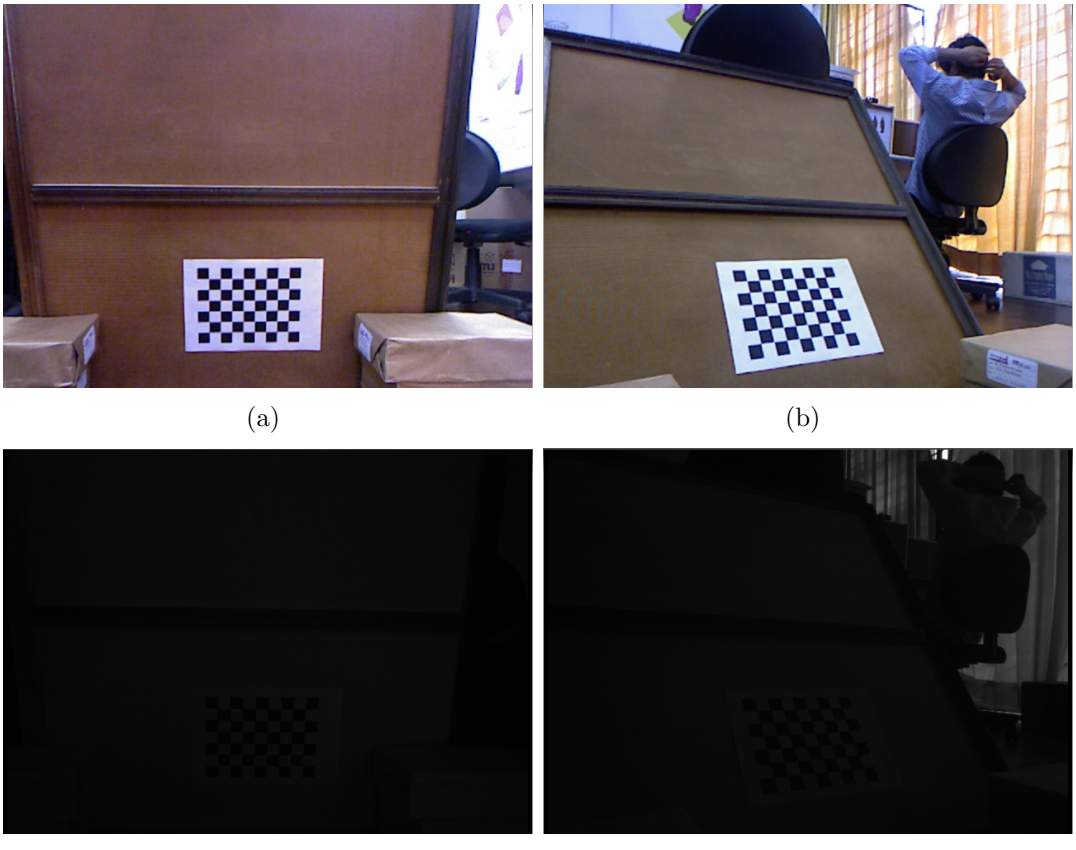
$$A = \begin{bmatrix} f_x & \gamma & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 0 \end{bmatrix} \tag{2.1}$$

onde f_x e f_y correspondem à distância focal em largura e altura, c_x e c_y os pontos principais (geralmente o centro da imagem), e γ o coeficiente de distorção entre o eixo x e y, frequentemente representado pelo valor 0. Esta calibração é feita separadamente para as duas câmeras. Além dos parâmetros, a função de calibração também retorna o vetor de distorção:

$$D_{coef} = [k1, k2, p1, p2, k3] \tag{2.2}$$

onde $k1, k2, k3$ são coeficientes de distorção, e $p1, p2$ são a distorção tangencial. A matriz intrínseca não depende da cena vista, e quando estimada, pode ser reutilizada desde que a distância focal permaneça a mesma.

O parâmetro extrínseco é a relação entre o sistema de coordenadas 3D da câmera e o sistema de coordenadas global. Este processo necessita da matriz de calibração intrínseca e vetor de distorção do processo anterior das duas câmeras a serem calibradas. Um novo conjunto de imagens é capturado, desta vez adquiridos sob os mesmos pontos de visão e organizados na mesma ordem de captura, como demonstradas na figura 2.7.



(a) (b)

(c) (d)

Figura 2.7: Algumas imagens capturadas para o processo de calibração extrínseca. (a) e (b) correspondem às imagens de cor sob o mesmo ponto de vista das imagens IR (c) e (d) respectivamente.

A sua matriz $[R \mid t]$, a junção entre os dados de rotação e translação, é usado para descrever o deslocamento de um objeto na cena ou vice-versa:

$$[R \mid t] = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \tag{2.3}$$

e utilizando as matrizes 2.1 e 2.3, os pontos 3Ds de uma cena podem ser reprojetados para um plano 2D de uma imagem usando a transformação em perspectiva:

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & \gamma & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}, onde \quad z \neq 0 \tag{2.4}$$

O (u, v) corresponde à coordenada dos pontos de projeção do pixel e (X, Y, Z) a coordenada 3D do ponto no sistema de coordenadas global. Utilizamos a função *stereo-Calibrate* [8] para calcular a matriz de transformação entre a câmera IR e cor. Com estas informações, é possível fazer o processo inverso para o cálculo da nuvem de pontos com cor.

Nesta etapa, utilizamos um tabuleiro 9×6 com quadrados de $2,5$ centímetros de lado, cerca de 5 imagens para calibração intrínseca e 20 para a extrínseca. O processo levou dois minutos para a sua conclusão.

CAPÍTULO 3

MODELAGEM GEOMÉTRICA EM TEMPO REAL

A modelagem geométrica em tempo real engloba uma série de passos, executados no intervalo de um segundo, que resulta em modelos 3Ds com a mesma forma do objeto alvo. Este tipo de pesquisa alavancou com o surgimento dos dispositivos de captura em tempo real e com o avanço tecnológico dos computadores. Em 2002 por Rusinkiewicz *et al.* [51] foram responsáveis pelo primeiro sistema de reconstrução em tempo real capaz de renderizar o modelo a *10 fps*.

Weise *et al.* [64] apresentam um algoritmo baseado no de Rusinkiewicz *et al.* que combina um alinhamento mais robusto e uma diferente técnica de renderização para operar a *20 fps*. Diferente dos métodos anteriores, Henry *et al.* [24] combinam a informação de cor para melhorar a robustez do alinhamento, mas isto diminui o desempenho para *2 fps*.

Trabalhos mais recentes conseguem utilizar todo o potencial dos dispositivos de captura com o uso da programação paralela em GPU. Neumann *et al.* [41] conseguem reconstruir o cenário utilizando os *30 fps* do dispositivo Kinect.

Neste trabalho utilizamos a implementação *open source* do sistema estado-da-arte *KinectFusion* [27, 42], desenvolvido sob o projeto *Point Cloud Library* (PCL) [52]. O algoritmo, implementado em GPU, é uma combinação de um alinhamento rápido e uma representação volumétrica precisa.

A Figura 3.1 demonstra as etapas do nosso sistema de modelagem geométrica em tempo real, detalhadas nas próximas seções. A seção 2.1 descreveu a etapa de calibração do aparelho, realizado antes de todo o processo de modelagem. Em seguida, a etapa de aquisição, pré-processamento, alinhamento, representação volumétrica e renderização do modelo serão descritos nas seções 3.1, 3.2, 3.3, 3.4 e 3.5, respectivamente.

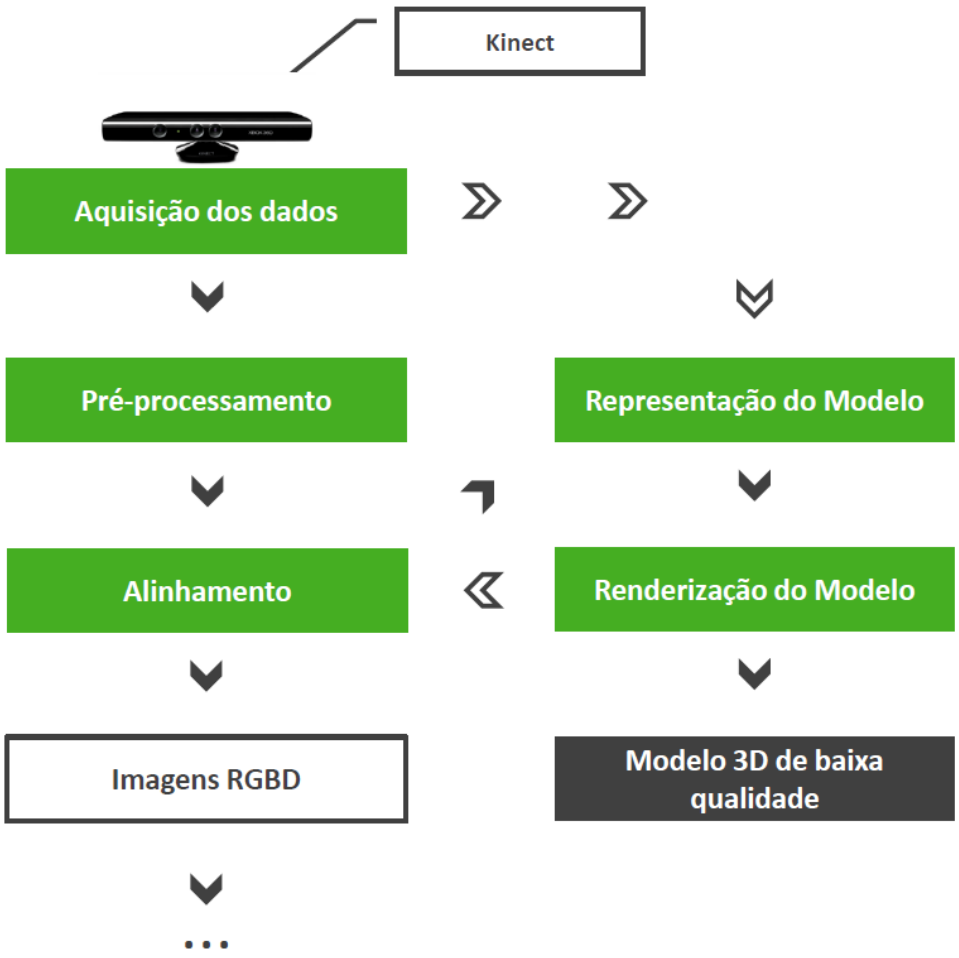


Figura 3.1: Primeira parte do nosso *pipeline*. Esta etapa é responsável pela modelagem geométrica em tempo-real. As setas preenchidas representam o fluxo do sistema, iniciando com a aquisição dos dados. As setas sem preenchimento correspondem às informações que a etapa necessita para o seu processo, como informações de rotação e translação para poder realizar o rastreamento.

3.1 Aquisição

O Kinect é capaz de capturar 30 fps de informação de profundidade, com precisão de 11 bits, e outros 30 fps de informação de cor [30] na resolução de 640×480 Além disso, ele possui um campo de visão horizontal e vertical de aproximadamente 60^o [24] e um alcance de $0,5$ a $5,0$ metros [30]. As imagens foram capturadas ao mover o Kinect em volta do objeto.

Entretanto, as imagens de profundidade apresentam uma grande quantidade de ruídos e discrepâncias [24, 27, 42]. Além disso, o erro da informação de profundidade aumenta quadraticamente conforme aumenta a distância do sensor ao objeto, e a resolução diminui

quadraticamente com o aumento da distância do sensor. Por este motivo, a informação deve ser capturada entre $1,0$ e $3,0$ metros para evitar a degradação do mapa de profundidade por ruído e baixa resolução [30]. Apesar dessas desvantagens, o Kinect ainda é uma alternativa interessante devido ao baixo custo e a captura em tempo real, sendo esta última a característica essencial para este trabalho.

3.2 Pré-processamento

Devido ao ruído e discrepâncias na imagem 3D, é evidente a necessidade de uma etapa para reduzi-los. No entanto, não nos é interessante os complexos métodos de super-resolução que utilizam a informação temporal, ou seja, os que precisam de n imagens para criar uma com alta resolução [14], pois tornaria um desafio paralelizar em GPU este processo.

Logo, o nosso sistema em tempo real utiliza o filtro bilateral [62], um processo simples que combina a filtragem espacial com a filtragem da informação de profundidade. Dado um determinado tempo k, e seja i o pixel na posição (u, v) na imagem $I_k \subset \mathbb{R}^2$, aplicamos o filtro na imagem bruta e recebemos a vista D_k com redução no ruído:

$$D_k(i) \quad = \quad \frac{1}{W_p} \sum_{j \in \Omega} f(\parallel i - j \parallel) g(\parallel I_k(i) - I_k(j) \parallel) I_k(j) \tag{3.1}$$

onde f é o filtro espacial, g é o filtro da informação de profundidade, W_p um fator de normalização, e i e j são as coordenadas do pixel. A soma de $f \cdot g$ filtra o peso e preserva as bordas, pois há pouca variação em $f \cdot g$ à medida que aumentamos a distância espacial e os valores de profundidade. Este filtro bilateral provou ser eficiente no sistema de reconstrução *KinectFusion* [42]. Além de reduzir o ruído, aumenta significativamente a qualidade do mapa das normais produzido e melhora a associação de dados, necessários para o rastreamento.

A informação de profundidade é capturada na forma de uma matriz 2D, e cada pixel representa um ponto 3D. O valor contido no mapa de profundidade varia de 0 a 2047, porém, este valor não corresponde à distância real do ponto. A equação a seguir é usada

para transformar as coordenadas (x, y, z) do sistema de coordenadas globais para coordenadas de pixel (u, v):

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = R \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + t \tag{3.2}$$
$$u = \frac{f_x \times x}{z} + c_x$$
$$v = \frac{f_y \times y}{z} + c_y$$

onde (X, Y, Z) são as atuais coordenadas globais, $[R \mid t]$ a matriz de parâmetros extrínsecos (equação 2.3), e f_x, f_y, c_x e c_y são os parâmetros de calibração intrínseca (equação 2.1). Para encontrar a nuvem de pontos através de imagens, temos:

$$\begin{aligned} x &= \frac{u - c_x}{f_x} \times z \\ y &= \frac{v - c_y}{f_y} \times z \end{aligned} \tag{3.3}$$

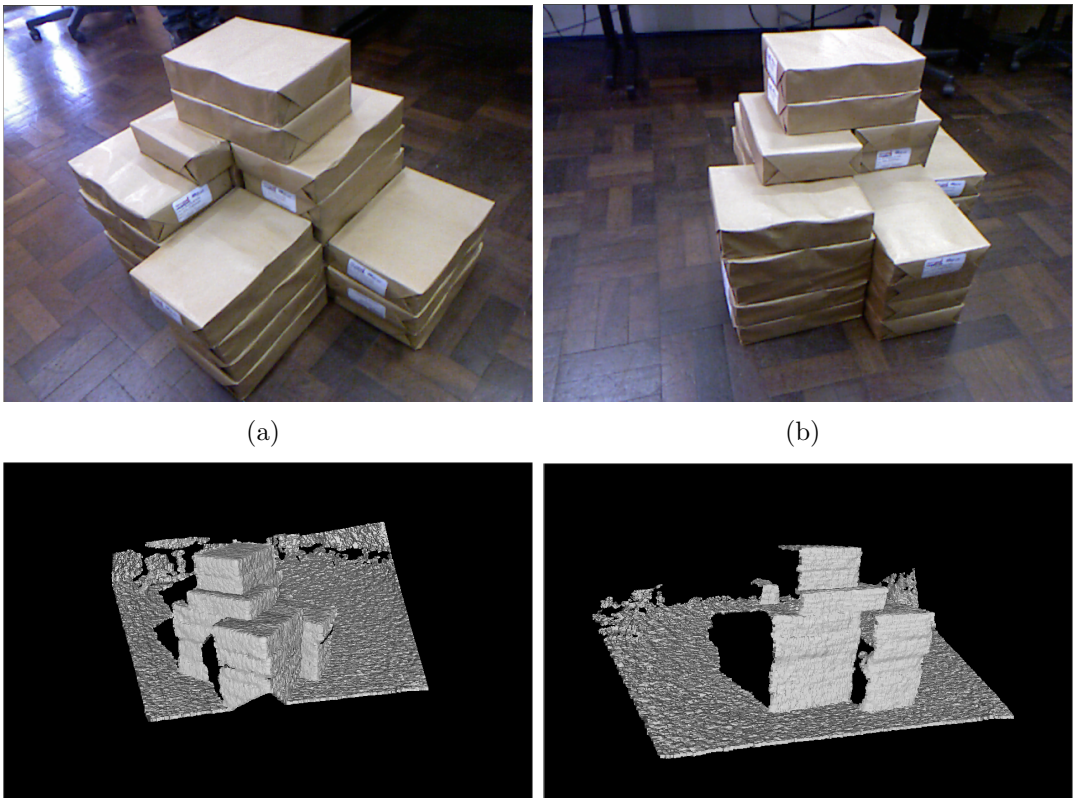
A coordenada z é calculada com os parâmetros oferecidos pelo *framework Robot Operating System*[1] (ROS):

$$\begin{aligned} z &= \frac{1}{p(u,v) * c1 + c2} \\ c1 &= -0.0030711016 \\ c2 &= 3.3309495161 \end{aligned} \tag{3.4}$$

onde $p(u, v)$ é o valor do pixel na posição (u, v). O resultado deste cálculo é uma nuvem de pontos (x, y, z) calibrada como visto na figura 3.2. Esta nuvem de pontosé reprojetada na imagem de cor para a posição correta usando a equação 3.2 previamente utilizada, desta

[1] http://www.ros.org/wiki/

vez utilizando os pontos (x, y, z) da nuvem de pontos como valores de entrada para $[R \mid t]$. Assim, encontramos o pixel (u, v) para cada um desses pontos utilizando as equações 3.3 e 3.3. A figura 3.3 mostra o resultado final do processo de transformação.



(a) (b)

(c) (d)

Figura 3.2: Resultado do filtro e cálculo dos vértices. (a) e (b) fotografia do objeto, capturados sob diferentes perspectivas; (c) e (d) imagens de profundidade do objeto.

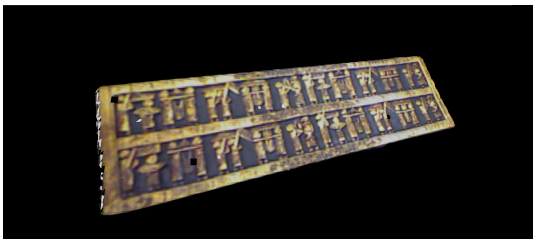
3.3 Alinhamento

Durante o processo de aquisição são capturadas imagens de diferentes vistas devido ao limite óptico, ou oclusão, que impossibilita a reconstrução do objeto 3D usando apenas uma imagem [24,27,32,33,42,63], como é demonstrado na figura 3.4. Cada imagem representa uma região parcial do objeto a ser reconstruído e possui um sistema de coordenadas próprio. O objetivo do alinhamento é posicionar estas imagens em um mesmo sistema de coordenadas global. O sucesso do método depende da qualidade e área de sobreposição entre as imagens, ou seja, da região em comum entre elas.

Inicialmente, é necessário calcular uma estimativa inicial do alinhamento através

(a)

(b)



(c)

Figura 3.3: Resultado do processo de calibração e transformação usado no projeto.(a) Fotografia do objeto. (b) e (c) Resultado da combinação da nuvem de pontos com a de cor.

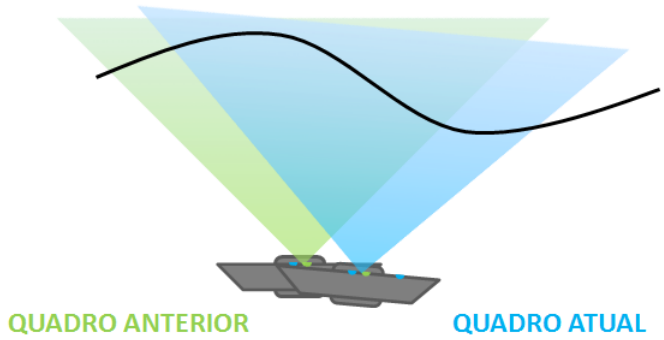


Figura 3.4: Simulação do processo de aquisição, demonstrando a limitação dos sensores de captura.

do pré-alinhamento [22]. Como o Kinect captura as imagens de profundidade a uma frequência de *30 fps*, o curto intervalo de tempo entre as imagens durante o processo de captura já é suficiente para servir de base para a estimativa inicial.

A próxima etapa consiste em refinar a matriz de transformação para encontrar a solução mais precisa possível. Para isto, muitos algoritmos utilizam correspondências temporais, também conhecidas como pontos mais próximos. O alinhamento necessita encontrar os pontos correspondentes e, infelizmente, esta é uma das etapas que requer maior tempo computacional ($O(N^2)$). Muitas foram as propostas para maximizar a sua eficiência [9, 10, 19, 40, 68].

Existem diversos métodos e as suas variantes para realizar o alinhamento. Dentre eles *Signed Distance Fields* [6,36], Algoritmos Genéticos (GA) [13,59], e *Iterative Closest Point* (ICP) [5,12]. Neste trabalho, será adotado o ICP, pois este já provou ter resultados mais robustos que os demais métodos [54]. O ICP e suas variantes [24, 50, 56] são os métodos de alinhamento mais utilizados para imagens de profundidade (*i.e.* alinhamento par-a-par).

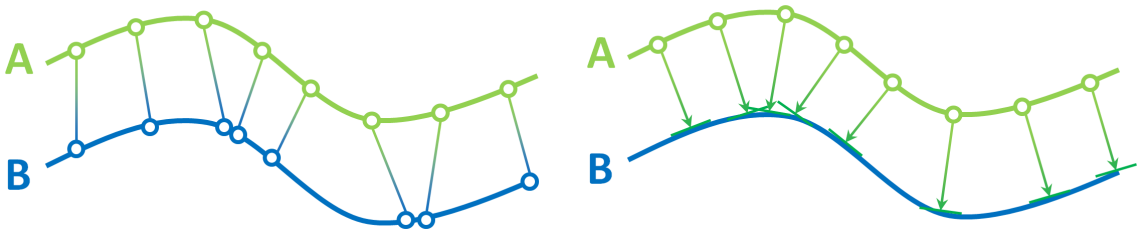
A ideia básica do ICP é calcular a matriz de tranformação de tamanho 4×4 que minimize a distância entre pontos de controle na primeira imagem com seus respectivos pontos mais próximos na segunda imagem. Essa característica faz com que muitos sistemas baseados em câmeras ativas, *i.e.* câmeras que iluminam o objeto (iluminação estruturada), optem por este alinhamento ao contrário de câmeras passivas que realizam extração de características para correspondência.

Resumidamente, o ICP funciona da seguinte maneira: dados dois conjuntos de pontos A e B, para cada ponto de controle a_i em A, encontrar o ponto mais próximo b_i em B. Em seguida uma transformação T é obtida para minimizar as distâncias entre pares de pontos correspondentes (a_i, b_i). Todos os pontos em A são então transformados por T, e o processo é repetido até que a distância entre os pontos correspondentes seja menor do que um limiar d_{max} ou até a convergência.

O ICP alcança bons resultados mesmo na presença de ruído gaussiano. No entanto, o método não é eficaz quando há pouca sobreposição entre A e B, aumentando a quantidade de iterações necessárias, e, em alguns casos, convergindo para um mínimo local.

A primeira variante do ICP proposta por Chen e Medioni [12] é baseada na minimização da distância entre ponto e plano. Considerando um ponto na primeira imagem, é projetado seu vetor normal na segunda imagem, e o ponto correspondente é escolhido onde o vetor incidiu sobre a superfície. No ICP, dado um ponto na primeira imagem, é simplesmente escolhido o ponto mais próximo na segunda como correspondência. A figura 3.5 retrata a diferença entre os dois tipos de alinhamento. A abordagem de Chen e Medioni é ligeiramente mais complexa do que o ICP original, mas seu método é robusto para imagens ruidosas e imagens com pouca sobreposição entre si, além de utilizar menos

iterações.



(a) (b)

Figura 3.5: A busca pelos pontos correspondentes na etapa de alinhamento. (a) Método ponto-a-ponto do ICP. (b) Método ponto-ao-plano de Chen e Medioni.

Rusinkiewicz e Levoy [50] aprimoraram a abordagem de Chen e Medioni usando uma técnica chamada *Normal Space Sampling*. Esta técnica prioriza a seleção de pontos de controle em regiões onde as normais são diferentes de outras partes da imagem, obtendo resultados melhores em superfícies de baixa qualidade. Weise *et al.* [64] combinam o ICP com a detecção de falhas no alinhamento para conseguir um sistema de modelagem mais robusto.

Johnson e Kang [28] combinam a informação de cor com a informação 3D criando uma técnica mais robusta chamada *Color ICP*. Adicionando a diferença de cor na métrica de distância usada para encontrar pontos correspondentes garante um alinhamento mais eficaz, principalmente em regiões onde há ambiguidade na informação 3D [32, 41]. Este tipo de alinhamente é usado em sistemas de recostrução 3D, como de Henry *et al.* [24] e Neumann *et al.* [41]. A desvantagem em usar a cor é a limitação que ela impõe, já que o sistema deixa de funcionar para ambientes de pouca ou nenhuma iluminação.

O algoritmo ICP continua sendo a técnica estado-da-arte em alinhamento, estando presente nos atuais sistemas de reconstrução em tempo real [24, 27, 41, 42]. Por este motivo, este trabalho também utilizará o alinhamento ICP adaptado à GPU.

O maior gargalo do ICP é a etapa de busca pelos pontos correspondentes. Em termos de complexidade, um método de força-bruta necessita de tempo $O(N^2)$, um custo computacional muito alto quando estamos manipulando uma grande quantidade de informação, enquanto que o método da árvore *kd-tree* [18], uma técnica de busca baseada na divisão da estrutura de dados, reduz o tempo computacional para a complexidade $O(N \log N)$.

Entraremos em mais detalhes nas próximas subseções.

3.3.1 Busca por força-bruta

Nesta subseção serão mencionados e analisados os trabalhos que focam no desempenho dos algoritmos de força-bruta em GPU que, dependendo de como são implementados, consegue superar as *kd-trees* em CPU graças à característica SIMD dos multiprocessadores GPU [19].

Cayton [9] apresentou uma abordagem que explora a vantagem da força bruta e do paralelismo, o *Random Ball Cover*(RBC). O algoritmo RBC consiste somente em duas operações de força-bruta, e cada operação observa apenas uma fração da nuvem de pontos. Assumindo duas imagens, $A \subset \mathbb{R}^d$ e $B \subset \mathbb{R}^d$, onde d é o número de dimensões, iremos considerar um cojunto de n_a pontos de controle $C_A \subset \mathbb{R}^d$ que são os pontos cujo ponto mais próximo, contido na vista B, é desejado. Embora o trabalho possa lidar com qualquer métrica, o foco do trabalho são as distâncias baseadas na *norma p*, onde $p \geq 1$, ou seja:

$$d(a,b) = \| a - b \|_p = (\sum_{i=1}^{n} (a_i - b_i)^p)^{\frac{1}{p}} \tag{3.5}$$

Na vista B, temos um conjunto n_b de pontos representativos $R_B \subset B$, que são escolhidos aleatoriamente, onde cada ponto representativos aponta para um subconjunto da vista B. Para cada $b \in R_B$, a estrutura de dados mantém uma lista L_b que contém os s_b pontos mais próximos de b em B. Como pode ser observado na figura 3.6, pode haver sobreposição de listas.

A busca RBC de um ponto de controle da imagem A pelo ponto mais próximo em B é realizado da seguinte maneira: I) o algoritmo compara a com os elementos do conjunto R_B, e escolhe o que tem a menor distância; II) encontrado o menor ponto representativo, o b^*, o ponto a agora procura por todos os elementos da lista L_{b^*} para encontrar seu ponto mais próximo. Utilizando este método, a complexidade do problema diminui para $O(\sqrt{N} \log N)$.

Este algoritmo tem a possibilidade de retornar um valor incorreto, mas a probabi-

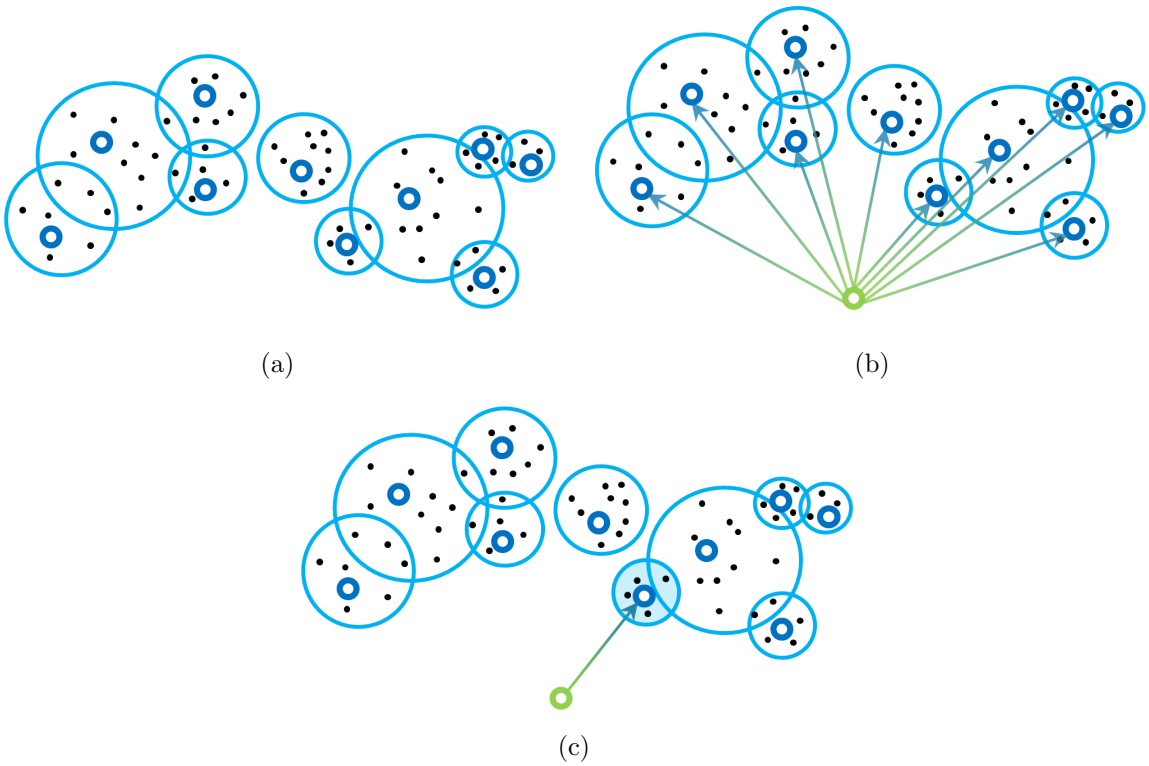


Figura 3.6: O funcionamento da técnica RBC. (a) Os pontos pretos representam os pontos que fazem parte da vista B. Os pequenos círculos azuis são os pontos representativos b e os círculos grandes as listas L_b de cada ponto representante. (b) A primeira busca é feita verificando o ponto representativo mais próximo. (c) Encontrado o b^* , a segunda busca verifica em L_{b^*} pelo ponto mais próximo de a e conclui o algoritmo. Fonte: Cayton [9]

lidade de acontecer este evento pode ser reduzida com o aumento do tamanho das listas ou o número de pontos representativos. Nos primeiros testes com o sistema *open source* do grupo PCL, o *KinFu*, implementamos a variação do RBC, proposta por Neumann *et al.* [41] por simplificar a construção do algoritmo para uma única busca por força bruta, como pode ser observado na figura 3.7. Em troca de rapidez, sacrificamos a robustez do método, o que causou eventuais problemas de alinhamento durante o processo de captura. Abandonamos a idéia de utilizar este método de busca por pontos correspondentes pelas busca em árvores kd-trees.

3.3.2 Busca por K-d trees

A árvore *k-d tree*, desenvolvida por Friedman *et al.* [39], é uma eficiente estrutura de dados utilizado na busca dos pontos mais próximos em um espaço de k dimensões. Cada nível da *k-d tree* divide todos os seus filhos sob uma dimensão específica. A figura 3.8

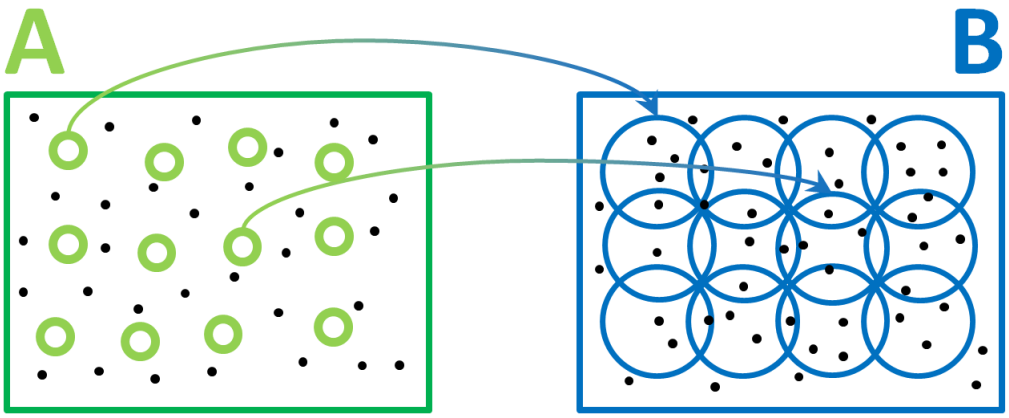


Figura 3.7: O funcionamento da variante do RBC. Na busca de Neumann *et al.* [41], os pontos de controle de A procuram em regiões distintas de B pelo seus pontos mais próximos.

demonstra o funcionamento de árvore kd-tree. No nó raiz da árvore, os filhos serão jogados em uma sub-árvore à esquerda caso o valor da primeira dimensão for menor que o valor da raiz, caso contrário serão jogados na sub-árvore à direita. Este processo se repete nas sub-árvores (esquerda e direita) das próximas dimensões, retornando sempre à primeira dimensão quando as outras forem exploradas. O ciclo termina quando a última sub-árvore for composta por um único elemento.

O kd-tree foi projetado para baixas dimensões, o que explica a sua rápida degradação de performance em altas dimensões. Trabalhos que implementaram a *kd-tree* em GPU [40,48,68] coletaram excelentes resultados em desempenho de busca, porém, o desempenho geral caiu devido ao custo adicional em criar a estrutura da árvore em CPU.

No entanto, Muja e Lowe [39] propuseram um método para busca de pontos vizinhos que detecta automaticamente o melhor algoritmo e valores de parâmetro para determinado conjunto de pontos. Muja e Lowe comparam diferentes algoritmos de busca em diversos conjuntos de dados, e encontram as duas melhores abordagens: a *kd-tree* aleatória e a árvore *k-means* hierárquica.

Diferente das árvore de busca clássicas, a *kd-tree* aleatória divide em sub-árvores aleatórias a partir de uma dimensão D em que os dados apresentam maior variação. Os autores utilizaram o valor $D = 5$ pois este apresentou o melhor resultado em seu conjunto de dados.

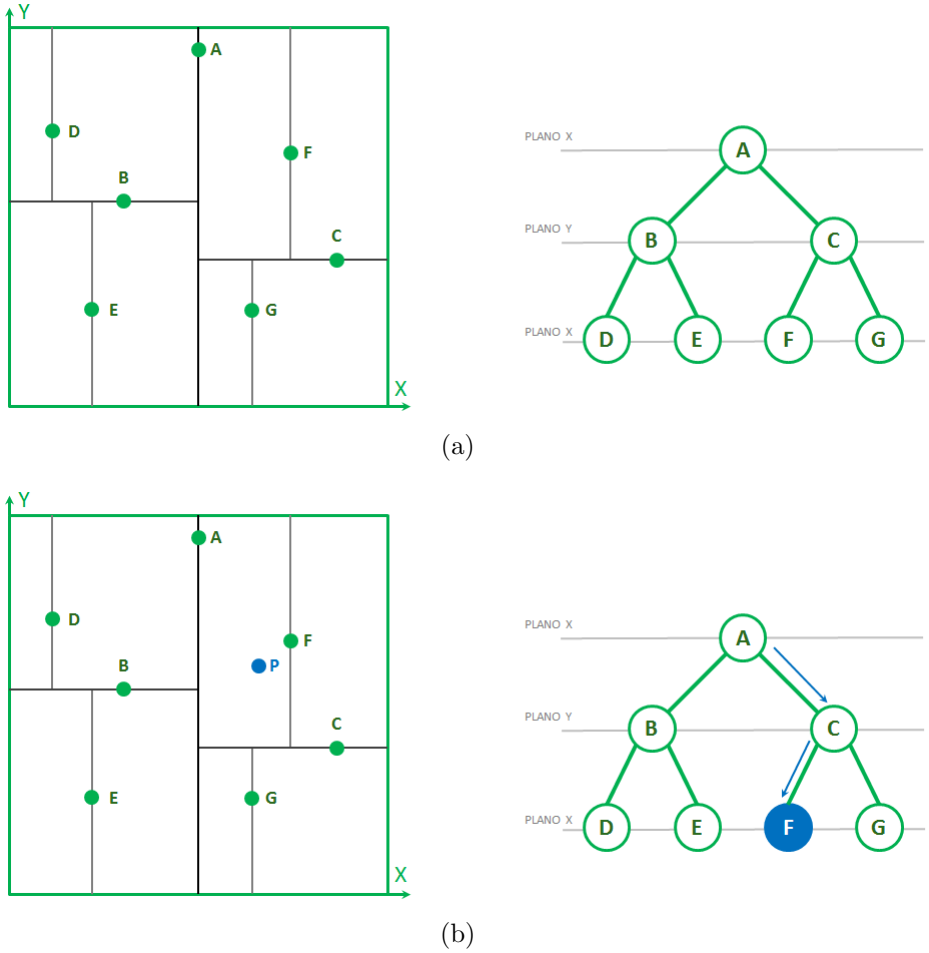


Figura 3.8: Demonstração de uma kd-tree 2D. A figura (a) representa o processo de construção da kd-tree que se inicia no ponto A e plano X. Valores menores que A são colocados na sub-árvore da esquerda, e os valores maiores na direita. O mesmo se repete para o plano Y e assim por diante. O processo termina quando o elemento da sub-árvore for uma folha. Na figura (b) temos o processo de busca, que percorre apenas o subconjunto que o leva ao ponto mais próximo.

A árvore *k-means* hierárquica divide o conjunto de dados em K regiões distintas usando *k-means*, e então aplica o mesmo método recursivamente aos pontos da mesma região. A recursão termina quando o número de pontos em uma região é menor que K. O algoritmo de Muja e Lowe difere da técnica tradicional de busca em profundidade, pois a busca é feita utilizando uma fila de prioridades, proporcoinando os melhores resultados para a maioria do conjunto de dados.

Neste projeto, é utilizado o método de busca pelos pontos correspondentes desenvolvido por Muja e Lowe, que encontra automaticamente o melhor algoritmo de busca por *kd-tree* para o nosso conjunto de pontos 3D.

3.4 Representação do Modelo

A visualização do modelo reconstruído é a melhor alternativa para acompanhar o processo de alinhamento. O processo mais simples é a renderização de todas as imagens capturadas, mas isso afeta consideravelmente o desempenho do sistema pois há muita informação redundante. Para melhorar o processo de visualização é necessário eliminar os pontos sobrepostos das imagens alinhadas para que o modelo 3D fique o mais simples possível.

Rusinkiewicz *et al.* [51] renderizam seus modelos com *splats*, uma técnica simples que desenha um círculo em cada ponto, com tamanho suficiente para que estejam sobrepostas e que não haja lacunas. Os resultados são rápidos, porém possuem qualidade inferior à triangulação dos pontos e à construção das malhas poligonais. A reconstrução final é efetuada após o término das etapas anteriores, com o algoritmo *Volumetric Range Image Processing* (VRIP).

Weise *et al.* [64] e Henry *et al.* [24] utilizam *surfels* [46], uma técnica que armazena o grau de confiança dos pontos, isto é, o grau aumenta conforme o mesmo ponto aparece em diferentes imagens durante a captura.

Dentre os tipos de métodos de integração existentes, o volumétrico oferece menor restrição aos objetos reconstruídos [63]. Enquanto que técnicas como a triangulação de pontos e geração de malhas possuem custo computacional alto e sensibilidade a ruído, o volumétrico consegue trabalhar com dados de baixa resolução e atenua o ruído com as imagens disponíveis. Sua grande desvantagem é o alto custo em processamento e memória, que pode ser contornado usando programação em GPU.

Utilizamos a representação volumétrica do *KinectFusion*, que é baseada no método de Curless e Levoy [15]. Dada a pose global da câmera, pontos orientados são convertidos para coordenadas globais, e um único volume é atualizado. Este volume é subdividido uniformemente em uma grade 3D de *pixels* volumétricos, ou *voxels*. Os vértices globais são integrados nos *voxels* usando a Função de Distância Sinalizada (SDF, ou *Signed Distance Function*) [15], especificando uma distância relativa da superfície atual. A metodologia do SDF é simples, como pode ser vista na figura 3.9. A superfície é representada pelo valor zero, à frente da superfície estão os valores positivos, e atrás dela os negativos. O

sistema armazena apenas uma região truncada da superfície atual (TSDF) e, embora a construção do volume não seja eficiente em memória, são muito eficientes em velocidade: os acessos levando cerca de $2ms$.

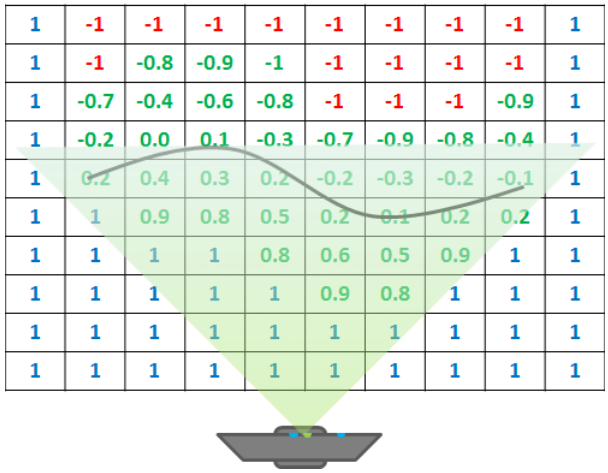


Figura 3.9: Funcionamento do TSDF. O valor "zero"corresponde à superfície do objeto.

O *KinectFusion* trabalha com um TSDF restrito, mas Whelan *et al.* [66] propuseram uma técnica para manipular dinamicamente o volume, apresentando um impacto mínimo no desempenho do rastreamento e reconstrução da superfície.

3.5 Renderização do Modelo

A renderização do modelo do nosso sistema é feita com *raycast* [45] em paralelo, pois ele combina perfeitamente com a metodologia do TSDF. Cada *thread* de GPU escolhe um ponto de partida e a direção do raio, percorrendo os *voxels* no caminho até encontrar o cruzamento em zero [44], *i.e.* quando houver troca de sinal positivo para negativo ou vice-versa, e extrair a posição da superfície implícita. O ponto da superfície final é calculado com uma simples interpolação linear, usando o valor positivo e negativo entre o cruzamento em zero. Como o gradiente de *voxels* é ortogonal para a interface da superfície, podemos calcular a normal diretamente com o TSDF.

O modelo iterativo possui uma alta qualidade, já que tem a informação de múltiplas vistas, atenuando os problemas de rastreamento e reduzindo os erros no alinhamento.

CAPÍTULO 4

MÉTODOS DE SUPER-RESOLUÇÃO

Uma preocupação em utilizar os dados adquiridos por dispositivos de baixo custo, tais como o Kinect, é devida ao elevado nível de ruído neles presente. Técnicas para o tratamento das imagens de profundidade vêm sendo estudadas desde os dispositivos tempo-de-voo.

Neste capítulo serão apresentados os dois tipos de pesquisa relacionados à super-resolução de imagens de profundidade. Na seção 4.1 serão detalhados os métodos que utilizam apenas a informação de profundidade, e na seção 4.2 as técnicas que utilizam a cor no processo de super-resolução. Na seção 4.3 apresentamos o método escolhido para o nosso pipeline (Figura 4.1).

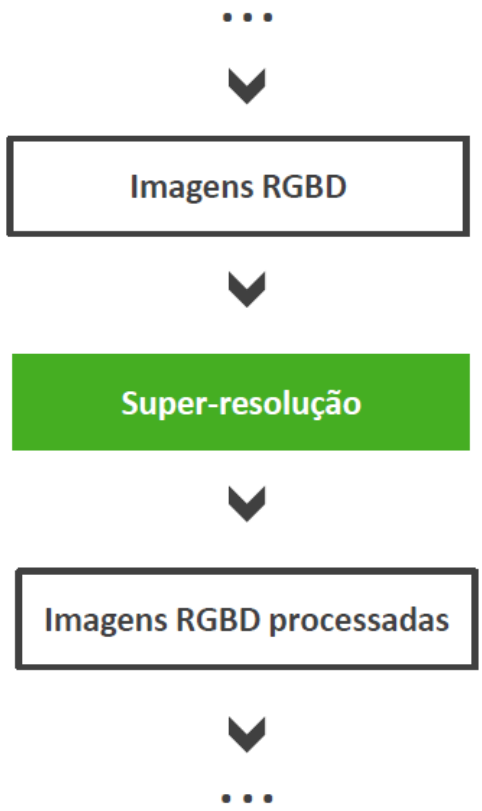


Figura 4.1: Segunda parte do nosso *pipeline*. Esta etapa é responsável por diminuir o ruído e aumentar a qualidade das imagens de profundidade.

4.1 Imagens de Profundidade

Uma das técnicas mais eficientes para a diminuição de ruído é o filtro bilateral de Tomasi e Manduchi [62], descrito na seção 3.2. Ele é amplamente utilizado por ser um método simples que utiliza a informação espacial para suavizar uma única imagem e preservar as suas bordas.

Outra abordagem que minimiza o ruído de diferentes imagens é a super-resolução temporal, utilizado em trabalhos como de Schuon *et al.* [55] e Cui *et al.* [14]. Inicialmente temos uma sequência $i = 1, ..., n$ imagens de profundidade $D_i \in \mathbb{R}^{n \times m}$ com resolução $n \times m$. Estas imagens são distribuídas em $l = 1, ..., k$ subconjuntos, onde cada subconjunto c_l é composto de m imagens subsequentes começando pelo índice $\rho(l)$, ou seja, o subconjunto $c_l = (D_{\rho(l)}, ..., D_{\rho(l)+m})$. Em cada subconjunto, as imagens são alinhadas à imagem central do subconjunto utilizando fluxo optico. É importante que este alinhamento seja preciso para que a super-resolução se mantenha correta. Em seguida, é calculada uma única imagem de profundidade com resolução β vezes maior que a original utilizando *LidarBoost*, técnica que extrai uma imagem de alta resolução para cada subconjunto. O processo é feito de forma otimizada ao comparar e escolher os pixels mais confiáveis que farão parte da nova imagem. Logo, após todo o processo, é criado k novas imagens $D_k \in \mathbb{R}^{\beta n \times \beta m}$.

Gevrekci e Pakin [21] propuseram uma técnica de projeção em conjuntos convexos (POCS, ou *Projection Onto Convex Sets*). Similar ao método mencionado anteriormente, ele divide o conjunto de imagens em vários subconjuntos menores e cria uma imagem para cada uma delas. Porém, as imagens dentro de um subconjunto diferem na intensidade da profundidade, ou seja, as imagens são de um mesmo ponto de vista mas com diferentes níveis de profundidade, eliminando desta forma os dados contaminados por saturação ou ruído. Esta técnica espacial e temporal foi empregado com sucesso para super-resolução de imagens 2D, e o seu trabalho é o primeiro a aplicá-la para imagens 3D. Os resultados são imagens de profundidade mais nítidas.

Aodha *et al.* [35] propuseram uma nova técnica para aumentar sinteticamente a qualidade de uma imagem de profundidade sem a necessidade de imagens adicionais. Seu

trabalho é o primeiro a explorar a técnica de "remendar", ou *patching*.

Os dados disponíveis estão na forma de uma imagem de profundidade de baixa resolução A, que é criado a partir de uma imagem desconhecida de alta resolução B^* por uma técnica de subamostragem (ou *downsampling*) $\downarrow_{d_a}$. Utilizando a coleção de imagens $A = a_1, ..., a_n$ não sobrepostos e de tamanho $M \times M$, é produzido as remendas normalizadas α_i. O objetivo é utilizar imagens deste "banco de dados de remendos" para formar uma nova imagem plausível B que irá substituir a imagem original. Infelizmente o seu método não explora o contexto temporal, o que leva a produzir resultados aceitáveis ao invés de precisos.

4.2 Imagens RGBD

As descontinuidades de profundidade numa cena estão relacionadas com as mudanças de cor e brilho da imagem. Alguns trabalhos exploram esta característica e combinam as duas informações para melhorar a informação de profundidade. Outro fato importante é que a tecnologia das câmeras coloridas oferece uma resolução muito superior ao 3D. Como exemplo, o trabalho de Garro *et al.* [20] re-projeta a nuvem de pontos do mapa de profundidade em uma imagem de cor segmentada e em seguida refina a imagem de profundidade com a proposta técnica de interpolação de imagens.

Na abordagem de Yang *et al.* [67], cada imagem de profundidade D_i passa por um refinamento iterativo. O volume de custo C é construído baseado no atual mapa de profundidade. Ao aplicar C_i e o filtro bilateral em D_i, é criado um novo volume de custo C_i^*. Os pixels na imagem de cor também são utilizados no filtro bilateral para contribuir na preservação das bordas. Baseado no volume C_i^*, a imagem refinada D_{i+1} é gerada e utilizada no próximo ciclo. O processo termina após n iterações, e pelo fato deste método ser um processo iterativo, o seu custo computacional é alto.

Diebel e Thrun [17] descrevem como aplicar o Campo Aleatório de Markov (*Markov Random Field* ou MRF), um modelo muito utilizado em aplicações de processamento de imagem de baixo nível, para produzir imagens de profundidade de alta resolução. O *Joint Bilateral Upsampling* (JBU) de Kopf *et al.* [31] procura aplicar um filtro espacial (geral-

mente o filtro Gaussiano) na imagem de profundidade de baixa resolução D^* enquanto um filtro similar é aplicado na imagem de cor de alta resolução I^*. Seja i e j as coordenadas do pixel de I^*, e $i_\downarrow$ e $j_\downarrow$ a possível coordenada correspondente na imagem D^*, a solução de amostragem (ou *upsampling*), dado um determinado tempo k, é adquirida com a seguinte fórmula:

$$D_k(i) = \frac{1}{W_p}\sum_{j_\downarrow \in \Omega} f(\| i_\downarrow - j_\downarrow \|) g(\| I_k^*(i) - I_k^*(j) \|) D_k^*(j_\downarrow) \tag{4.1}$$

A fórmula acima é similar à do filtro bilateral (equação 3.1). Sendo assim, f representa o filtro espacial, g o filtro da informação de profundidade, e W_p um fator de normalização. Esta técnica é interessante para as imagens de profundidade, pois permite encontrar a disparidade entre a informação de profundidade de baixa resolução e a informação de cor de alta resolução.

Chan *et al.* [11] propuseram uma abordagem inspirada no filtro bilateral, o *Noise-Aware Filter for Depth Upsampling* (NAFDU). O NAFDU procura aumentar a resolução espacial da imagem de profundidade dos dispositivos ToF utilizando a cor, e evita que esta cor afete a geometria em regiões onde a geometria deveria estar suavizada. Ele alcança esse resultado modificando o peso α da intensidade de cor e profundidade:

$$\begin{aligned} D_k(i) &= \frac{1}{W_p}\sum_{j_\downarrow \in \Omega} f(\| i_\downarrow - j_\downarrow \|) \omega_{ij} D_k^*(j_\downarrow) \\ \omega_{ij} &= \alpha(\Delta) g(\| I_k^*(i) - I_k^*(j) \|) + (1 - \alpha(\Delta)) h(\| D_k^*(i_\downarrow) - D_k^*(j_\downarrow) \|) \end{aligned} \tag{4.2}$$

onde g e h são os filtros Gaussianos de intensidade de cor e profundidade, respectivamente. Aumentando o α, o filtro se comporta como um JBU.

O sistema apresentado por Snavely *et al.* [60] inicialmente segmenta as imagens de profundidade utilizando como limiar a informação RGBD. O processo é simples, visto que as imagens são capturadas em uma cena com fundo preto. Em seguida é realizada

o tratamento da informação de profundidade. Os buracos são preenchidos com a técnica de interpolação a partir das bordas do buraco, e o ruído é reduzido com o filtro bilateral. A direção das normais e o princípio da curvatura de cada pixel são calculadas utilizando a abordagem de Hameiri e Shimshoni [23]. As outras etapas deste sistema são utilizadas para criar vídeos com informação 3D.

Um aprimoramento do trabalho de Snavely *et al.* é apresentado por Richardt *et al.* [49]. O autor apresenta uma abordagem que faz uso de um filtro espaço-temporal para reduzir ruído e aumentar a amostragem das imagens de profundidade.

Inicialmente, as câmeras são calibradas utilizando o mesmo processo descrito na seção 2.1. Em seguida os dados de profundidade são alinhados e reprojetados à imagem de cor.

Os pixels próximos às bordas da informação de profundidade são removidos utilizando o filtro 3×3 de Sobel por não serem confiáveis.

Na próxima etapa os buracos são preenchidos utilizando uma abordagem com n níveis de resolução, onde 0 representa o nível mais fino e a $n-1$ o mais inferior. Cada k nível tem duas entradas e uma saída, todos com a mesma resolução: a imagem de cor I_k e o mapa de profundidade D_k são as entradas, e o mapa de profundidade com os buracos preenchidos, ou F_k, é a saída. O nível mais inferior, $n-1$, preenche pixels inválidos do mapa D_{n-1} baseado na imagem I_{n-1} usando o filtro JBU, resultando no mapa F_{n-1}.

Para os níveis $k = 0, ..., n-2$, o filtro funciona da seguinte maneira:

(1) A resolução das imagens I_k e D_k são reduzidos selecionando o g pixel ao longo do eixo x e y, resultando nas imagens I_{k+1} e D_{k+1}. Estas novas imagens serão utilizadas no nível abaixo, ou $k+1$, que retorna a imagem preenchida F_{k+1} após todo o processo recursivo acabar.

(2) Os níveis mais inferiores preencheram recursivamente todos os pixels inválidos D_{k+1} e F_{k+1}. Estes novos pixels serão extraídos por amostragem (upsample) para uma nova grade de pixels que será utilizado no mapa D_k. Isto resulta em uma novo mapa U_k com os valores preenchidos em cada g pixel.

(3) O mesmo filtro JBU é aplicado nos mapas U_k e na imagem I_k para preencher os pixels inválidos de D_k. Este processo remove os artefatos presentes nos pixels extraídos

por amostragem.

Os parâmetros utilizados pelo autor foram $n = 3$, $g = 2$. Para o filtro bilateral: $\sigma_s = 10$ e $\sigma_r = 0.05$.

Finalmente, um filtro espaço-temporal, que incorpora a informação dos quadros anteriores, diminui o ruído e melhora o mapa de profundidade explorando a coincidência entre cor e bordas de profundidade.

Este método produz mapas de distância de alta qualidade. No entanto, em áreas de rápido movimento, o fluxo óptico tende a ser instável devido ao borrão de movimento. Isto pode ser evitado com a captura numa alta taxa de quadros.

Matsuo *et al.* [37] apresentam uma variante de filtro bilateral, chamada *Weighted Joint Bilateral Filter*(WJBF), e o combinam com um novo filtro criado por eles, o *Slope depth compensation filter*(SDCF). O objetivo do primeiro é projetar o contorno da imagem de profundidade no contorno da imagem de cor. A melhoria em relação ao filtro original está no mapa de peso R, que controla a influência do valor de profundidade no pixel e é fixo durante toda a filtragem:

$$D_k(i) = \frac{1}{W_p}\sum_{j\in\Omega} f(\parallel i-j \parallel) g(\parallel I_k(i) - I_k(j) \parallel) R_j I_k(j) \tag{4.3}$$

$$R_j = \sum_{j_\downarrow \in \Omega'} M_j \cdot f(\parallel j - j' \parallel) g(\parallel I_k(j) - I_k(j') \parallel) e^{-\frac{1}{2}(\frac{\|D_j - D_{j'}\|}{\sigma})} \tag{4.4}$$

onde j' é a coordenada do *pixel* que serve como suporte à coordenada central perto do *pixel* j, e σ representa o peso do filtro Gaussiano. M_j é a máscara *speckle* que possui peso 0 caso as regiões possuam *speckle*, ou 1 caso contrario. Sendo assim, pesos altos correspondem a pixels com valores de profundidade confiáveis. Além de proporcionar uma ótima redução de ruído, o mapa de pesos corrige os erros relacionados aos borrões nas bordas. Como a diferença entre o objeto capturado e o fundo é muito grande para imagens de profundidade, qualquer filtro Gaussiano irá balancear os valores, causando declives entre as bordas do objeto e o fundo do cenário. O filtro SDCF remove esses declives, substituindo estes valores, após o WJBF, por outros da imagem original:

$$\begin{aligned} D_i^{SDCF} &= D_k^{INICIAL} \\ \text{tal que } k &= \arg\min_{j \in \omega} \parallel D_i^{WJBF} - D_j^{INICIAL} \parallel \end{aligned} \tag{4.5}$$

onde D^X são as imagens de profundidade e $X \in \{INICIAL, WJBF, SDCF\}$, $\| \cdot \|$ representa a função de normal L2, i a posição do *pixel* alvo, j é a posição do *pixel* de apoio, e k é a posição do *pixel* o qual aponta o mínimo da função.

4.3 Método escolhido

No início do projeto, foi escolhido o trabalho de Richardt *et al.* [49] como o método de super-resolução para o nosso pipeline. Os resultados com este método demonstraram um alto nível de detalhe e redução de ruídos, principalmente utilizando imagens coloridas de maior resolução. Mas como o foco do autor era criar vídeos 3D com uma variedade de efeitos visuais, não havia informações à respeito desta super-resolução aplicada na reconstrução de modelos 3D. Ao utilizá-lo em nosso pipeline, notamos a dificuldade que o alinhamento encontrava ao combinar as novas imagens. A justificativa é que o método depende da informação de intensidade luminosa, logo, diferentes resultados podem surgir para cada imagem de uma mesma cena, resultando em pequenas deformações para cada uma.

Um trabalho recente que chamou a atenção por sua simplicidade foi o filtro WJBF de Matsuo *et al.* [37]. O processo é rápido, proporciona resultados precisos, e preserva muito bem a região das bordas. A principal vantagem é a remoção dos pontos correspondentes às áreas de declive entre o objeto e o fundo, também presente no método de amostragem de Richardt *et al.* [49]. No capítulo 6, comparamos o método de amostragem com a filtragem de Matsuo *et al.* e justificamos o uso deste último filtro nesta etapa do pipeline.

CAPÍTULO 5

RECONSTRUÇÃO 3D DE ALTO CUSTO

Na reconstrução 3D de alto custo utilizamos técnicas de maior processamento e tempo computacional, técnicas que comprometeriam o desempenho dos algoritmos de tempo real (seção 3). O objetivo desta etapa é unir todas as imagens de alta qualidade, produzidas no processo de super-resolução (seção 4), e reconstruir o modelo digital de alta qualidade.

Neste trabalho, escolhemos o sistema desenvolvido por Vrubel *et al.* [63] pois este vem sendo empregado com sucesso em diversos projetos de preservação de patrimônios culturais que demandam alta precisão para os seus modelos digitais. Os métodos de alinhamento, integração, preenchimento de buracos e geração do modelo 3D serão brevemente detalhados nas seções 5.1, 5.2, 5.3 e 5.4, respectivamente.

5.1 Alinhamento

Esta etapa tem o mesmo objetivo do alinhamento mencionado na seção 3.3: posicionar todas as imagens em um mesmo sistema de coordenadas cartesianas. O sistema de Vrubel *et al.* também utiliza o ICP, mas antes realiza um processo de pré-alinhamento automático de vistas [22, 57] que elimina em 95% o processo manual de aproximar duas imagens para o posterior alinhamento. O método utiliza *Speeded Up Robust Features* (SURF) para detectar correspondências e usá-las no alinhamento propriamente dito.

Este pré-alinhamento é ideal para imagens capturadas em ângulos muito distintos, o que não acontece com as imagens capturadas com um sensor de tempo real. Em nossos experimentos, o pré-alinhamento atrapalha o resultado do alinhamento. No entanto, ainda o utilizamos quando queremos aumentar a confiabilidade no alinhamento entre duas imagens não consecutivas.

A variante do ICP utilizado é dividida em duas etapas [63]. A primeira consiste no registro par-a-par utilizando métrica ponto-ao-plano de Chen e Medioni [12], que nos garante

máxima convergência. A segunda é o algoritmo de Pulli [47], que trata do alinhamento global e evita o problema da distribuição dos erros acumulados pelo processo iterativo de alinhamento. Ou seja, diferente do sistema em tempo real, Vrubel *et al.* [63] utilizam uma variante com foco inicial na precisão do alinhamento e em seguida no desempenho.

5.2 Integração

O processo de integração combina todas as imagens alinhadas para gerar um único modelo 3D. Entre as abordagens estudadas, o método volumétrico foi escolhido por funcionar mesmo na presença de imagens com pouca resolução. Os algoritmos volumétricos também provaram ser eficientes em projetos de preservação digital, como a digitalização dos Grandes Budas [26], das obras do escultor Michelangelo [33] e do escultor Aleijadinho [2].

Na primeira versão do sistema de Vrubel *et al.* [63], utilizava-se o VRIP [15] para criação do SDF do objeto integrado. Esta metodologia é similar à representação volumétrica do sistema em tempo real (seção 3.4), que calcula para cada *voxel* do volume a distância com sinal do centro do *voxel* em relação à superfície. O que muda nesta etapa é a precisão dos resultados ao manipular o tamanho dos *voxels*. *Voxels* menores proporcionam maior fidelidade ao custo de mais tempo de processamento.

O sistema adotou recentemente outro algoritmo volumétrico chamado IMAGO Volumetric Integration Algorithm (IVIA) [29], que combina o VRIP e *Consensus Surfaces* [65]. O objetivo desta fusão é superar as falhas que cada método apresenta. Na primeira fase é criada uma representação volumétrica do modelo com o algoritmo VRIP modificado. Em seguida é aplicada a integração definitiva utilizando a representação da primeira fase para detectar e desconsiderar as discrepâncias durante a construção do modelo volumétrico, resultando em um método mais preciso e com menos artefatos.

5.3 Preenchimento de Buracos

O objetivo desta etapa é inferir informação nas regiões com pouco ou nenhum dado, resultando em um modelo "fechado". Dependendo da complexidade do objeto é impossível

a reconstrução completa com as imagens do sensor, devido aos limites da tecnologia de captura (seção 3.1). Logo, esta etapa se torna fundamental e contribui para possíveis aplicações futuras como a criação de réplicas e modelos.

O *space carving* é uma técnica simples e eficiente apresentada por Curless e Levoy [15]. O método classifica os pontos do volume em três regiões: as vazias, regiões que correspondem aos *voxels* localizados entre a origem da captura e a superfície; os próximos da superfície; e os não vistos, que não entram em nenhuma classificação anterior. A fronteira entre *voxels* vazios e não vistos corresponde às regiões onde o algoritmo insere superfícies para eliminar os buracos.

Sagawa e Ikeuchi [53] preenche os buracos através da garantia de consistência dos sinais do SDF. Sem essa consistência, os resultados da propagação da superfície não são satisfatórios, uma vez que as superfícies próximas aos buracos são muito instáveis.

A abordagem de Davis *et al.* [16] é baseada em um processo de difusão que lida com topologias difíceis, cria superfícies suaves e opera sobre representações volumétricas. O método também utiliza a informação do *space carving* para detectar os espaços vazios ao redor do objeto.

O sistema adotado utiliza a abordagem de Davis *et al.* [16], pois além das vantagens apresentadas, opera naturalmente sobre a representação volumétrica criada na etapa anterior. Assim, temos a garantia de que os buracos serão fechados e que as mudanças na superfície serão menores que os ruídos relacionados ao sensor.

5.4 Geração do modelo 3D

O processo de geração da malha é trivial quando comparado às etapas anteriores. A estratégia mais utilizada para a representação da superfície é por meio da criação da malha triangular. Vrubel *et al.* [63] utilizam a abordagem mais popular, o *Marching Cubes*, proposto por Lorensen *et al.* [34].

Os triângulos criados com o *Marching Cubes* dependem da magnitude dos *voxels* nos vértices de cada cubo. Isto significa que uma interpolação linear é necessária em cada aresta para descobrir a posição real da superfície.

A malha é criada ao percorrer ordenadamente os *voxels* do volume. Quando o valor do SDF é igual a zero, significa que o vértice foi encontrado; valores positivos representam as regiões fora da superfície; e valores negativos são os valores dentro do objeto. É importante destacar que a informação só será coerente se os dados na grade volumétrica representarem a superfície implícita do objeto.

A limitação do *Marching Cubes* é a distribuição homogênea dos *voxels*, ou seja, todos os *voxels* devem ter o mesmo tamanho, o que pode afetar o resultado para objetos muito complexos. Entretanto, a sua simplicidade e compatibilidade com os dados volumétricos, proveniente das etapas anteriores, fazem com que o *Marching Cubes* seja uma ferramenta poderosa na criação de modelos tridimensionais.

CAPÍTULO 6

EXPERIMENTOS

Uma das metas deste trabalho é o estudo das ferramentas e sistemas que trabalhem com os sensores de captura em tempo real, colaborando na criação de soluções que integrem o sensor e suas informações ao *pipeline* desenvolvido por [63]. Outro objetivo foi a avaliação de técnicas existentes de super-resolução de forma a melhorar a qualidade das imagens do sensor e, por conseguinte, melhorar os modelos 3D reconstruídos.

O *pipeline* proposto visa combinar a captura em tempo real com a posterior reconstrução de alta qualidade, criando um sistema vantajoso para alguns campos de pesquisa, por exemplo:

- Na robótica, um braço mecânico ou robô autoguiado poderia capturar automaticamente o ambiente, *e.g.* um prédio histórico ou sítio arqueológico. Após este processo, seria possível reconstruir o ambiente com uma qualidade superior;
- Na biometria, um modelo de maior qualidade poderia aumentar a acurácia do reconhecimento facial;
- Na medicina, uma cirurgia usaria o modelo de alta qualidade deste *pipeline* para garantir a exatidão das operações realizadas via um braço robótico.

Os algoritmos foram implementados com a linguagem de programação C++. Além disso, utilizamos várias bibliotecas para auxiliar o desenvolvimento de cada uma das etapas, que serão mencionadas nas próximas seções. O computador utilizado para o projeto possui as seguintes especificações: uma placa gráfica *GeForce GTX 670*, processador *Intel Core i7-3820 3.6GHz* e 16GB de RAM.

Os detalhes e resultados de cada etapa serão exibidos nas próximas seções. A Seção 6.1 apresenta a etapa de modelagem geométrica em tempo-real; a Seção 6.2, o método de

super-resolução; a Seção 6.3, os modelos do processo final de reconstrução; e a Seção 6.4 explica as vantagens e limitações do nosso método.

6.1 Modelagem geométrica em tempo real

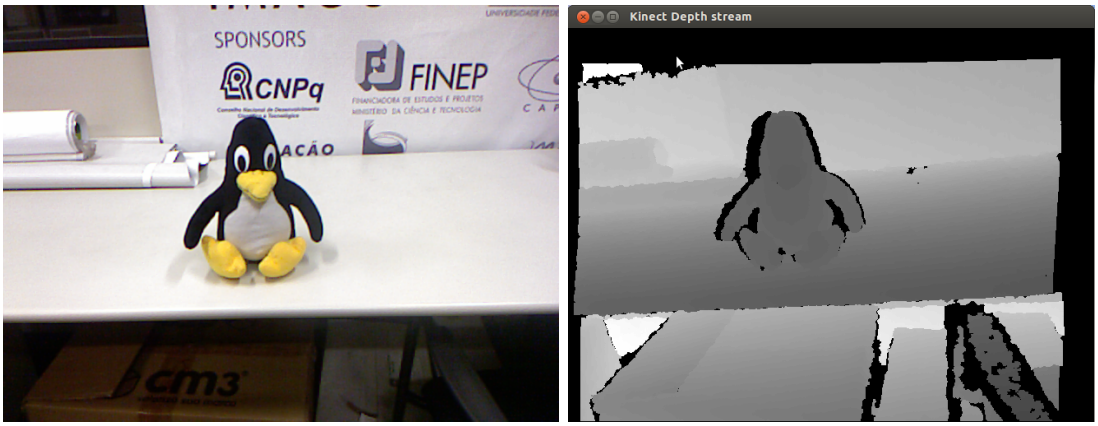
Nesta etapa modificamos o sistema *open source* de modelagem geométrica *KinFu*, desenvolvido pelo projeto PCL para plataformas paralelas CUDA. Foi mantida a interface do programa (figura 6.1) bem como a estrutura principal do programa, responsável pela reconstrução em tempo-real da cena. No entanto, criamos uma etapa adicional para o armazenamento, também em tempo-real, após o processo de alinhamento. Desta maneira, as informações de profundidade são salvas com o filtro bilateral aplicado e com a informação de profundidade z já calculada em metros (equação 3.4).

A resolução utilizada para o projeto foi 640×480 para profundidade e cor. Como o programa não tinha suporte para captura de imagens coloridas de 1280×960, foi adicionada esta opção no *KinFu*. Vale lembrar que aumentar a resolução de cor não melhora o resultado do modelo criado. O objetivo é utilizar imagens com esta resolução em algumas técnicas de super-resolução, demonstrado na próxima seção. Infelizmente, o desempenho do programa diminui quando este modo de captura é ativado (de 60 fps para 30-45 fps). Como há um intervalo maior entre cada quadro, o alinhamento se perde com mais facilidade quando o sensor está em movimento. É necessário um ritmo mais desacelerado para a captura.

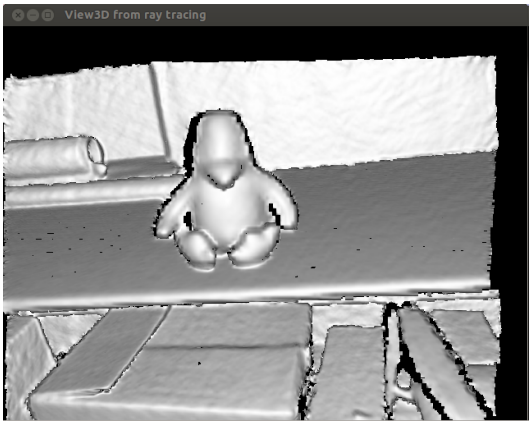
Entre as bibliotecas utilizadas, estão: OpenNI[1] para habilitar o acesso e comunicação do dispositivo Kinect com o computador; *Eigen*[2] para algebra linear; FLANN[3] para a busca dos pontos vizinhos mais próximos; VTK[4] para visualização e computação gráfica; e o *Boost*[5], um pacote de bibliotecas que proporciona suporte às mais variadas tarefas como algebra linear, *multithreading*, processamento de imagem e outros.

As imagens são salvas no formato *.pgm* e *.ppm* para profundidade e cor respectiva-

[1] http://www.openni.org/
[2] http://eigen.tuxfamily.org/
[3] http://www.cs.ubc.ca/ mariusm/index.php/FLANN/FLANN
[4] http://www.vtk.org/
[5] http://www.boost.org/



(a) (b)



(c)

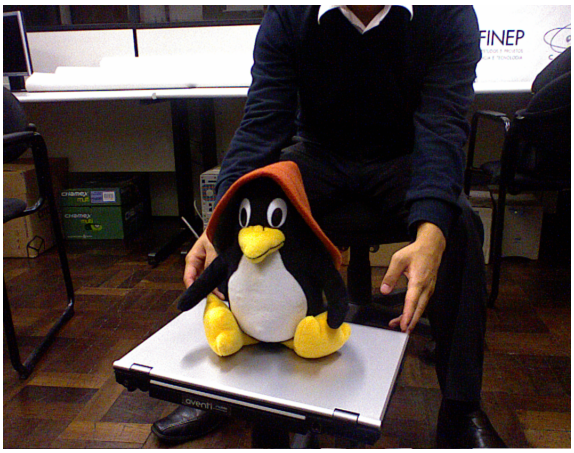
Figura 6.1: Interface do sistema. (a) Fotografia da cena. (b)(c)(d)Interface do programa. A imagem (b) é o mapa de profundidade e (c) o resultado da reconstrução em tempo real.

mente. Adotamos esta metodologia por ser simples e rápida (menos de $2ms$ para salvar cada imagem) sem comprometer com o desempenho do sistema.

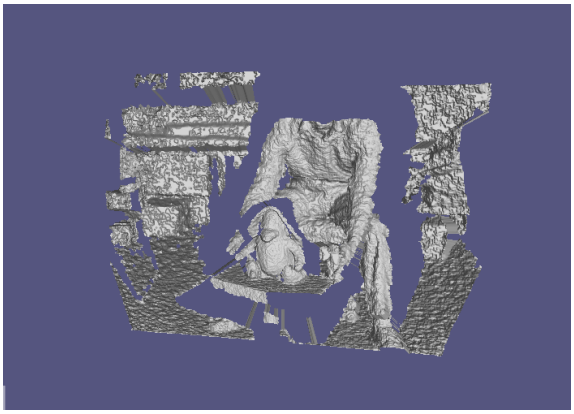
6.2 Métodos de super-resolução

Neste trabalho, foram mencionados dois tipos de método de super-resolução: por profundidade (Seção 4.1) e por profundidade com cor (Seção 4.2). Dentre os dois, foi observado que as abordagens que utilizam a informação da cor oferecem um resultado superior por adicionar mais um tipo de informação no processo de melhoria.

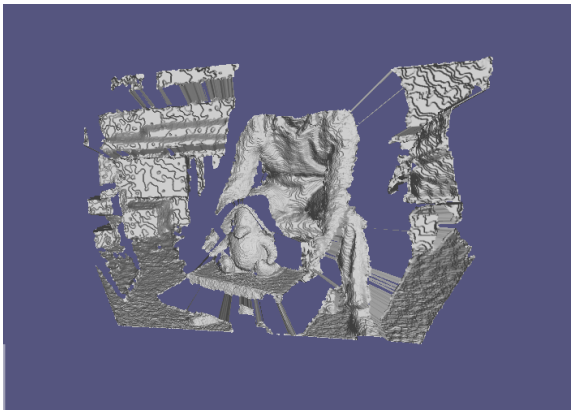
Foram estudadas duas abordagens recentes com o objetivo de escolher uma delas para fazer parte do *pipeline proposto*. A primeira foi o método de amostragem de Richardt *et al.* [49] e a segunda o filtro bilateral WJBF+SDCF de Matsuo *et al.* [37].



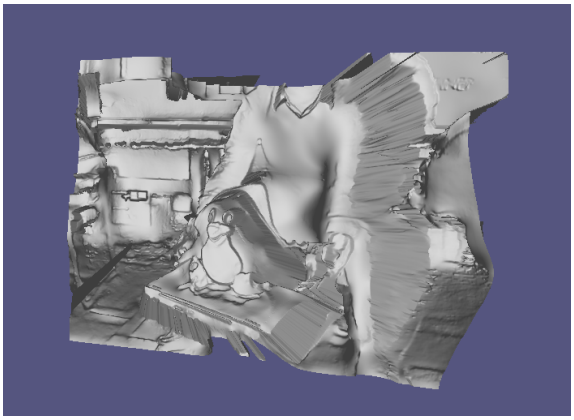
(a)



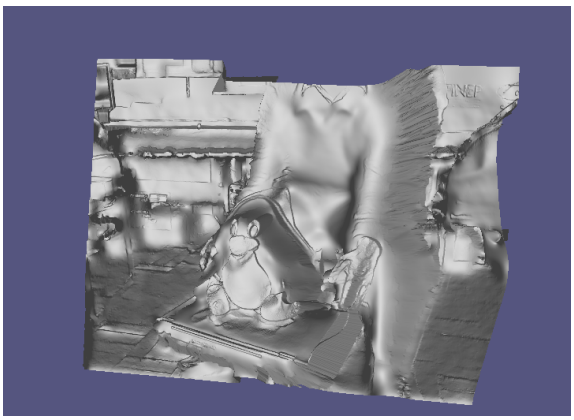
(b)



(c)



(d)



(e)

Figura 6.2: Demonstração dos resultados da super-resolução de Richardt *et al.* e Matsuo *et al.*. A visualização foi feita utilizando vértices e faces (a)Fotografia da cena (b)Imagem de profundidade original com resolução 640×480 e 221.238 pontos. (c)Método de Matsuo *et al.* com resolução 640×480 e 222.151 pontos. (d)Método de Richardt *et al.* com resolução 640×480 e 226.919 pontos. (e)Método de Richardt *et al.* com resolução 1066×833 e 887.978 pontos.

O algoritmo de Richardt *et al.* [49] foi desenvolvido em C++ e C# e utiliza várias bibliotecas de processamento desenvolvidas pelo autor. Richardt inclusive desenvolveu o formato *.raw* para imagens de entrada e saída, tanto para profundidade e cor. As informações quanto ao tipo, tamanho e resolução estão contidos no cabeçalho dentro de cada imagem. O sistema de Matsuo *et al.* [37] foi desenvolvido em C++ com SIMD e paralelização por *Threading Building Blocks*[6] (TBB). Os arquivos de entrada são os tipos mais conhecidos (*.jpg*, *.jpeg*, *.png*, *.pgm*, *.ppm*) e de saída o *.png*.

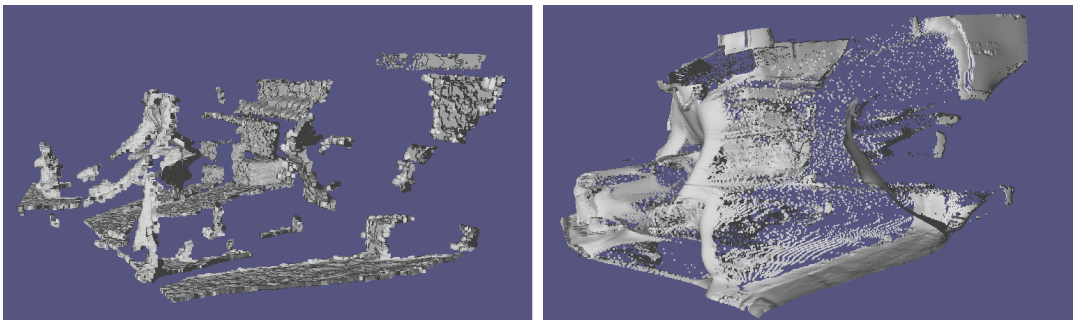
Ambos os métodos foram testados com imagens de resolução 640×480 para cor e profundidade. O diferencial do método de amostragem é que as imagens de cores podem ter uma resolução maior para melhorar o seu resultado. Dito isso, o método foi testado com imagens de 1280×960 e após o processo de super-resolução, ambos os tipos de imagens receberam uma nova resolução de 1066×833.

Analisando a figura 6.2, percebe-se a melhoria dos métodos em relação ao modelo 3D original 6.2(b). Quanto ao nível de ruído removido, as técnicas de amostagem 6.2(d) e 6.2(e) prevalecem. Neste método, a profundidade é reprojetada na imagem da cor com o objetivo de melhorar o mapa de profundidade em si. O filtro WJBF 6.2(c), por outro lado, reprojeta apenas o contorno da profundidade no contorno da cor com o objetivo de preservar as bordas, afetadas pelo processo de filtragem.

Em nível de detalhe geométrico, o filtro WJBF apresenta um resultado superior ao da amostragem. Embora a redução de ruído do método de amostragem seja surpreendente, o alto nível de suavização acarreta na perda da qualidade da geometria do objeto, como observado em 6.2(d) e 6.2(e). Outro fator importante é que o filtro WJBF é combinado com SDCF, um método que remove artefatos causados pelo processo de filtragem, presentes entre o objeto e cena (figura 6.3).

Concluímos que as duas técnicas de super-resolução possuem estratégias muito distintas. A amostragem utiliza a imagem de cor para remover os ruídos e aumentar a qualidade do mapa de profundidade, enquanto que o WJBF procura usar a cor para diminuir os efeitos do borramento provenientes do filtro bilateral. O fator decisivo para a escolha

[6] http://threadingbuildingblocks.org/



(a) (b)

Figura 6.3: Demonstração dos artefatos gerados no processo de filtragem. (a)Método de Matsuo *et al.*. (b)Método de Richardt *et al.*. A técnica SDCF de Matsuo *et al.* evita a criação de artefatos, presente na técnica de amostragem de Richardt *et al.*.

do método foi em termos de qualidade geométrica. O uso da intensidade luminosa na técnica de amostragem causa pequenas deformações na geometria de cada imagem, proveniente das variações luminosas do ambiente, dificultando o processo de alinhamento e reconstrução. O filtro WJBF além de amenizar o ruído, remove os artefatos causados pelo processo de filtragem.

6.3 Reconstrução 3D de alto custo

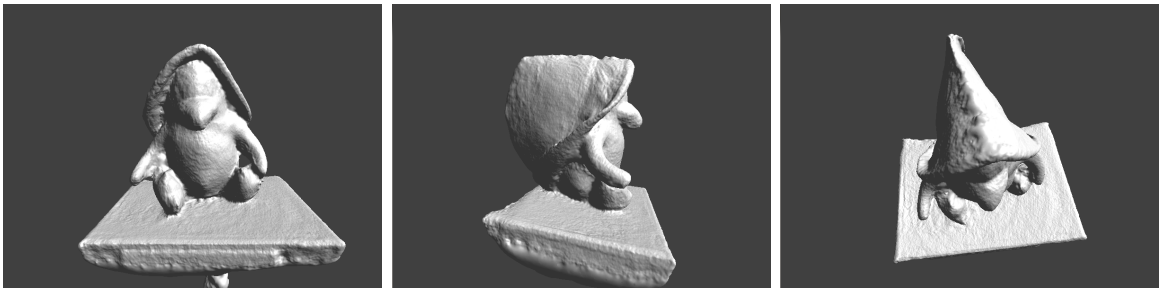
Nesta etapa, apresentamos o resultado do *pipeline* proposto. Após a captura em tempo real e filtragem com super-resolução, aplicamos a abordagem de Vrubel *et al.* [63] para reconstruir o modelo 3D de alta qualidade. Comparamos o resultado dos experimentos envolvendo os seguintes *pipelines*:

- Pipeline proposto (*KinFu* e Matsuo et al. e Vrubel et al.);
- *KinFu* e Richardt et al. (resolução 640×480) e Vrubel et al.;
- *KinFu* e Richardt et al. (resolução 1066×833) e Vrubel et al..

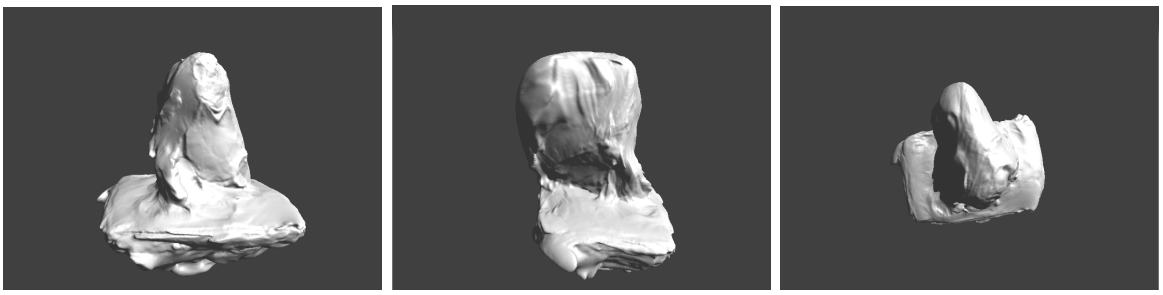
Devido à falta de um computador portátil com as especificações necessárias para executar o sistema proposto, nossos experimentos se limitaram aos objetos disponíveis no laboratório. O *pipeline* funciona para qualquer tipo de reconstrução, seja objeto ou cenário. Entretanto, testamos apenas com objetos menores para avaliar a qualidade geométrica do

sistema. Selecionamos um desses objetos, o "pinguim de pelúcia", para fazer a avaliação e comparação dos métodos.

No modelo criado com a combinação do *KinFu*, super-resolução de Richardt et al. e reconstrução de Vrubel *et al.*, notamos as grandes diferenças em relação ao modelo criado com o método proposto (figura 6.4). A limitação do método 6.4(b) é a deformidade causada pela luz ambiente, que altera significativamente os resultados da super-resolução. Como mencionado, essa diferença também afeta o processo de alinhamento, causando um aspecto mais ruidoso em algumas regiões da figura.



(a)

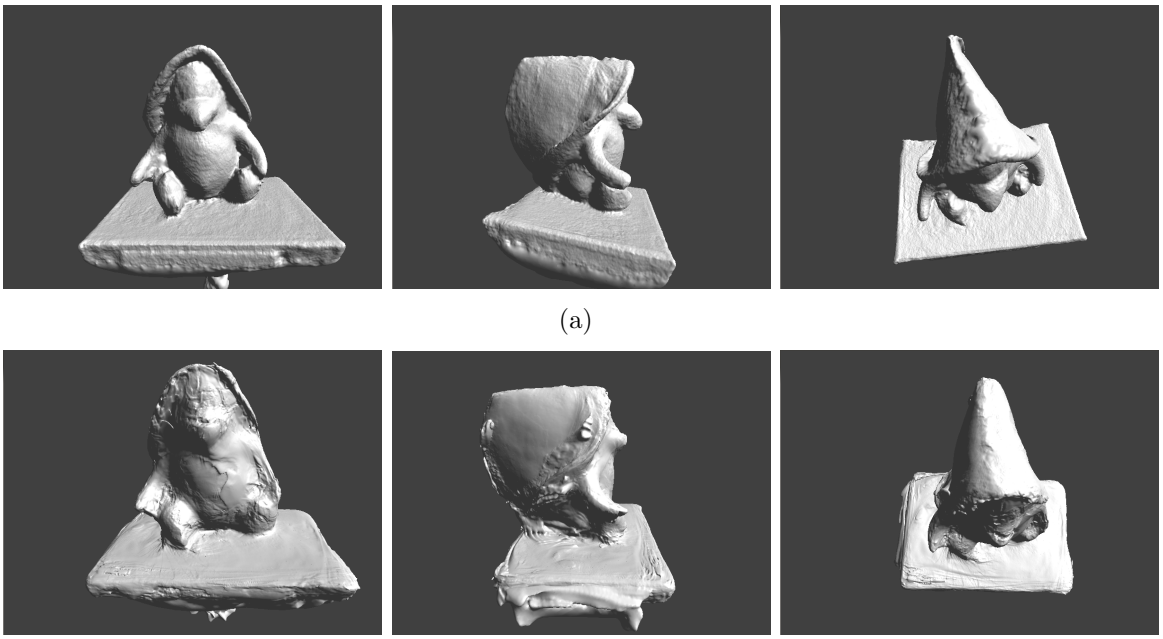


(b)

Figura 6.4: Primeira comparação dos resultados. (a)Modelo gerado pelo pipeline proposto. (b)Modelo gerado pelo *KinFu* com super-resolução de Richardt *et al.* (640×480) e reconstrução de Vrubel *et al.*

No próximo experimento, utilizamos as imagens processadas com a resolução de 1066×833, criado com o método de Richardt *et al.* (figura 6.5(b)). Esta super-resolução melhorou ainda mais a imagem de profundidade, chegando a triplicar a densidade da nuvem de pontos em relação à resolução 640×480, embora as distorções ainda sejam evidentes.

Foram capturadas cerca de 980 imagens RGBD do objeto, desses utilizamos 98 na etapa de reconstrução de alto custo por ser uma quantidade suficiente para a geração do



(a)

(b)

Figura 6.5: Segunda comparação dos resultados. (a)Modelo gerado pelo pipeline proposto. (b)Modelo gerado pelo *KinFu* com super-resolução de Richardt *et al.* (1066 × 833) e reconstrução de Vrubel *et al.*

modelo. Foram criados cerca de 1.587.493 pontos para o nosso *pipeline*, cerca de 556.542 pontos para o método de amostragem com resolução de 640 × 480, e 1.374.254 pontos para a amostragem de 1066 × 833 de resolução.

6.4 Avaliação do método proposto

O processo para a criação do modelo 3D do método proposto neste trabalho é muito mais lento quando comparado a um sistema de modelagem geométrica em tempo real. No entanto, apresentamos algumas vantagens que o nosso sistema oferece em relação ao sistema de modelagem *KinFu*:

- O modelo 3D criado com o método proposto possui mais detalhes, pois é utilizado um processo volumétrico que trabalha com volumes menores, o que resulta em maior acurácia ao custo de mais memória e tempo computacional.

- O alinhamento é mais robusto e o *pipeline* oferece um controle melhor sobre as imagens utilizadas na reconstrução. Como a reconstrução do *KinFu* depende de

seu rastreamento, um alinhamento incorreto pode afetar todo o modelo antes do término da captura.

O *pipeline* proposto possui uma limitação, que é o fato de não possuir um algoritmo eficiente para remoção de fundo. Procuramos manualmente pelo limiar entre o objeto e a cena através da distância dos pontos de uma imagem. Em seguida aplicamos este mesmo limiar nas outras. Esta técnica só irá ser eficiente se a distância entre o sensor e o objeto permanecer similar para cada imagem. Uma abordagem interessante seria utilizar algum tipo de rastreamento com uma imagem 3D como referência. A estimativa da pose do objeto poderia ser feita utilizando as técnicas de amostragem de *Monte Carlo* [69] combinadas com as métricas conhecidas, como pontos cartesianos, cor e a normal da superfície.

CAPÍTULO 7

CONCLUSÃO

O objetivo deste trabalho foi a ampliação e melhoria do atual pipeline de reconstrução 3D de Vrubel *et al.* [63], expandindo o sistema para que também trabalhe com imagens adquiridas por um sensor de captura em tempo real.

Como a informação capturada pelo Kinect é muito ruidosa, também exploramos algumas técnicas de super-resolução para suprir a baixa qualidade das imagens. Testamos primeiramente a técnica de amostragem de Richardt *et al.* [49], que reduziu uma grande quantidade de ruído. No entanto, seu alto nível de filtragem cria artefatos e diminui a fidelidade da geometria. Optamos pelo filtro WJBF e SDCF de Matsuo *et al.* [37] por diminuir o efeito negativo do filtro bilateral e eliminar os artefatos próximos às bordas.

Concluímos que a melhor forma de produzir um resultado mais fidedigno dentro do escopo deste projeto, seria criar um novo pipeline que se resume a três etapas principais:

- A aquisição completa em tempo real utilizando o sistema *KinFu*;
- Aplicação do método de super-resolução de Matsuo *et al.* [37] nas imagens capturadas;
- Reconstrução 3D utilizando o sistema descrito em Vrubel *et al.* [49] nas imagens processadas.

O sistema de reconstrução em tempo real provou ser muito útil no processo de aquisição das imagens de cor e profundidade, e o modelo de baixa qualidade atualizado em tempo real é ideal para guiar o usuário às regiões que não foram capturadas. Em seguida, o método de super-resolução aumenta a resolução das imagens originais para uma posterior criação de modelos geométricos de alta qualidade.

Nossos experimentos indicaram um ganho elevado na qualidade geométrica da abordagem desenvolvida, no entanto, mais tempo precisa ser investido para melhorar ainda mais

o sistema. Pretendemos avaliar os resultados com outros métodos de super-resolução para conseguir melhores resultados geométricos. Outra contribuição futura seria incorporar o processo de geração de texturas utilizando uma câmera digital de alta resolução [1] para criar texturas de alta qualidade ao modelo final.

BIBLIOGRAFIA

[1] Beatriz T. Andrade, Olga R. P. Bellon, Luciano Silva, e Alexandre Vrubel. Digital preservation of Brazilian indigenous artworks: Generating high quality textures for 3D models. *Journal of Cultural Heritage*, 13(1):28–39, 2012.

[2] Beatriz T. Andrade, Caroline M. Mendes, Jurandir Santos Jr., Olga R. P. Bellon, e Luciano Silva. 3D preserving XVIII century barroque masterpiece: Challenges and results on the digital preservation of Aleijadinho's sculpture of the Prophet Joel. *Journal of Cultural Heritage*, 13(2):210–214, 2012.

[3] Freedman Barak, Shpunt Alexander, Machline Meir, e Arieli Yoel. Depth mapping using projected patterns. *US Patent : WO 2008/120217 A2*, 2008.

[4] S. Bauer, B. Berkels, J. Hornegger, e M. Rumpf. Joint ToF image denoising and registration with a CT surface in radiation therapy. *Proc. of the Conference on Scale Space and Variational Methods*, páginas 98–109, 2011.

[5] P.J. Besl e H.D. McKay. A method for registration of 3-D shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256, Fevereiro de 1992.

[6] G. Blais e M.D. Levine. Registering multiview range data to create 3d computer objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(8):820–824, Agosto de 1995.

[7] D. Blythe. Rise of the graphics processor. *Proc. of the IEEE*, 96(5):761–778, Maio de 2008.

[8] Gary Bradski e Adrian Kaehler. *Learning OpenCV: Computer Vision with the OpenCV Library*. O'Reilly, Cambridge, MA, 2008.

[9] L. Cayton. A nearest neighbor data structure for graphics hardware. *Proc. of the 1st International Workshop on Accelerating Data Management Systems Using Modern Processor and Storage Architectures*, páginas 1–6, 9 de 2010.

[10] Lawrence Cayton. Accelerating nearest neighbor search on manycore systems. *Computing Research Repository*, abs/1103.2635, 2011.

[11] Derek Chan, Hylke Buisman, Christian Theobalt, e Sebastian Thrun. A noise-aware filter for real-time depth upsampling. *Proc. of the ECCV Workshop on Multi-camera and Multi-modal Sensor Fusion Algorithms and Applications*, páginas 1–12, 2008.

[12] Y. Chen e G. Medioni. Object modeling by registration of multiple range images. *Proc. of the IEEE International Conference on Robotics and Automation*, volume 3, páginas 2724–2729, Abril de 1991.

[13] C. Chow, T. Tsui, e T. Lee. Surface registration using a dynamic genetic algorithm. *Pattern Recognition*, volume 37, páginas 105–117, 2004.

[14] Y. Cui, S. Schuon, D. Chan, S. Thrun, e C. Theobalt. 3D shape scanning with a Time-of-Flight camera. *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, páginas 1173–1180, 2010.

[15] Brian Curless e Marc Levoy. A volumetric method for building complex models from range images. *Proc. of the 23rd Annual Conference on Computer graphics and Interactive Techniques*, páginas 303–312, 1996.

[16] James Davis, Stephen R. Marschner, Matt Garr, e Marc Levoy. Filling holes in complex surfaces using volumetric diffusion. *Proc. of the 1st International Symposium on 3D Data Processing Visualization and Transmission*, páginas 428–438, Junho de 2002.

[17] J. Diebel e S. Thrun. An application of Markov Random Fields to range sensing. *Proc. of Neural Information Processing Systems*, páginas 291–298, 2005.

[18] Jerome H. Friedman, Jon Luis Bentley, e Raphael Ari Finkel. An algorithm for finding best matches in logarithmic expected time. *ACM Transactions on Mathematics Software*, 3(3):209–226, 1977.

[19] V. Garcia, E. Debreuve, F. Nielsen, e M. Barlaud. K-nearest neighbor search: Fast GPU-based implementations and application to high-dimensional feature matching. *Proc. of the 17th IEEE International Conference on Image Processing*, páginas 3757–3760, Setembro de 2010.

[20] V. Garro, P. Zanuttigh, e G.M. Cortelazzo. A new super resolution technique for range data. *Annual Conference on Associazione Gruppo Telecomunicazioni e Tecnologie dell'Informazione*, 2009.

[21] M. Gevrekci e K. Pakin. Depth map super resolution. *Proc. of the 18th IEEE International Conference on Image Processing*, páginas 3449–3452, Setembro de 2011.

[22] Leonardo Gomes, Luciano Silva, e Olga R. P. Bellon. Alinhamento de imagens de profundidade: Pré-alinhamento automático para aplicação na modelagem 3D de objetos de patrimônios culturais. *Proc. of the XXIV Conference on Graphics, Patterns and Images*, 2011.

[23] E. Hameiri e I. Shimshoni. Estimating the principal curvatures and the Darboux frame from real 3D range data. páginas 258–267, 2002.

[24] Peter Henry, Michael Krainin, Evan Herbst, Xiaofeng Ren, e Dieter Fox. RGB-D mapping: Using depth cameras for dense 3D modeling of indoor environments. *Proc. of the RGB-D: Advanced Reasoning with Depth Cameras Workshop in conjunction with RSS*, 2010.

[25] Benjamin Huhle, Philipp Jenke, e Wolfgang Strasser. On-the-fly scene acquisition with a handy multi-sensor system. *International Journal of Intelligent Systems Technologies and Applications*, 5(3/4):255–263, Novembro de 2008.

[26] Katsushi Ikeuchi, Takeshi Oishi, Jun Takamatsu, Ryusuke Sagawa, Atsushi Nakazawa, Ryo Kurazume, Ko Nishino, Mawo Kamakura, e Yasuhide Okamoto. The great Buddha project: Digitally archiving, restoring, and analyzing cultural heritage objects. *International Journal of Computer Vision*, 75(1):189–208, Outubro de 2007.

[27] Shahram Izadi, David Kim, Otmar Hilliges, David Molyneaux, Richard Newcombe, Pushmeet Kohli, Jamie Shotton, Steve Hodges, Dustin Freeman, Andrew Davison, e Andrew Fitzgibbon. Kinectfusion: real-time 3D reconstruction and interaction using a moving depth camera. *Proc. of the 24th Annual ACM Symposium on User Interface Software and Technology*, páginas 559–568, 2011.

[28] Edie Andrew Johnson e Sing Bing Kang. Registration and integration of textured 3D data. *Proc. of the International Conference on Recent Advances in 3-D Digital Imaging and Modeling*, páginas 234–241, 1997.

[29] Jurandir Santos Jr., Olga R. P. Bellon, Luciano Silva, e Alexandre Vrubel. Improving 3D reconstruction for digital art preservation. *Proc. of the 16th international Conference on Image Analysis and Processing: Part I*, páginas 374–383, 2011.

[30] Kourosh Khoshelham e Sander Oude Elberink. Accuracy and resolution of Kinect depth data for indoor mapping applications. *Sensors*, 12(2):1437–1454, 2012.

[31] Johannes Kopf, Michael F. Cohen, Dani Lischinski, e Matt Uyttendaele. Joint bilateral upsampling. *ACM Proc. of Computer Graphics and Interactive Techniques*, 26(3), 2007.

[32] Michael Krainin, Peter Henry, Xiaofeng Ren, e Dieter Fox. Manipulator and object tracking for in-hand 3D object modeling. *International Journal of Robotics Research*, 30(11):1311–1327, Setembro de 2011.

[33] Mark Levoy. The digital Michelangelo project. *Proc. of the 2nd International Conference on 3-D Digital Imaging and Modeling*, páginas 2–11, 1999.

[34] William E. Lorensen e Harvey E. Cline. Marching cubes: A high resolution 3D surface construction algorithm. *Proc. of the 14th Annual Conference on Computer Graphics and Interactive Techniques*, páginas 163–169, 1987.

[35] Oisin Mac Aodha, Neill D.F. Campbell, Arun Nair, e Gabriel J. Brostow. Patch based synthesis for single depth image super-resolution. *Proc. of European Conference on Computer Vision*, páginas 71–84, 2012.

[36] T. Masuda. Object shape modelling from multiple range images by matching signed distance fields. *Proc. of the 1st International Symposium on 3D Data Processing Visualization and Transmission*, páginas 439–448, 2002.

[37] T. Matsuo, N. Fukushima, e Y. Ishibashi. Weighted joint bilateral filter with slope depth compensation filter for depth map refinement. *Proc. of International Conference on Computer Vision Theory and Applications*, 2013.

[38] Caroline Mendes, Luciano Silva, e Olga R. P. Bellon. IMAGO visualization system: An interactive web-based 3D visualization system for cultural heritage applications. *Journal of Multimedia*, 7(2), 2012.

[39] Marius Muja e David G. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. *Proc. of the International Conference on Computer Vision Theory and Application*, páginas 331–340, 2009.

[40] Naohito Nakasato. Implementation of a parallel tree method on a GPU. *Journal of Computational Science*, páginas 132–141, Dezembro de 2012.

[41] D. Neumann, F. Lugauer, S. Bauer, J. Wasza, e J. Hornegger. Real-time RGB-D mapping and 3-D modeling on the GPU using the Random Ball Cover data structure. *Proc. of the IEEE International Conference on Computer Vision Workshops*, páginas 1161–1167, Novembro de 2011.

[42] Richard A. Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J. Davison, Pushmeet Kohli, Jamie Shotton, Steve Hodges, e Andrew

Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. *Proc. of the 10th IEEE International Symposium on Mixed and Augmented Reality*, páginas 127–136, 2011.

[43] Anton Nordmark. Kinect 3d mapping. Dissertação de Mestrado, Linkoping University, Computer Vision, The Institute of Technology, 2012.

[44] Stanley Osher e Ronald Fedkiw. *Level Set Methods and Dynamic Implicit Surfaces*. Springer Verlag, 2002.

[45] Steven Parker, Peter Shirley, Yarden Livnat, Charles Hansen, e Peter-Pike Sloan. Interactive ray tracing for isosurface rendering. *Proc. of the Conference on Visualization*, páginas 233–238, 1998.

[46] Hanspeter Pfister, Matthias Zwicker, Jeroen van Baar, e Markus Gross. Surfels: surface elements as rendering primitives. *Proc. of the 27th Annual Conference on Computer Graphics and Interactive Techniques*, páginas 335–342, 2000.

[47] Kari Pulli. Multiview registration for large data sets. *Proc. of the 2nd International Conference on 3-D digital Imaging and Modeling*, páginas 160–168, 1999.

[48] Deyuan Qiu, Stefan May, e Andreas Nüchter. GPU-accelerated nearest neighbor search for 3D registration. *Proc. of the 7th International Conference on Computer Vision Systems*, páginas 194–203, 2009.

[49] C. Richardt, C. Stoll, N. A. Dodgson, Hans-Peter Seidel, e C. Theobalt. Coherent spatiotemporal filtering, upsampling and rendering of RGBZ videos. *Proc. of Eurographics Computer Graphics Forum*, volume 31, Maio de 2012.

[50] S. Rusinkiewicz e M. Levoy. Efficient variants of the ICP algorithm. *Proc. of the 3rd International Conference on 3-D Digital Imaging and Modeling*, páginas 145–152, 2001.

[51] Szymon Rusinkiewicz, Olaf Hall-Holt, e Marc Levoy. Real-time 3D model acquisition. *ACM Transactions on Graphics*, 21:438–446, Julho de 2002.

[52] R.B. Rusu e S. Cousins. 3D is here: Point Cloud Library (PCL). *Proc. of the IEEE International Conference on Robotics and Automation*, páginas 1–4, Maio de 2011.

[53] Ryusuke Sagawa e Katsushi Ikeuchi. Taking consensus of signed distance field for complementing unobservable surface. *Proc. of the 4th International Conference on 3D Digital Imaging and Modeling*, páginas 410–417, 2003.

[54] Joaquim Salvi, Carles Matabosch, David Fofi, e Josep Forest. A review of recent range image registration methods with accuracy evaluation. *Image and Vision Computing*, 25(5):578–596, 2007.

[55] S. Schuon, C. Theobalt, J. Davis, e S. Thrun. Lidarboost: Depth superresolution for ToF 3D shape scanning. *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, páginas 343–350, Junho de 2009.

[56] A. Segal, D. Haehnel, e S. Thrun. Generalized-ICP. *Proc. of Robotics: Science and Systems*, Junho de 2009.

[57] Mauricio P. Segundo, Leonardo Gomes, Olga R.P. Bellon, e Luciano Silva. Automating 3D reconstruction pipeline by SURF-based alignment. *Proc. of the IEEE International Conference on Image Processing*, 2012.

[58] Mauricio P. Segundo, Luciano Silva, e Olga R. P. Bellon. Towards image acquisition and preprocessing for real-time 3D face recognition. *Proc. of the XXIV Conference on Graphics, Patterns and Images*, 2011.

[59] Luciano Silva, Olga R.P. Bellon, e K.L. Boyer. Enhanced, robust genetic algorithms for multiview range image registration. *Proc. of the 4th International Conference on 3-D Digital Imaging and Modeling*, páginas 268–275, outubro de 2003.

[60] N. Snavely, C.L. Zitnick, S.B. Kang, e M. F. Cohen. Stylizing 2.5-D video. *Proc. of Non-Photorealistic Animation and Rendering*, 2006.

[61] Evgeny Spektor, Zafrir Mor, e Dmitri Rais. Integrated processor for 3D mapping. *US Patent : US 2010/0007717 A1*, 2010.

[62] C. Tomasi e R. Manduchi. Bilateral filtering for gray and color images. *Proc. of the 6th International Conference on Computer Vision*, páginas 839–846, Janeiro de 1998.

[63] Alexandre Vrubel, Olga R. P. Bellon, e Luciano Silva. A 3D reconstruction pipeline for digital preservation. *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, páginas 2687–2694, Junho de 2009.

[64] T. Weise, T. Wismer, B. Leibe, e L. Van Gool. In-hand scanning with online loop closure. *Proc. of the 12th International IEEE Conference on Computer Vision Workshops*, páginas 1630 –1637, Outubro de 2009.

[65] Mark D. Wheeler, Yoichi Sato, e Katsushi Ikeuchi. Consensus surfaces for modeling 3D objects from multiple range images. *Proc. of the 6th International Conference on Computer Vision*, páginas 917–924, 1998.

[66] T. Whelan, M. Kaess, M.F. Fallon, H. Johannsson, J.J. Leonard, e J.B. McDonald. Kintinuous: Spatially extended KinectFusion. *Proc. of the RSS Workshop on RGB-D: Advanced Reasoning with Depth Cameras*, 2012.

[67] Q. Yang, R. Yang, J. Davis, e D. Nister. Spatial-depth super resolution for range images. *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, páginas 1–8, 2007.

[68] Kun Zhou, Qiming Hou, Rui Wang, e Baining Guo. Real-time KD-tree construction on graphics hardware. *ACM Transactions on Graphics*, 27(5):126:1–126:11, Dezembro de 2008.

[69] Xiuzhuang Zhou, Yao Lu, Jiwen Lu, e Jie Zhou. Abrupt motion tracking via intensively adaptive Markov-Chain Monte Carlo sampling. *IEEE Transactions on Image Processing*, 21(2):789–801, 2012.