UNIVERSIDADE FEDERAL FLUMINENSE

EDUARDO VERA SOUSA

D-KHT: DETECÇÃO EM TEMPO REAL DE PLANOS EM IMAGENS DE PROFUNDIDADE

NITERÓI

2016

UNIVERSIDADE FEDERAL FLUMINENSE

EDUARDO VERA SOUSA

D-KHT: DETECÇÃO EM TEMPO REAL DE PLANOS EM IMAGENS DE PROFUNDIDADE

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense como requisito parcial para a obtenção do Grau de Mestre em Computação. Área de concentração: Computação Visual

Orientador:
Prof. D.Sc. Leandro Augusto Frata Fernandes

Co-orientador:
Prof. Ph.D. Luiz Velho

NITERÓI

2016

EDUARDO VERA SOUSA

D-KHT: DETECÇÃO EM TEMPO REAL DE PLANOS EM IMAGENS DE PROFUNDIDADE

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense como requisito parcial para a obtenção do Grau de Mestre em Computação. Área de concentração: Computação Visual.

Aprovada em Julho de 2016.

BANCA EXAMINADORA

Prof. Leandro Augusto Frata Fernandes - Orientador, UFF

Prof. Luiz Carlos Pacheco Rodrigues Velho, IMPA

Prof. Manuel Menezes de Oliveira Neto, UFRGS

Prof. Cristina Nader Vasconcelos, UFF

Niterói

2016

Para os Vera(s) Sousa, base de tudo o que sou.

Agradecimentos

Acredito que devo começar agradecendo à minha família, sempre comigo nos momentos bons e ruins. Agradeço a Cleonice e Raimundo por me ensinarem o valor do trabalho, dedicação e família. Vinícius, Mayana e Elis: obrigado pelo companheirismo, piadas ruins e bons momentos em videochamadas aos finais de semana. Otto Vinícius, obrigado por chegar e me mostrar que tenho algum jeito com crianças. Seu tio/padrinho te ama muito. Agradeço a Savya Mendonça por estar ao meu lado sempre carinhosa, atenciosa e (quase sempre) paciente: seu apoio foi muito importante.

Gostaria de agradecer também ao orientador, Leandro Augusto Frata Fernandes, pelos ensinamentos, apoio e pelo voto de confiança que me foi dado. Isso foi muito importante para mim e me motivou mais ainda. Agradeço também à Luiz Velho, por compartilhar um pouco de sua experiência e conhecimento no desenvolvimento desse trabalho; e a Djalma Lucio pelo trabalho em equipe ao longo do projeto, que já começa a dar resultados.

Meu obrigado a Ney Paranaguá, Adalton Sena e aos amigos da Infoway e Uniplam por sua contribuição com meu crescimento, tanto no âmbito pessoal quanto profissional.

Obrigado também à Arthur Fortes, Edmilson Frank, Lucas Aquiles, Flávia Araújo, Ariela Raissa, Márcia Soares, Mayara Thaís, Wislanildo Júnior e Leila Dulce por sua amizade sempre presente. Agradeço também a Marlon Tozzi, Jhonatan Victorino, Vinícius Aleixo, Amanda Bueno, Vitória Ohana, Jamille Passalini, Altobelli Mantuan, Heder Dorneles, Rafael Lins, Danilo Galvão, Lucas Monçôres e André Fraga: cada um de vocês contribuiu de algum modo para meu crescimento e espero poder fazer o mesmo por vocês algum dia.

Agradeço a Manuel M. Oliveira, Frederico Limberger e Vinícius Breda por sua ajuda com a implementação da 3-D KHT. Vocês possibilitaram a validação desse trabalho.

Resumo

Neste trabalho é descrita uma abordagem para detecção de planos, em tempo real, em imagens de profundidade. Explorando o modelo de captura e a estrutura regular de imagens de profundidade, imagens integrais e uma *quadtree* são empregadas no cálculo eficiente de regiões aproximadamente coplanares da cena. A incerteza do plano que melhor se ajusta a cada uma dessas regiões é usada como entrada para uma Transformada de Hough com votação baseada em *kernels*. Após a votação, um método computacionalmente eficiente de detecção de máximos locais, baseado em subida de gradiente, é aplicado na identificação dos planos que melhor se ajustam às amostras na imagem de entrada.

Dentre as contribuições deste trabalho está o desenvolvimento de uma técnica de complexidade computacional linear voltada à detecção de planos em imagens de profundidade, ainda que essas possuam descontinuidades causadas por oclusão ou ruídos. Além disso, a técnica se mostra capaz de ignorar completamente superfícies não-planares.

Para a validação da técnica descrita, nesse trabalho foram utilizadas como entrada tanto imagens de cenas sintéticas contendo pontos perfeitamente alinhados sobre os planos, quanto imagens de cenas reais com ruídos e distorções introduzidas pelo dispositivo de captura. Além disso, para cada um dos casos, foi considerada a presença de objetos não planares, que poderiam levar à detecção de planos espúrios. A qualidade da detecção e os tempos de execução foram comparados com a técnica estado-da-arte.

A abordagem apresentada para detecção de planos possui diversas aplicações. Dentre elas, está sendo desenvolvida uma técnica para a aproximação da geometria de ambientes internos por recortes de planos. Essa técnica recebe como entrada imagens de profundidade capturadas por dispositivos móveis, em tempo real.

Palavras-chave: Transformada de Hough, detecção de planos, tempo real, reconhecimento de padrões.

Lista de Figuras

Lista de Tabelas

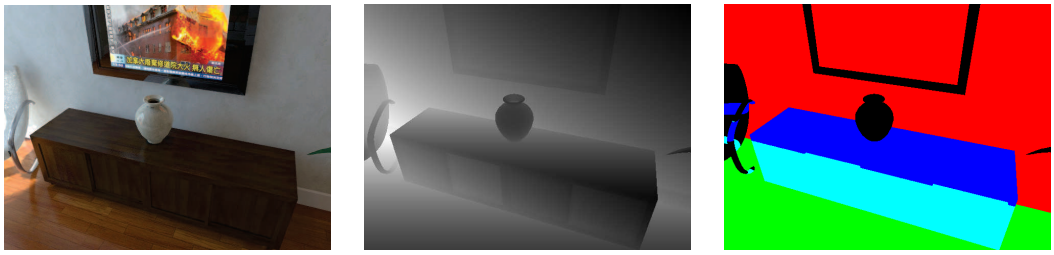
Sumário

Capítulo 1

Introdução

A crescente popularidade de sensores de profundidade embutidos em dispositivos móveis, como o Structure Sensor, da Occipital [1], e o Projeto Tango da Google [2], faz a tecnologia de escaneamento 3-D mais barata e onipresente. Esse tipo de sensor, de forma similar a câmeras comuns, mapeia as superfícies visíveis pela câmera, mas gerando uma imagem que transmite a noção de profundidade para cada pixel. Esse tipo de reconstrução, é chamada 2,5-D. A facilidade de acesso a dispositivos desse tipo também acaba por motivar o desenvolvimento de variantes em tempo real de técnicas clássicas de processamento de imagens e visão computacional. Como exemplo, pode ser citada a necessidade de detecção, em tempo real, de estruturas planares a partir de imagens de profundidade, visto como um passo importante para a solução de vários problemas práticos, cujas aplicações incluem: reconstrução baseada em imagens [3, 4], calibração de câmeras [5], navegação autônoma de veículos [6] e realidade aumentada [7]. Entretanto, o estado-da-arte para detecção de planos em nuvens de pontos desorganizados não é adequado para lidar com dados de profundidade, considerando que muitas vezes necessitam de estruturas de dados especiais para organizar os pontos no espaço 3-D [8], ou explorar estratégias não determinísticas para reduzir o processamento necessário para a execução dessas técnicas [9]. Por outro lado, técnicas baseadas em Crescimento de Superfície (SG, do inglês *Surface Growing*) [10, 11, 12, 13, 14] para a detecção de fragmentos planares, apesar de bem adaptadas a imagens de profundidade, são bastante dependentes de bons palpites iniciais para o ajuste dos planos (ver Capítulo 2) e de níveis de ruído nos dados. Além disso, oclusões ou regiões da imagem sem dados de profundidade podem levar a múltiplas detecções de um mesmo plano. Tais técnicas, assim como suas características e limitações, serão melhor exploradas no Capítulo 2.

Nesse trabalho, é apresentada uma técnica eficiente para a detecção de planos em



(a) Imagem colorida (b) Imagem de profundidade (c) Planos detectados

Figura 1.1: Resultado produzido pela abordagem apresentada: (a) Imagem colorida da cena; (b) Imagem de profundidade utilizada como entrada para o algoritmo, com 640×480 pixels e 16 bits de profundidade; e (c) Planos detectados, identificados por cores. Pixels pretos em (c) indicam superfícies não-planares. Perceba que o algoritmo se comporta bem, mesmo na presença de superfícies não-planares como o vaso, a cadeira e a planta.

imagens de profundidade. A abordagem proposta explora a natureza 2,5-D e a estrutura regular de imagens de profundidade, além de tirar proveito de seu formato retangular para evitar recálculos. A solução determinística proposta permite a implementação do algoritmo visando sua detecção em tempo real em um computador pessoal (PC), com baixo consumo de memória. Além disso, a abordagem proposta é robusta à ruído tipicamente encontrado nesse tipo de imagem e à detecções múltiplas do mesmo plano, mesmo com oclusões e áreas sem informação de profundidade. A Figura 1.1c mostra o resultado obtido pelo algoritmo proposto aplicado à imagem na Figura 1.1b. Cada cor na Figura 1.1c representa um plano distinto detectado por nosso algoritmo em 15,34 ms em um PC com processador de 3,4 GHz. Regiões na cor preta indicam superfícies identificadas pela técnica como não-planares. A imagem de entrada (Figura 1.1b) possui 640×480 pixels com profundidade de 16 bits. A imagem corresponde ao Frame 294 da base de imagens *Living Room* [15].

Dentre as técnicas clássicas para detecção de entidades geométricas analíticas (*e.g.* planos, retas, círculos, elipses, etc), a Transformada de Hough [16] (HT, do inglês *Hough Transform*) é uma que se mostra bastante eficaz. Em sua versão original, consiste em avaliações de todos os pontos de entrada residentes em um espaço n-dimensional e na identificação de todas as instâncias que descrevem as entidades geométricas pretendidas e que poderiam conter o ponto de entrada avaliado. Avaliadas as instâncias, é realizada uma votação no chamado espaço de parâmetros da representação analítica das entidades geométricas pretendidas e, ao final dessa votação, os máximos locais, quando mapeados de volta para o espaço n-dimensional inicial, correspondem às instâncias do modelo pré-determinado que melhor se ajustam aos dados de entrada. Porém, em sua forma clássica,

a HT, além de possuir alto consumo de memória, realiza um procedimento de votação com base em força bruta, analisando cada uma das entradas de maneira individual, o que torna a técnica computacionalmente custosa. Contudo, é possível clusterizar entradas com características semelhantes e realizar a votação para cada cluster, conforme demonstrado na Transformada de Hough Baseada em Kernels (KHT, do inglês *Kernel-Based Hough Transform*) por Fernandes e Oliveira [17], para o caso de detecção de retas em imagens binárias, e estendida por Limberger e Oliveira [8], para o caso de detecção de planos em nuvens de pontos (3-D KHT). Entretanto, a 3-D KHT apresenta complexidade assintótica $O(n \log n)$, pois faz uso de uma *octree* como estrutura auxiliar na clusterização de pontos de entrada no espaço 3-D. O que leva a técnica a não executar em tempo real quando aplicada sobre imagens de profundidade de cenas complexas. Além disso, a 3-D KHT utiliza ordenação de um vetor para a detecção de máximos locais, com complexidade assintótica $n \log_2 n$, onde n corresponde à quantidade de células que receberam votos no espaço de parâmetros discretizado, enquanto a técnica descrita, fazendo uso de subida no gradiente, realiza a detecção de picos em tempo $O(n)$ para uma quantidade de picos muito menor que n, sendo possível assumir complexidade constante, conforme explorado no Capítulo 4.

1.1 Ideia Central

A ideia central desse trabalho pode ser descrita como:

> É possível detectar planos em imagens de profundidade em tempo linear ao considerar as restrições impostas pelo modelo de captura e pela estrutura regular de imagens de profundidade, bem como a coerência entre fragmentos planares associados a cada grupo de pixels vizinhos aproximadamente coplanares e o plano a ser detectado.

De modo geral, em uma HT convencional, a votação e a detecção de picos, passos que serão melhor abordados ao longo do Capítulo 2 e Capítulo 4, são os passos que possuem maior impacto no consumo de memória e tempo de processamento, respectivamente. Considerando as ideias propostas por Limberger e Oliveira [8], o tempo de processamento e o consumo de memória são reduzidos consideravelmente, mas o tempo de clusterização de pontos aproximadamente coplanares deve passar a ser considerado. Assim, dentre os desafios deste trabalho, é possível explicitar:

1. Aproveitar a estrutura regular e o modelo de captura de imagens de profundidade para reduzir o tempo de processamento necessário à clusterização;

2. Utilizar a observação de que os máximos locais no espaço de parâmetros estão tipicamente próximos dos parâmetros do plano médio de cada cluster para reduzir o tempo de processamento necessário à detecção de máximos locais, isto é, existe uma coerência entre os fragmentos planares presentes em cada cluster e o plano a ser detectado; e

3. Apresentar uma técnica para detecção de planos com qualidade e tempo de execução que supere o estado-da-arte, quando aplicado a imagens de profundidade.

Conforme mencionado anteriormente, na solução desses problemas foram empregadas, dentre outros recursos, restrições impostas pelo tipo de entrada, que serão descritas ao longo dos Capítulos 3 e 4.

1.2 Resultados e Contribuição

A principal contribuição desse trabalho é uma abordagem para detecção de planos em imagens de profundidade: a D-KHT, do inglês *Depth Kernel-Based Hough Transform*, que possui custo computacional assintótico da ordem de $O(n)$ e, conforme demonstrado no Capítulo 5, é de ~ 3 a ~ 10 vezes mais rápido que o estado-da-arte. A técnica apresentada é um dos produtos resultantes do desenvolvimento de uma abordagem para a aproximação da geometria de ambientes internos por planos, em tempo real e com a utilização de sensores de profundidade em dispositivos móveis. Alguns do detalhes dessa abordagem estão descritos no Capítulo 6.

Para a validação dos resultados, a D-KHT foi comparada com a 3-D KHT, considerada estado-da-arte para a detecção de planos em nuvens de pontos. A técnica apresentada não foi comparada com SG, dada sua característica de múltiplas detecções em caso de descontinuidade na profundidade. Tal problema é abordado no Capítulo 2. Os quesitos avaliados foram o tempo de execução e a qualidade da detecção. Nesse sentido, foram considerados frames provenientes de bases de imagens reais e sintéticas, com e sem superfícies não-planares, principais causas da detecção de planos espúrios. Os experimentos conduzidos e os respectivos resultados estão descritos no Capítulo 5.

Capítulo 2

Trabalhos Relacionados

Nesse capítulo são enumeradas algumas técnicas utilizadas na detecção de entidades geométricas analíticas, bem como suas características e limitações. É dada atenção especial a técnicas baseadas em Transformada de Hough (HT) e espaços de baixa dimensionalidade.

2.1 Técnicas Baseadas em Transformada de Hough

A HT [16] é uma das técnicas mais populares para a detecção de entidades geométricas analíticas em espaços de baixa dimensionalidade. Ela foi proposta originalmente para a detecção de retas em imagens binárias, mas, dada a simplicidade da técnica, é facilmente estendida para a detecção de outras estruturas. Duda e Hart [18] propuseram a utilização da equação normal da reta como modelo de parametrização para a *Standard Hough Transform* (SHT), dado por:

$$\rho = x\,\cos\theta + y\,\sin\theta, \tag{2.1}$$

onde $(x, y)^T$ corresponde às coordenadas de um ponto sobre a reta $(\rho, \theta)^T$, $\rho \in [-R, +R]$ é a distância da reta ao centro da imagem (origem), e $\theta \in [0, \pi)$ é o ângulo entre o eixo x e a direção normal da reta. $R = \sqrt{w^2 + h^2}/2$, onde w e h correspondem à altura e à largura da imagem, respectivamente.

Dada uma imagem binarizada após a aplicação de um filtro para a detecção de bordas, para cada pixel $(x', y')^T$, a SHT identifica todos os pares $(\rho, \theta)^T$ que satisfazem (2.1), *i.e.,* todas as linhas que passam por cada pixel, arbitrando valores discretos para θ e computando ρ utilizando (2.1). Na SHT, o espaço definido pelos parâmetros (ρ, θ) da reta, *i.e.,* o espaço de parâmetros, deve ser apropriadamente discretizado como um mapa de acumuladores. As células do mapa de acumuladores associados aos parâmetros das

retas que passam por cada pixel de borda são incrementadas em uma unidade para cada pixel. Os máximos locais criados no mapa de acumuladores por esse processo de votação correspondem aos parâmetros das retas detectadas. Os passos de discretização assumidos para ρ e θ influenciam diretamente o tempo de execução, consumo de memória e qualidade das detecções.

Para a detecção de planos, o modelo para retas (2.1) é naturalmente substituído pela equação normal do plano:

$$\rho = x \sin\phi \cos\theta + y \sin\phi \sin\theta + z \cos\phi, \tag{2.2}$$

onde $(x, y, z)^T$ correspondem às coordenadas cartesianas de um ponto sobre o plano $(\rho, \phi, \theta)^T$, $\rho \in \mathbb{R}^+$ é a distância não-negativa do plano à origem do espaço tridimensional, e $\phi \in [0, \pi)$ e $\theta \in [0, 2\pi)$ são os coeficientes angulares do plano.

Vosselman *et al.* [19] apresentaram uma técnica baseada em HT para a detecção de planos. Os autores dividem o processo de detecção em duas etapas. Em primeiro lugar, exploram a conectividade de nuvens de pontos obtidas com *scanners* a laser para calcular o vetor normal do plano que melhor se ajusta ao conjunto de entradas. Em seguida, as orientações mais frequentes são identificadas por votações no espaço de parâmetros 2-D dos coeficientes angulares de (2.2). Por fim, os pontos atribuídos à orientação mais frequente votam para a obtenção da distância do plano à origem num espaço de parâmetros 1-D. A abordagem de Vosselmann *et al.* possui baixo consumo de memória. No entanto, é mais lenta que outras transformadas (ver Seção 2.1.1) quando aplicada a nuvens de pontos desorganizados. Além disso, a utilização de uma única direção por ponto reduz a robustez da técnica em níveis elevados de ruído. Uma proposta similar foi apresentada por Nguyen *et al.* [20]. As principais diferenças estão no modo como é feito o cálculo dos vetores normais e a identificação das prováveis orientações no primeiro espaço de parâmetros.

2.1.1 Transformada de Hough baseada em *kernels*

Fernandes e Oliveira [17] observaram que a votação por força bruta, em que todos as entradas participam do processo de votação de forma independente, como é realizada na SHT, apresenta o problema de alto custo computacional e, desse modo, introduziram a proposta de um método baseado em agrupamento de pontos aproximadamente colineares para realizar a votação na detecção de retas: a Transformada de Hough baseada em Kernels (KHT, do inglês *Kernel-Based Hough Transform*).

De modo geral, a abordagem adotada na KHT consiste na verificação dos pontos de borda conexos e aproximadamente colineares, no agrupamento desses pontos em clusters e na votação para os parâmetros da reta, descritos em (2.1), que melhor se ajustam a cada cluster, considerando que existe incerteza, já que os pontos não são exatamente colineares. Assim, a KHT modela essa incerteza como uma distribuição gaussiana bivariada, utilizando os parâmetros ρ e θ obtidos de cada segmento de reta como as variáveis aleatórias consideradas. Uma vez computada a distribuição sobre $(\rho, \theta)^T$, é feita a votação para cada cluster, evitando assim a formação de um mapa de acumuladores denso. O algoritmo realiza uma detecção bastante rápida e robusta, mesmo quando executado com entradas ruidosas e sobre imagens de alta resolução.

Limberger e Oliveira [8] estenderam a KHT para a detecção de planos que melhor se ajustam a nuvens de pontos desorganizados em um espaço 3-D. A 3-D KHT não assume uma pré-organização dos pontos de entrada. Assim, os autores propuseram o uso de uma *octree* para subdividir a nuvem de pontos de modo que cada nó-folha da árvore inclua um cluster de pontos aproximadamente coplanares ou um conjunto de *outliers*. Nesse último caso, o cluster é desconsiderado no processo de votação. A 3-D KHT assume um espaço de parâmetros esférico e, de maneira similar à KHT, o mapa de acumuladores é atualizado baseando-se na contribuição de uma distribuição gaussiana que modela a incerteza do plano que melhor se ajusta a cada cluster. Experimentos mostraram que a 3-D KHT supera outras técnicas baseadas em HT para detecção de planos [8].

2.2 Outras Abordagens

Random Sample Consensus (RANSAC) [9] é um método iterativo não-determinístico para ajuste de parâmetros de um modelo. Sua principal vantagem é a robustez à presença de ruído. Infelizmente, apenas uma instância do modelo desejado é detectado por execução, e boa parte do custo computacional é decorrente do tratamento de entradas espacialmente desorganizadas. Schnabel *et al.* [21] introduziram uma otimização para RANSAC que utiliza uma *octree* para estabelecer a proximidade espacial entre as entradas, aprimorando a detecção de planos, esferas, cilindros, cones e *tori* utilizando pontos num espaço 3-D como entrada. Apesar de superar várias deficiências do modelo clássico de RANSAC, a abordagem de Schnabel *et al.* se mostrou menos eficiente que a 3-D KHT [8].

Fernandes e Oliveira [22, 23] propuseram um *framework* bastante genérico baseado em votações para detecção de estruturas em bases de dados grandes, multidimensionais,

ruidosas e desorganizadas. Diferentemente de técnicas clássicas como HT e RANSAC, que são construídas visando a detecção de um tipo específico de estrutura considerando um tipo específico de entrada, sua abordagem é independente das propriedades geométricas das estruturas a serem detectadas, assim como é independente do tipo de entrada fornecida. Dessa forma, essa técnica pode ser usada para detecção de planos com ou sem incertezas associadas à entrada. Essa técnica, no entanto, possui alto custo computacional e não é executada em tempo real.

Crescimento de Superfícies (SG, do inglês *Surface Growing*) [10, 11, 12, 13, 14] é uma técnica baseada em Crescimento de Regiões [24] que detecta fragmentos planares utilizando uma restrição de suavidade em imagens de profundidade. Consiste em selecionar pontos na imagem (*seeds*), analisar a vizinhança desses pontos e expandir a região enquanto a superfície se mantiver suave. Em outras palavras, se um ponto está dentro de um limiar de suavidade para uma superfície planar, isto é, se um ponto, dado um limiar, é considerado como pertencente a uma porção de um plano, tal porção é expandida e os vizinhos do ponto serão avaliados em seguida. Apesar de SG funcionar bem mesmo na presença de pequenas quantidades de ruído, essa técnica é fortemente relacionada à seleção de bons *seeds*. Assim, a técnica é sensível a um bom palpite inicial. Além disso, SG requer conectividade entre os pixels associados ao mesmo plano para que sejam evitadas múltiplas detecções concorrentes, o que não é sempre observado em imagens de profundidade, considerando que porções de informação podem não ser reconstruídas devido à oclusões e presença de materiais reflexivos ou que absorvem luz [25].

Xiao *et al.* [14] acrescentaram à literatura existente uma modificação de SG voltada para nuvens de pontos e imagens de profundidade ao utilizar informações locais dos *seeds*, visando evitar o crescimento baseado em força bruta. De modo geral, são ajustados *grids* sobre a imagem de profundidade e cada *grid* é classificado de acordo com sua forma. Em caso de *grids* vizinhos coplanares, aquela região é expandida. A restrição da técnica é o tempo de processamento, que possui uma relação direta com a quantidade de planos detectados na cena. Muitos planos detectados implicam em mais tempo de processamento, demonstrando que a complexidade do algoritmo é sensível à saída.

2.3 Discussão

Ao longo desse capítulo foram relacionados, em alto nível, alguns dos trabalhos mais próximos à técnica aqui proposta, bem como suas características e limitações.

A SHT é uma boa base para as técnicas baseadas em Transformada de Hough mas, conforme foi visto, sua votação baseada em força bruta acaba por impactar negativamente no custo computacional do algoritmo. Tal custo computacional foi posteriormente reduzido com a introdução da KHT para o caso de retas em 2D. Para o caso 3D, a KHT foi expandida para a 3-D KHT, que lida com a detecção de planos em nuvens de pontos desorganizadas. Por serem baseadas em Transformada de Hough e por pré-processamentos envolvendo a clusterização das entradas e a posterior análise de incerteza, essas são as técnicas que mais se assemelham à técnica proposta.

O *framework* proposto por Fernandes e Oliveira [22, 23], por sua ampla flexibilidade no que diz respeito à dimensionalidade das entidades geométricas buscadas, acaba por possuir alto custo computacional, o que a torna inviável se a aplicação requer tempo real.

RANSAC, por sua vez, apesar da robustez à presença de ruído e das melhorias sugeridas por Schnabel *et al.* [21], possui alguns problemas em relação ao tempo de processamento, posto que o ajuste pode demorar várias iterações para convergir e a saída consiste em apenas uma instância da entidade geométrica buscada por vez.

SG é uma técnica bastante sensível a oclusões e descontinuidades. Isso se deve ao fato de requerer conectividade entre as porções de uma mesma superfície, o que muitas vezes não ocorre em imagens de profundidade, levando a múltiplas detecções de um mesmo plano, além do fato de requerer um bom palpite inicial para os *seeds*.

Capítulo 3

Fundamentação Teórica

Esse capítulo introduz o modelo de registro de imagens de profundidade (Seção 3.1), utilizadas como entrada para a técnica proposta neste trabalho. As Seções 3.2 e 3.3 abordam de forma conceitual as ideias de *quadtree* e imagens integrais (SAT, do inglês *Summed Area Table*), respectivamente, esses são recursos de fundamental importância para a complexidade computacional do algoritmo proposto, conforme será visto no Capítulo 4.

3.1 Imagens de Profundidade

O modelo de câmera adotado para o registro das imagens é o da câmera *pinhole*. Dado um ponto $\mathrm{X} = (X, Y, Z)^T$ representado nos sistema de coordenadas da câmera, no espaço 3-D, sua projeção no ponto nas coordenadas da imagem é dada por:

$$\mathrm{x} = \begin{pmatrix} i \\ j \end{pmatrix} = \begin{pmatrix} \lfloor \frac{x'}{w} \rfloor \\ \lfloor \frac{y'}{w} \rfloor \end{pmatrix}, \tag{3.1}$$

para

$$\begin{pmatrix} x' \\ y' \\ w \end{pmatrix} = K \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}, \tag{3.2}$$

onde K é a matriz de parâmetros intrínsecos da câmera, dada por:

$$K = \begin{pmatrix} \alpha_x & \gamma & c_x \\ 0 & \alpha_y & c_y \\ 0 & 0 & 1 \end{pmatrix}, \tag{3.3}$$

onde α_x e α_y correspondem à distância focal em termos de pixels, c_x e c_y correspondem às coordenadas do centro da câmera e γ é referente ao *skew*, aqui igualado a zero. É possível desconsiderar a matriz de parâmetros extrínsecos da projeção se for assumido que o centro da câmera é também a origem do espaço tridimensional e que o eixo principal da câmera está alinhado ao eixo z do espaço 3-D.

A ideia de reconstrução de profundidade em uma cena consiste em mapear um pixel de uma imagem (2-D) capturada em um ponto de vista qualquer dessa cena a um ponto na cena (3-D). As técnicas para a reconstrução costumam ser classificadas como ativas ou passivas. A reconstrução passiva baseia-se em uma (*e.g. Shape from shading*) ou mais (*e.g.* triangulação passiva) imagens capturadas da cena e, a partir da análise dessas imagens, a posição do ponto em 3-D é estimada. Já na reconstrução ativa, algum tipo de sinal é adicionado à cena e, com base na deformação projetiva desse sinal, estimam-se as coordenadas de cada ponto da cena.

As imagens de profundidade geradas pelos dispositivos utilizados para a captura das entradas do algoritmo proposto foram obtidas por reconstrução ativa por luz estruturada. Ou seja, os dispositivos projetam um padrão com luz infravermelha na cena e, de acordo com a deformação projetiva desse padrão de luz, estimam a profundidade em cada ponto. Assim, em imagens de profundidade, não é armazenado no pixel a intensidade de cor presente no ponto correspondente na cena em 3-D, mas a profundidade obtida com base na triangulação ativa. Assim, para cada pixel da imagem de profundidade existe apenas um valor armazenado, isto é, não existe sobreposição de pontos ao longo dos raios que partem do centro da câmera, passam pelo plano de imagem e atingem as superfícies mais próximas na cena. Esse tipo de reconstrução da profundidade das superfícies mais próximas corresponde ao chamado 2,5-D. Assim, a definição formal de imagem de profundidade é dada por:

$$d(i, j) : \mathbb{R}^2 \rightarrow \mathbb{R}^+, \tag{3.4}$$

onde (i, j) correspondem às coordenadas de um pixel na imagem e $d(i, j)$ resulta em um único valor que representa a distância $z \geq 0$ do centro da câmera a um ponto na cena.

É importante notar a maneira como são alocados os valores de profundidade para cada pixel: em uma imagem RGB, por exemplo, costuma-se alocar 8 bits para cada canal de cor, o que permite 256 variações de intensidade por canal. Para imagens de profundidade, no entanto, não é interessante essa baixa resolução em faixas de profundidade. Assim, os dispositivos utilizados nesse trabalho, como o Structure Sensor [1] e o Projeto Tango [2], reservam 16 bits para armazenamento de profundidade, o que gera 65.536 diferentes faixas

de profundidade.

3.2 *Quadtree*

Quadtrees [26] são estruturas utilizadas para representar dados espaciais, clusterizando-os segundo suas propriedades. Sua principal característica é a divisão recursiva do espaço, de forma alinhada aos eixos, segundo um critério pré-determinado.

Dada uma região $\mathcal{Q}$ a ser subdividida com uma *quadtree*, o primeiro nível é composto pela região toda. Caso o critério de clusterização não seja satisfeito, a região é subdividida em quadrantes de mesmo tamanho. Em seguida, o critério é avaliado novamente para cada quadrante e a subdivisão é feita de forma recursiva até que o critério seja satisfeito ou que seja atingido um ponto de parada que diz respeito à resolução dessa *quadtree*, *i.e.,* o tamanho mínimo de cada nó da *quadtree*. A Figura 3.1 exemplifica a subdivisão. Para essa figura, extraída de [27], foi adotado como critério de subdivisão a presença de no mínimo um pixel de borda do círculo.

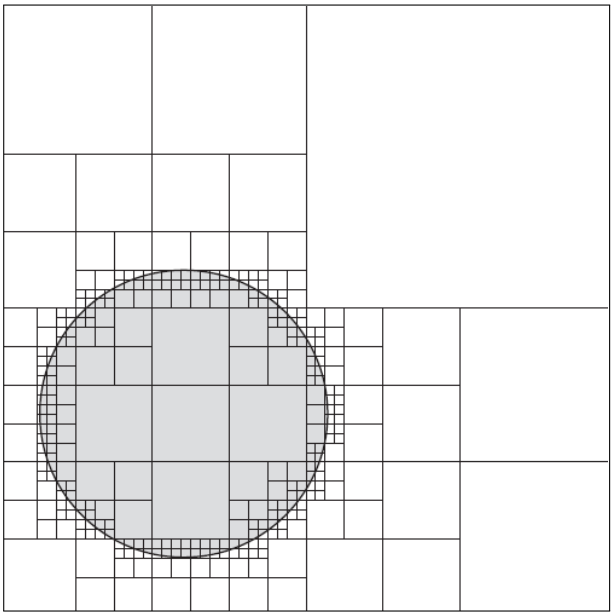


Figura 3.1: Exemplo de *quadtree* utilizando, como critério de subdivisão, a presença de pixels de borda do círculo.

3.3 Imagens Integrais - SATs

Imagens integrais (SATs, do inglês *Summed Area Tables*) [28] são recursos criados originalmente para o cálculo de coordenadas de texturas e que permitem calcular rapidamente

a soma de valores contidos em uma região retangular de uma matriz. De modo geral, uma SAT S_Σ calculada sobre uma matriz M de tamanho $m \times n$ pode ser definida como:

$$S_\Sigma(i,j) = M(i,j) + S_\Sigma(i-1,j) + S_\Sigma(i,j-1) - S_\Sigma(i-1,j-1), \tag{3.5}$$

sendo $0 \leq i < n$ e $0 \leq j < m$ os índices de um elemento da matriz, levando em conta que $S_\Sigma(a,b) = 0$ se $a < 0$ ou $b < 0$.

Para melhor exemplificar, considere a matriz

$$M = \begin{array}{|c|c|c|c|} \hline 4 & 9 & 8 & 2 \\ \hline 4 & 5 & 3 & 6 \\ \hline 1 & 0 & 0 & 7 \\ \hline 7 & 2 & 6 & 9 \\ \hline \end{array}.$$

Sua SAT é dada por:

$$S_\Sigma = \begin{array}{|c|c|c|c|} \hline 4 & 13 & 21 & 23 \\ \hline 8 & 22 & 33 & 41 \\ \hline 9 & 23 & 34 & 49 \\ \hline 16 & 32 & 49 & 73 \\ \hline \end{array},$$

construída com base em (3.5), em tempo linear [28]. Por outro lado, sua utilização possui custo computacional constante ($O(1)$), sendo necessárias apenas quatro consultas à S_Σ, duas subtrações e uma adição para avaliar o somatório de qualquer região retangular de M, pois:

$$\sum_{a=i}^{i+\Delta i} \sum_{b=j}^{j+\Delta j} M(a,b) = S_\Sigma(i+\Delta i, j+\Delta j) - S_\Sigma(i, j+\Delta j) - S_\Sigma(i+\Delta i, j) + S_\Sigma(i,j), \tag{3.6}$$

onde $i + \Delta i$ e $j + \Delta j$ correspondem às coordenadas dos limites superiores da área retangular e i e j correspondem às coordenadas dos limites inferiores da área retangular.

3.4 Discussão

Foram apresentados nesse Capítulo alguns dos elementos-chave para a baixa complexidade computacional do algoritmo descrito no Capítulo 4. Foi abordada a geometria da captura, bem como o tipo de reconstrução da noção de profundidade da cena e as implicações

desse tipo de reconstrução. De modo geral, com base na definição formal de imagem de profundidade, apontada em (3.4), são extraídas algumas das restrições que reduzem o custo computacional do algoritmo proposto.

A Seção 3.2 abordou o conceito de *quadtrees*, necessário ao entendimento geral da técnica e, aliado ao que foi demonstrado na Seção 3.3, que tratou da construção e utilização de SATs, constituem a base da abordagem apresentada. A maneira como tais conceitos são aplicados ao contexto de detecção de planos é explicada no Capítulo 4.

Capítulo 4

Depth Kernel-Based Hough Transform - D-KHT

Ao longo desse Capítulo, será descrita a técnica proposta, a D-KHT, detalhando cada parte de seu processo, apresentado em alto nível na Figura 4.1. Em linhas gerais, a técnica proposta: (i) subdivide a imagem de profundidade como uma *quadtree* onde os nós folhas contém amostras aproximadamente coplanares da cena ou amostras de superfícies não planares. Utilizando SATs [28], executa Análise de Componentes Principais (PCA, do inglês *Principal Component Analysis*) [29] em tempo constante para identificar o plano que melhor se ajusta aos pontos contidos em cada nó folha planar da *quadtree*, isto é, a instância da entidade geométrica buscada que melhor explica as entradas presentes em cada cluster. Em seguida, (ii) utilizando propagação de incerteza em primeira ordem [30], são computados os parâmetros do plano médio em coordenadas esféricas 2.2 e a distribuição gaussiana multivariada sob a qual são incrementados os votos no mapa de acumuladores. Por fim, (iii) utilizando os parâmetros do plano médio de cada cluster como início para uma subida no gradiente, são detectados os máximos locais que correspondem aos parâmetros do modelo buscado. Ao longo deste Capítulo, cada parte do processo será melhor detalhado.

4.1 Clusterização

Conforme visto na Seção 3.1, o modelo e a estrutura da captura de imagens de profundidade possibilitam restrições que favorecem a redução da complexidade na detecção de planos em imagens de profundidade, quando comparada à detecção de planos em nuvens de pontos quaisquer. São elas:

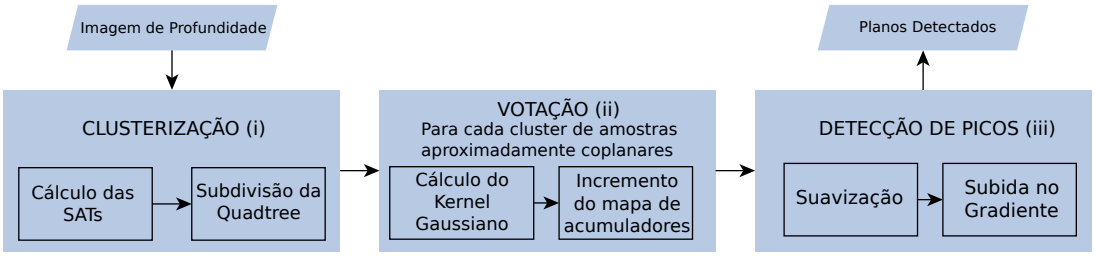


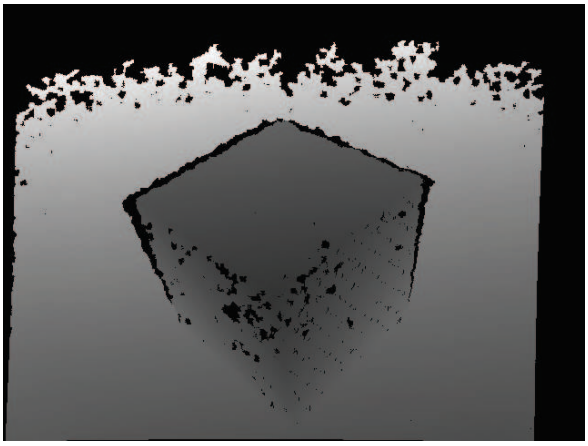
Figura 4.1: Fluxograma da D-KHT: (i) Dada uma imagem de profundidade como entrada, são computadas SATs para garantir a eficiência do processo de clusterização. Em seguida, utilizamos uma *quadtree* implícita para identificar regiões retangulares da imagem com pontos aproximadamente coplanares no espaço 3-D; (ii) Para cada região identificada no passo (i), é calculada uma distribuição gaussiana que descreve a incerteza do plano que melhor se ajusta ao conjunto de pontos pertencentes à região e utiliza-se essa gaussiana para incrementar o mapa de acumuladores esférico; (iii) Por fim, é utilizada uma técnica de subida no gradiente para detectar máximos locais no mapa de acumuladores. As coordenadas dos picos no acumulador correspondem aos planos mais prováveis na imagem de profundidade inicial.

- Para todo ponto na imagem de profundidade, sempre será registrado um valor não-negativo para o eixo z;
- Para cada ponto na imagem de profundidade existe apenas um valor armazenado, isto é, não existe sobreposição de pontos ao longo do eixo z; e
- Uma imagem de profundidade é uma matriz (discreta) retangular de elementos, permitindo assim a aplicação de SATs.

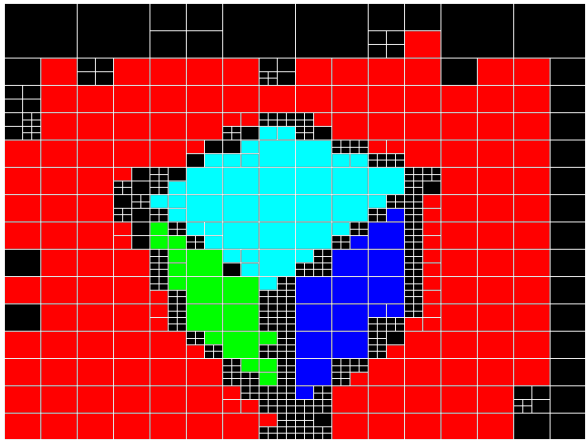
Utilizando-se dessas restrições, é razoável sugerir que, diferentemente do modelo de subdivisão adotado por Limberger e Oliveira [8], não se faz necessária uma divisão da entrada sobre o eixo z. Desse modo, a *octree* utilizada naquela abordagem sobre a nuvem de pontos de entrada pode ser substituída por uma *quadtree* implícita sobre os eixos x e y da imagem de profundidade utilizada como entrada, conforme pode ser observado na Figura 4.2. É importante notar que a *quadtree* não precisa, de fato, manipular pontos em cada quadrante. Dada a estrutura retangular da imagem, são conhecidos os limites superiores e inferiores de cada nó. Por essa razão, o modelo de *quadtree* aqui utilizado é considerado implícito.

O processo de geração da *quadtree*, descrito no Algoritmo 1, é feito de maneira recursiva, iniciando-se na imagem toda. O critério para a subdivisão de cada nó é dado com base na análise de covariância entre os pontos pertencentes àquele nó da *quadtree*.

Para cada nó, é aplicado PCA, decompondo a matriz de covariância associada aos pontos contidos no nó em autovalores e autovetores e, caso o autovalor menos significativo



(a) Imagem de profundidade de entrada



(b) *Quadtree* e planos detectados

Figura 4.2: O frame 1 da base de imagens *Cube* consiste num ponto de vista de um cubo sobre uma superfície plana (a). A imagem foi obtida utilizando um Structure Sensor. Pixels na cor preta em (a) indicam áreas sem informação de profundidade. Assim, tais pixels não correspondem a pontos em 3-D e não são utilizados como amostras pela técnica. A imagem (b) mostra os nós-folha da *quadtree* construída sobre a imagem de profundidade. Nós com a mesma cor indicam clusters de pontos que votaram para o mesmo plano detectados. Em (b), nós pretos correspondem a nós que não participam do processo de votação.

(que reflete a espessura da distribuição de pontos próximos plano que melhor se ajusta aos pontos do nó) seja inferior a um limiar pré-determinado, o nó analisado é marcado como coplanar para posterior votação no mapa de acumuladores. Caso contrário, o nó em questão é subdividido até que se obtenham nós coplanares ou que a quantidade mínima de pontos por nó tenha sido atingida. Nesse caso, o nó não participará do processo de votação e será descartado. Ao utilizar PCA, é importante notar, no entanto, que a profundidade em cada ponto da imagem costuma ser dada na unidade de medida padrão do dispositivo utilizado na captura, enquanto o par de índices da matriz de pixels que constituem a imagem não reflete a disposição exatas dos pontos no espaço. Desse modo, é necessário estabelecer a relação entre cada pixel da imagem e o ponto correspondente em 3-D:

$$\begin{pmatrix} x_{ij} \\ y_{ij} \\ z_{ij} \end{pmatrix} = \begin{pmatrix} \frac{i-c_x}{\alpha_x} z_{ij} \\ \frac{j-c_y}{\alpha_y} z_{ij} \\ z_{ij} \end{pmatrix}, \tag{4.1}$$

onde $(c_x, c_y)^T$ correspondem às coordenadas do ponto principal da câmera e α_x e α_y correspondem à distância focal em termos de pixels. Esses valores são obtidos da matriz de parâmetros intrínsecos da câmera, enquanto z_{ij}, que corresponde à profundidade no ponto (i, j) da imagem, pode ser obtido da própria entrada. Em (4.1), é assumida uma câmera

Algoritmo 1: Subdivisão recursiva da *quadtree* para a clusterização dos pontos de entrada.

```
1  Procedimento Clustering(node)
       Entrada: node representa os limites da região da imagem para o nó atual.
       Dados: s_ms é o mínimo de amostras necessárias a um cluster; e s_t é espessura
              máxima de um plano
       Resultado: Atualizações nas listas de clusters
2      se node inclui pelo menos s_ms amostras então
3          Σ_(x,y,z) ← a matriz de covariância de amostras em node;
4          V_(x,y,z), Λ_(x,y,z) ← os autovetores e autovalores de Σ_(x,y,z), onde λ_1 é o menor
            autovalor;
5          se 2√λ_1 < s_t então
6              node é nó-folha. Insira Σ_(x,y,z), V_(x,y,z), e os limites de node na lista de
                clusters;
7          senão
8              para cada quadrante de node faça
9                  Crie um nó child com os limites do quadrante;
10                 Chame Clustering(child);
11             fim
12         fim
13     senão
           /* node é um nó-folha e suas amostras não são aproximadamente
              coplanares.                                                    */
14     fim
15 fim
```

pinhole com *skew* igual a zero. Nos experimentos realizados, os parâmetros intrínsecos da câmera foram obtidos das bases de imagens utilizadas ou da calibração do dispositivo. Uma vez que esses valores sejam obtidos, é possível prosseguir com o cálculo da covariância para cada nó da *quadtree*.

A variância e covariância são medidas de probabilidade que dizem respeito à dispersão e correlação entre conjuntos de variáveis aleatórias, na unidade de medida das variáveis. Uma matriz de covariância é uma matriz simétrica que sumariza a variância e a covariância entre variáveis aleatórias. Utilizando as coordenadas das amostras no espaço retangular $\mathcal{R}$ correspondente ao nó da *quadtree*, o ponto médio e a matriz de covariância das amostras são calculados como:

$$\mu_{(x,y,z)} = \begin{pmatrix} \mu_x \\ \mu_y \\ \mu_z \end{pmatrix}, \Sigma_{(x,y,z)} = \begin{pmatrix} \sigma_{xx} & \sigma_{xy} & \sigma_{xz} \\ \sigma_{xy} & \sigma_{yy} & \sigma_{yz} \\ \sigma_{xz} & \sigma_{yz} & \sigma_{zz} \end{pmatrix}, \tag{4.2}$$

onde

$$\sigma_{ab} = \frac{1}{m-1}\left(\sum_{k=1}^{m} a_k\, b_k - \mu_b \sum_{k=1}^{m} a_k - \mu_a \sum_{k=1}^{m} b_k + m\,\mu_a\,\mu_b\right), \tag{4.3}$$

e

$$\mu_a = \frac{1}{m}\sum_{k=1}^{m} a_k, \quad \mu_b = \frac{1}{m}\sum_{k=1}^{m} b_k. \tag{4.4}$$

Em (4.3) e (4.4), m é o número de amostras em $\mathcal{R}$, e $\{a_k\}_{k=1}^{m}$ e $\{b_k\}_{k=1}^{m}$ correspondem às coordenadas x, y, ou z das amostras (4.1). Para manter a simplicidade das expressões, foi substituída a notação com dois índices $\square_{ij}$ de (4.1) por uma notação com apenas um índice $\square_k$ em (4.3) e (4.4). Perceba que m pode ser menor ou igual ao número de pixels em $\mathcal{R}$.

O cálculo de cada $\Sigma_{(x,y,z)}$ usando as expressões apresentadas em (4.2) custam $O(m)$. O cálculo de todas as matrizes de covariância custam $O(n \log_4 n)$ no pior caso, sendo n o número de pixels da imagem. No entanto, utilizando SATs, apresentadas no Capítulo 3, é possível pré-calcular as SATs x_{ij}, y_{ij}, z_{ij}, x_{ij}^2, y_{ij}^2, z_{ij}^2, $x_{ij}\,y_{ij}$, $x_{ij}\,z_{ij}$, e $y_{ij}\,z_{ij}$ reduzindo o custo computacional do cálculo da matriz de covariância para $O(1)$, desse modo, os somatórios descritos em (4.3) e (4.4)) são avaliados em tempo constante utilizando apenas quatro consultas à SATs e três operações aritméticas [28]. Um algoritmo linear para a construção das SATs é apresentado em [28]. Assim, o custo para computar as nove SATs necessárias é de $O(n)$, levando a um custo total de $O(n)$ para computar todas as matrizes $\Sigma_{(x,y,z)}$ no pior caso.

Os autovetores $V_{(x,y,z)} = \{\overrightarrow{v}_1, \overrightarrow{v}_2, \overrightarrow{v}_3\}$ e autovalores $\Lambda_{(x,y,z)} = \{\lambda_1, \lambda_2, \lambda_3\}$ computados no Algoritmo 1 (linha 4), com $0 \leq \lambda_1 \leq \lambda_2 \leq \lambda_3$, descrevem a distribuição das amostras no espaço 3-D. Considerando a raiz quadrada do menor autovalor ($\sqrt{\lambda_1}$), é obtido um desvio padrão da distribuição das amostras ao longo da componente principal menos significativa ($\overrightarrow{v}_1$). No Algoritmo 1 (linha 5) é avaliado se dois desvios padrões desse componente incluem o limiar s_t. Ao assumirmos os pontos com distribuição normal nessa componente, isso gera uma probabilidade de 95,4% de que o plano médio esteja ajustado sobre esse conjunto de pontos, fazendo com que s_t represente um envelope de pontos em torno do plano médio ajustado.

Limberger e Oliveira [8] utilizaram uma *octree* para gerar *clusters* de pontos. A subdivisão da *octree* é guiada por dois parâmetros: s_α e s_β, que correspondem às tolerâncias relativas associadas a, respectivamente, espessura e isotropia da distribuição das amostras em um dado cluster. De acordo com os experimentos realizados, a natureza 2,5-D de imagens de profundidade torna a definição de isotropia relativa uma tarefa desafiadora,

considerando que a camada sobre o qual são espalhados os pontos 3-D de um plano é influenciada diretamente pelo ângulo relativo entre o eixo principal da câmera e a normal do plano. Além disso, aqui é defendido que um único parâmetro (s_t) na mesma unidade de medida assumida pelo dispositivo tende a ser mais intuitiva para o usuário.

4.2 Ajuste de Kernel Gaussiano

Seja $\mathcal{C}$ um cluster contendo pontos aproximadamente coplanares em um nó-folha da *quad-tree* (os m pixels válidos em $\mathcal{R}$), cujos autovetores unitários $V_{(x,y,z)} = \{\overrightarrow{v}_1, \overrightarrow{v}_2, \overrightarrow{v}_3\}$ e autovalores $\Lambda_{(x,y,z)} = \{\lambda_1, \lambda_2, \lambda_3\}$ foram computados no Algoritmo 1 (linha 4). O plano que melhor se ajusta aos pontos em $\mathcal{C}$ possui vetor normal $\overrightarrow{n} = \overrightarrow{v}_1 = (n_x, n_y, n_z)^T$ e passa pelo centróide $\mu_{(x,y,z)}$ (4.2) dos pontos. Esse plano é mapeado para o espaço de parâmetros esféricos usando:

$$\mu_{(\rho,\phi,\theta)} = \begin{pmatrix} \mu_\rho \\ \mu_\phi \\ \mu_\theta \end{pmatrix} = \begin{pmatrix} n_x\,\mu_x + n_y\,\mu_y + n_z\,\mu_z \\ \cos^{-1}(n_z) \\ \tan^{-1}\left(\frac{n_y}{n_x}\right) \end{pmatrix}. \tag{4.5}$$

Assim, o kernel gaussiano que avalia pesos para os votos será centrado nos parâmetros $\mu_{(\rho,\phi,\theta)}$ do plano que melhor se ajusta aos pontos no cluster, *i.e.,* o plano médio.

A matriz de covariância do kernel gaussiano pode ser obtida da covariância das amostras de entrada (4.2) usando propagação de erros em primeira ordem:

$$\Sigma_{(\rho,\phi,\theta)} = \begin{pmatrix} \sigma_{\rho\rho} & \sigma_{\rho\phi} & \sigma_{\rho\theta} \\ \sigma_{\rho\phi} & \sigma_{\phi\phi} & \sigma_{\phi\theta} \\ \sigma_{\rho\theta} & \sigma_{\phi\theta} & \sigma_{\theta\theta} \end{pmatrix} = J\,\Sigma_{(x,y,z)}\,J^T, \tag{4.6}$$

onde J é a matriz Jacobiana de (4.5):

$$J = \begin{pmatrix} \partial_x\rho & \partial_y\rho & \partial_z\rho \\ \partial_x\phi & \partial_y\phi & \partial_z\phi \\ \partial_x\theta & \partial_y\theta & \partial_z\theta \end{pmatrix} = \begin{pmatrix} n_x & n_y & n_z \\ \frac{n_x\,n_z}{\rho\,\omega} & \frac{n_y\,n_z}{\rho\,\omega} & -\frac{\omega}{\rho} \\ -\frac{n_y}{\omega} & \frac{n_x}{\omega} & 0 \end{pmatrix},$$

com $\omega = \sqrt{n_x^2 + n_y^2}$.

Se todas as amostras presentes em um nó forem perfeitamente alinhadas sobre um plano, *i.e.,* sem qualquer ruído, o desvio padrão associado a ρ é igual a zero e, por consequência, sua variância, $\sigma_{\rho\rho} = 0$, o que tornaria $\Sigma_{(\rho,\phi,\theta)}$ singular. Visando evitar este caso específico durante o procedimento de votação, nos experimentos realizados, foi adicionado um valor pequeno $\varepsilon = 0{,}001$ a σ_ρ. Esta prática não influencia nos resultados

mas evita a singularidade da matriz.

4.3 Mapa de Acumuladores Esférico

Borrmann *et al.* [31] demonstraram que o uso de um mapa de acumuladores esférico $\mathcal{A}$ é ideal para HTs adaptadas à detecção de planos em espaços tridimensionais e afirmam que o modelo de mapa de acumuladores mais comumente utilizado, baseado em uma matriz tridimensional simples, favorece alguns parâmetros, de forma desigual. O mapa esférico de Borrmann *et al.* no entanto, foi construído para aplicar o mesmo grau de importância para cada célula de $\mathcal{A}$. Na D-KHT, utilizou-se o mapa esférico proposto por Borrmann's *et al.* para o acúmulo dos votos. Na implementação feita, o espaço de parâmetros esférico define planos cujo vetor normal apontam na direção oposta à origem do espaço cartesiano 3-D. Consideramos o centro da câmera e seu eixo principal, respectivamente, como a origem e o eixo z do espaço. Assim, definimos $\theta \in [-\pi, +\pi)$ com sua discretização dada em função do ângulo de elevação ϕ, restrito a $\phi \in [0, \pi)$. A quantidade de células presentes no eixo θ é dada por [31]:

$$N_\theta(\phi) = \max(1, 2\,N_\phi\,\sin(\phi)), \tag{4.7}$$

onde N_ϕ corresponde ao número de acumuladores no eixo ϕ. É assumida discretização linear para os eixos ϕ e θ de $\mathcal{A}$. É importante notar que o domínio trabalhado é cíclico. Essa observação deve ser levada em conta durante os processos de votação e detecção de picos, descritos das Seções 4.4 e 4.5, respectivamente.

O parâmetro $\rho \in [0, \rho_{high}]$ está relacionado à distância do plano à origem. Além disso, seu limite superior define o raio do espaço de parâmetros esférico. Foi escolhido ρ_{high} como a distância assumida entre a câmera e a amostra mais distante na imagem de profundidade. Os valores discretos assumidos por ρ em $\mathcal{A}$ são definidos por interpolação linear do intervalo $[0, \rho_{high}]$ em N_ρ amostras. Os valores de N_ρ e N_ϕ (utilizado em (4.7)) são definidos pelo usuário de acordo com a granularidade da detecção esperada. O Capítulo 5 indica os valores utilizados nos experimentos.

Seguindo a ideia de Limberger e Oliveira, o mapa de acumuladores é mantido como um mapa associativo que tem, para cada par $\{\phi, \theta\}$ possível, um vetor 1-D representando os acumuladores associados aos valores discretos assumidos por ρ. Todos os elementos do mapa associativo são criados antes do processo de votação, enquanto a terceira dimensão, ρ, é alocada conforme necessária durante o processo de votação. Tal construção garante uma alocação esparsa do mapa de parâmetros, levando à redução no consumo total de

memória.

4.4 Votação Baseada em Kernels

Na votação (Figura 4.1, passo ii), são utilizados kernels gaussianos computados de acordo com (4.5) e (4.6) para atualizar o mapa esférico de acumuladores $\mathcal{A}$ descrito na Seção 4.3. Uma vez computado o vetor de parâmetros $\mu_{(\rho,\phi,\theta)}$ (4.5) do plano médio de um cluster, e a matriz de covariância $\Sigma_{(\rho,\phi,\theta)}$ (4.6), o passo seguinte é incrementar as células de $\mathcal{A}$ que estão abaixo da gaussiana com um ponto de corte de até dois desvios padrões. Assim, a contribuição de um kernel é considerada apenas para as células cujo vetor de parâmetros $q = (\rho, \phi, \theta)^T$ satisfaz:

$$f(q) \geq f\left(\mu_{(\rho,\phi,\theta)} + 2\sqrt{\kappa}\,\overrightarrow{u}\right), \tag{4.8}$$

onde $\{\overrightarrow{u}, \kappa\}$ corresponde a qualquer par autovetor-autovalor de $\Sigma_{(\rho,\phi,\theta)}$, e f denota a função de densidade de probabilidade (FDP) da distribuição gaussiana [32]:

$$f(q) = \frac{1}{\sqrt{(2\pi)^3 \left|\Sigma_{(\rho,\phi,\theta)}\right|}} \exp\left(-\frac{1}{2}\,\delta^T\,\Sigma^{-1}_{(\rho,\phi,\theta)}\,\delta\right), \tag{4.9}$$

para $\delta = q - \mu_{(\rho,\phi,\theta)}$. $|\square|$ e $\square^{-1}$ denotam, respectivamente, o determinante e o inverso de uma matriz. Perceba que o lado direito de (4.8) equivale a um valor escalar constante diferente para cada kernel.

Sendo $\mathcal{A}$ um domínio discreto, a votação é iniciada na célula que contém $\mu_{(\rho,\phi,\theta)}$, endereçada por:

$$\rho_{index} = \left\lfloor \frac{\mu_\rho}{\rho_{high}}\,(N_\rho - 1) \right\rfloor, \tag{4.10a}$$

$$\phi_{index} = \left\lfloor \frac{\mu_\phi}{\pi}\,N_\phi \right\rfloor, \tag{4.10b}$$

$$\theta_{index} = \left\lfloor \frac{\mu_\theta + \pi}{2\,\pi}\,N_\theta\left(\mu_\phi\right) \right\rfloor, \tag{4.10c}$$

onde $\lfloor\square\rfloor$ corresponde à função piso, e $N_\theta(\square)$ é definido por (4.7). Células vizinhas são selecionadas para votação através do mesmo princípio utilizado no algoritmo de inundação (em inglês, *flood fill*), sendo (4.8) o critério de parada.

É importante notar que a densidade de pontos nos clusters varia com a proximidade da câmera, enquanto a integral de (4.9) deve se manter igual a um para que o axioma de probabilidade seja satisfeito [32], ou seja, caso o valor obtido de 4.9 seja utilizado diretamente na votação, a quantidade de votos não será correta, dado que o volume

abaixo de f em (4.9) é sempre um. Logo, a mesma importância seria dada a cada cluster, independentemente de seu tamanho e quantidade de amostras. Portanto, os votos de um kernel para cada célula devem ser computados como o produto da avaliação de (4.9) e o fator $w_{\mathcal{C}}$ (4.11) computado para um dado cluster $\mathcal{C}$ de acordo com a área relativa do nó correspondente e da quantidade relativa de amostras nele presentes. De maneira similar ao proposto por Limberger e Oliveira [8], mas com os devidos ajustes para o modelo de clusterização aqui descrito, é definido:

$$w_{\mathcal{C}} = w_a \frac{\mathcal{R}_{\text{area}}}{\mathcal{I}_{\text{area}}} + w_d \frac{m}{n}, \tag{4.11}$$

onde $\mathcal{I}_{\text{area}}$ e $\mathcal{R}_{\text{area}}$ correspondem às áreas (em pixels) da imagem e da região retangular $\mathcal{R}$ do nó, respectivamente. n é o número de amostras em toda a imagem, e m é o número de amostras no cluster. Os valores de w_a e w_d são restritos a $w_a + w_d = 1$. Durante os experimentos, foi confirmada a observação apresentada em [8] de que melhores resultados

Algoritmo 2: Esquema de votação adotado para a D-KHT.

```
1  Procedimento Vote(ρ_index, φ_index, θ_index, g_min)
       Entrada: ρ_index, φ_index e θ_index correspondem aos índices do mapa de
                acumuladores referentes aos parâmetros do plano médio ajustado
                para cada cluster, obtidos pela aplicação direta da eq. 4.10, ao passo
                em que g_min corresponde ao lado direito da eq. 4.8.
       Dados: Q é uma fila inicializada como vazia, f() corresponde ao lado
              esquerdo da inequação (4.8) e bin se refere à uma célula qualquer do
              mapa de acumuladores.
       Resultado: As células do mapa de acumuladores preenchidas com os votos
                  computados
2      adicione a célula endereçada por (ρ_index, φ_index, θ_index) à fila Q ;
3      marque a célula endereçada por (ρ_index, φ_index, θ_index) como VISITADA ;
4      enquanto Q não estiver vazia faça
5          bin ← elemento removido de Q;
6          q ← (ρ, φ, θ) /* Parâmetros de bin obtidos pela Equação (4.10)  */
7          se f(q) ≥ g_min então
8              votos em bin ← votos em bin + (w_C f(q)) ;
9              para cada vizinho v de bin faça
10                 se a célula v não estiver marcada como VISITADA então
11                     adicione a célula v à fila Q ;
12                     marque a célula v como VISITADA ;
13                 fim
14             fim
15         fim
16     fim
17 fim
```

são atingidos com $w_a = 0{,}75$ e $w_d = 0{,}25$. Assim, a área do nó contribui mais com a votação que o número de amostras. O Algoritmo 2 descreve o processo de votação para um cluster $\mathcal{C}$ qualquer.

4.5 Detecção de Máximos Locais

Após o processo de votação, os picos no mapa de acumuladores correspondem aos planos detectados (Figura 4.1, passo (iii)). É essencial, no entanto, aplicar um filtro passa-baixa para consolidar os máximos locais [17] antes de buscar por picos de votos. Nos experimentos realizados ao longo deste trabalho, foi computada a convolução do mapa de acumuladores e um filtro 3-D com seis células de peso 0,1333 conectadas a uma célula central com peso 0,2002. Essa operação de filtragem suaviza o mapa de votação, ajudando a consolidar picos adjacentes como um único plano detectado.

Foi aplicada, em seguida, uma estratégia de subida no gradiente para detectar picos de votos no mapa de acumuladores suavizado. Para cada kernel utilizado no procedimento de votação, *i.e.* o conteúdo da lista populada na linha 6 do Algoritmo 1, é avaliada a célula endereçada pelos parâmetros do plano médio ajustado para o kernel. Caso a célula seja um máximo local, é marcada como um plano detectado. Caso contrário, é escolhido o vizinho com a maior quantidade de votos dentre os vizinhos da célula atual e o processo é repetido até que se encontre um máximo local.

Uma vez que se tenha os parâmetros de cada plano detectado por subida no gradiente, o passo seguinte é reajustar a relevância dos planos para a imagem de entrada. A relevância é dada com função do fator de peso $w_{\mathcal{C}}$ (4.11) de cada cluster que contribuiu com aquele máximo local. A relevância de um pico $\mathcal{P}$ é dada por:

$$w_{\mathcal{P}} = \sum_{k=1}^{M} w_{\mathcal{C}_k},$$

onde $w_{\mathcal{P}}$ corresponde à representatividade do plano detectado na cena e M é a quantidade de clusters que contribuíram na votação para o pico $\mathcal{P}$. Essa estratégia leva a uma melhor ordenação dos planos detectados que o uso do número de votos registrados no acumulador esférico para imagens de profundidade. O Algoritmo 3 detalha o processo de detecção de picos. Por questão de simplicidade na leitura do algoritmo, foi utilizado como entrada o centro de apenas um kernel, levando assim à detecção de apenas um pico. Também visando a simplicidade da leitura do algoritmo, não foi descrita a maneira como se lida com a situação em que dois ou mais kernels levam à detecção de um mesmo pico. Na

prática, os picos detectados são adicionados a uma estrutura de dados do tipo *set*, que corresponde a um vetor em que não há repetição nos elementos.

Algoritmo 3: Esquema de detecção de máximos locais adotado para a D-KHT.

```
1  Procedimento PeakDetection(ρ_index, φ_index, θ_index)
       Entrada: ρ_index, φ_index e θ_index correspondem aos índices do mapa de
                acumuladores referentes aos parâmetros do plano médio ajustado
                para cada cluster, obtidos pela aplicação direta da eq. 4.10
       Resultado: A célula do mapa de acumuladores correspondente ao máximo
                  local mais próximo a C
2      nextBin ← célula endereçada por (ρ_index, φ_index, θ_index) ;
3      nextVotes ← votos presentes em nextBin;
4      repita
5          currBin ← nextBin;
6          currVotes ← quantidade de votos presentes em currBin;
7          nextVotes ← 0;
8          para cada vizinho v de currBin faça
9              se a quantidade de votos em v é maior que em nextVotes então
10                 nextVotes ← quantidade de votos presentes na célula v ;
11                 nextBin ← v ;
12             fim
13         fim
14     enquanto nextVotes > currVotes;
15     retorna nextBin;
16 fim
```

Dado o vetor de parâmetros $(\rho, \phi, \theta)^T$ de um plano detectado, seu vetor normal $\overrightarrow{n}$ é computado como:

$$\overrightarrow{n} = \begin{pmatrix} n_x \\ n_y \\ n_z \end{pmatrix} = \begin{pmatrix} \sin\phi\,\cos\theta \\ \sin\phi\,\sin\theta \\ \cos\phi \end{pmatrix}.$$

O vetor normal $\overrightarrow{n}$ e a distância ρ definem a equação normal do plano (2.2).

4.6 Discussão

Ao longo desse Capítulo foi apresentado o algoritmo proposto para a detecção de planos em imagens de profundidade. Foram demonstradas as aplicações de conceitos e restrições introduzidos no Capítulo 3, bem como seus impactos no tempo de execução e nas estruturas de dados utilizadas.

A abordagem proposta para detecção de planos é composta de três passos, cada um

com complexidade assintótica linear. Portanto, a complexidade de tempo da D-KHT também é da ordem de $O(n)$. Sem perda de generalidade, é possível assumir como entrada uma imagem de profundidade com $d \times d$ pixels e $n = d^2$ amostras para $d \geq 2$ sendo potência de dois. De acordo com a Seção 4.1, primeiramente são computadas as nove SATs para, em seguida, construir uma *quadtree* implícita, visando a subdivisão da imagem em clusters cujos pontos são aproximadamente coplanares. O custo para o cálculo de cada SAT é $O(n)$ [28]. Assumindo o cenário de pior caso, a *quadtree* subdividiria as amostras em $n/4$ clusters (nós folha) definidos por regiões da imagem de tamanho 2×2. Nesse caso, a altura da árvore seria $\log_4 n$, levando a $\sum_{l=1}^{log_4 n} 4^{l-1} = (n-1)/3$ nós. De acordo com o Algoritmo 1, para cada nó é necessário computar uma matriz de covariância $\Sigma_{(x,y,z)}$ de tamanho 3×3 e seus autovetores e autovalores. A utilização de SATs permite o cálculo de $\Sigma_{(x,y,z)}$ em tempo $O(1)$. A decomposição de $\Sigma_{(x,y,z)}$ também é realizada em tempo $O(1)$. Portanto, as operações no procedimento de clusterização são realizados em tempo $O(n)$.

O procedimento de votação consiste em aplicar propagação de incertezas em primeira ordem para computar a distribuição gaussiana da incerteza associada ao plano que melhor se ajusta a cada cluster e atualizar o mapa de acumuladores utilizando essas distribuições (ver Algoritmo 2). A primeira operação é claramente executada em tempo $O(1)$ para cada cluster. A quantidade de células do mapa de acumuladores atualizados por cada cluster depende da distribuição das amostras no cluster e da discretização do mapa de acumuladores. Mesmo assim, essa quantidade é bem menor que n, de forma que é possível considerar como um valor constante, levando a tempo $O(1)$ para cada cluster. No pior caso, são gerados $n/4$ clusters. Assim, o custo assintótico para o procedimento de votação é $O(n)$. É importante notar que, em casos em que a hajam poucas células ativadas pelo processo de votação, *i.e.*, na situação em que os votos estejam concentrados em uma mesma região do mapa de acumuladores, o custo computacional da subida no gradiente é maior em relação à ordenação. O caso extremo poderia ser observado se todos os clusters votarem sobre a mesma célula do acumulador, levando a um custo computacional linear para a subida no gradiente ao passo em que o custo para a ordenação seria constante.

Por fim, a aplicação do filtro passa-baixa em uma célula do mapa de acumuladores é realizada em tempo $O(1)$, com tempo total de $O(n)$, considerando que apenas células que receberam votos são filtradas. A subida no gradiente é realizada para cada cluster (ver Algoritmo 3), levando a uma pequena quantidade de acessos às células do mapa de acumuladores para cada cluster (muito menor que n). Assim, a complexidade

de tempo para a detecção de picos é $O(n)$, e a complexidade assintótica total da D-KHT é $O(n+n+n) = O(n)$.

A aplicação de SAT sobre o cálculo da matriz de covariâncias e a construção e manutenção de uma *quadtree* implícita, gerando a redução da complexidade do cálculo da matriz de covariâncias e a dispensabilidade da manipulação de pontos pertencentes a cada nó durante o processo de subdivisão, foram de fundamental importância para a performance do algoritmo proposto, assim como a utilização de subida no gradiente para a detecção de máximos locais, que reduziu a complexidade da detecção de picos.

A votação por kernels gaussianos no espaço de parâmetros permitiu a geração de um mapa de acumuladores esparso. Essa característica proporcionou, além de um impacto positivo no tempo de execução do algoritmo por evitar a votação por força bruta e pela rápida detecção de máximos locais, uma estabilidade no que diz respeito à detecção de planos espúrios, consequência direta de HTs baseadas em Kernel.

Capítulo 5

Experimentos e Resultados

A performance da D-KHT foi comparada com o estado da arte utilizando a implementação em C++ da abordagem proposta e a implementação em C++ da 3-D KHT, fornecida por Limberger e Oliveira. Ambas as implementações utilizam `dlib` [33] para computar as decomposições em autovalores e autovetores. No entanto, enquanto a 3-D KHT utiliza OpenMP para paralelizar o procedimento de clusterização, nossa implementação é estritamente sequencial. A D-KHT não foi comparada com técnicas como RANSAC, considerando que esta retorna apenas um plano por execução. Quanto a SG, não foi considerado um comparativo posto que a técnica não trabalha bem com descontinuidades. Esse problema pode ser melhor explicado com base na observação da Figura 5.1, gerada pela D-KHT e correspondente ao frame 294 da base de imagens *Living Room*, em que as portas do balcão (na cor rosa), pertencentes ao mesmo plano, seriam identificadas como planos diferentes na SG, dada a descontinuidade entre as regiões. Para a D-KHT, no entanto, esse problema não ocorre. Assim, os resultados obtidos foram comparados com os resultados obtidos pela 3-D KHT [8]. Os códigos em C++ foram compilados com Visual Studio 2013 e os experimentos foram executados em um processador Intel Core i7-4770 com 3,4 GHz e 16 Gb de RAM. Os códigos foram compilados para a arquitetura 64-bits visando a exploração total do potencial da máquina.

Foram utilizadas imagens de profundidade de quatro bases de imagens diferentes nos experimentos. As bases de imagens sintéticas foram fornecidos por Choi *et al.* [15]: *Living Room* e *Office*, cujos frames 294 e 2453 são apresentados nas Figuras 1.1 e 5.3, respectivamente. Bases sintéticas não incluem ruídos ou erros sistemáticos introduzidos pelo dispositivo de captura. Os frames de bases de imagens de cenas reais utilizadas foram: *Copy Room* e *Cube*. A primeira base foi capturada por Zhou e Koltun [34] com uma câmera Asus Xtion PRO LIVE. A Figura 5.4 mostra os frames 1063, 1791, 2297, e 4161

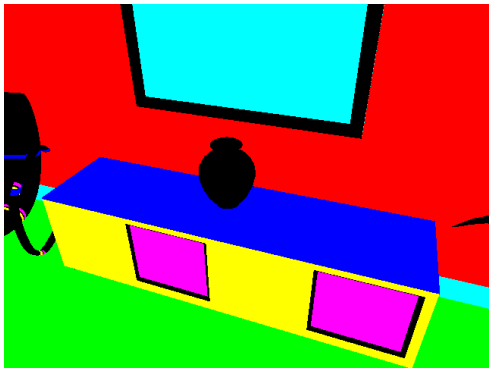


Figura 5.1: Saída da D-KHT para o frame 294 da base *Living Room* com discretização mais fina que a utilizada para a Figura 1.1. Destaque para as regiões em rosa na imagem, corretamente detectadas como coplanares mesmo com descontinuidade na imagem de profundidade. Para essa detecção, os parâmetros utilizados foram $N_\rho = 350$, $N_\phi = 430$, $s_{ms} = 150$ e $s_t = 0{,}0126$. A detecção foi feita em 28,63 ms.

da base de imagens *Copy Room*. A base de imagens *Cube* foi capturada pelo autor durante os experimentos utilizando um Structure Sensor. O frame único é apresentado na Figura 4.2. Bases de imagens reais incluem distorções óticas introduzidas pelo dispositivo de captura, ruído e porções da imagem sem dados de profundidade. Todas as imagens de profundidade possuem 640×480 pixels, com profundidades variando de 1 a ρ_{high}, onde ρ_{high} é apresentado na terceira coluna da Tabela 5.1 para cada imagem. Foi utilizada a expressão (4.1) para converter as imagens de profundidade em nuvens de pontos para servir como entrada para a 3-D KHT.

Nos planos apresentados nas Figuras 1.1, 5.3, e 5.4 foram pintados (com cores aleatórias) todos os pontos de entrada suficientemente próximos a algum plano detectado. Um problema notável nessa estratégia de pintura é que os planos detectados podem parecer incluir pontos que não contribuíram com a formação de um pico de votos. Perceba, por exemplo, as listras na cadeira presente na Figura 1.1c, a listra no teclado nas Figuras 5.3c e 5.3d, além dos artefatos visuais similares na Figura 5.4. Essa estratégia de pintura foi adotada considerando que a implementação referência da 3-D KHT não localiza e pinta apenas os pontos que contribuíram com os picos de votos. O uso de uma *quadtree* implícita, no entanto, simplifica a tarefa de localização dos pontos na D-KHT. Considerando que seria injusto prover uma comparação visual utilizando dois métodos de pintura diferentes, o mais simples foi utilizado. Acredita-se que, de forma similar à D-KHT, a 3-D KHT poderia ser adaptada para obter os pontos associados a cada plano suporte. Desse modo, os artefatos visuais nos exemplos demonstrados não devem ser vistos como problemas práticos em nenhuma das duas técnicas.

Um resumo detalhado dos tempos de execução para cada passo das estratégias de

Tabela 5.1: Comparação da performance da D-KHT e da 3-D KHT em relação aos exemplos produzidos e demonstrados ao longo deste trabalho. O tempo nas colunas *Clusterização*, *Votação*, *Picos*, e *Total* está expresso em milissegundos.

Entrada			D-KHT						3-D KHT					
Base (frame)	Amostras	ρ_{high}	Clusterização	Votação	Picos	Total	Clusters	Planos	Clusterização	Votação	Picos	Total	Clusters	Planos
Living Room (294)	307.200	2.848	4,685	8,194	2,465	15,344	82	4	13,009	39,937	29,920	82,866	67	4
Office (2453)	307.200	4.025	3,341	14,475	3,474	21,290	105	6	15,001	55,012	49,910	119,923	170	6
Copy Room (1063)	260.636	1.805	5,098	2,454	3,069	10,621	70	11	12,363	5,001	10,001	27,365	62	10
Copy Room (1791)	271.359	2.329	5,059	2,777	2,766	10,602	68	6	19,994	25,004	60,015	105,013	87	10
Copy Room (2297)	266.917	1.688	3,076	2,568	2,434	8,078	42	8	14,106	10,001	40,099	64,206	75	9
Copy Room (4161)	270.089	2.479	4,401	1,413	1,197	7,011	26	4	14,011	5.102	18,021	37,134	22	4
Cube (1)	225.296	2.051	2,117	3,770	2,491	8,378	237	4	11,033	10,700	23,914	45,647	44	4

detecção de planos é apresentada na Tabela 5.1 para sete imagens de entrada diferentes, mostradas nas Figuras 1.1, 4.2, 5.3 e 5.4. A complexidade das cenas é indicada pelo número de amostras (ver coluna *Amostras*) e a distribuição de superfícies planares e não-planares que pode ser vista nas respectivas imagens de profundidade. As colunas *Clusterização*, *Votação*, e *Picos* mostram o tempo de execução (em milissegundos) de cada etapa da D-KHT e da 3-D KHT. Os tempos de execução foram computados como a média de 50 execuções. As colunas *Total* apresentam o somatório das três colunas anteriores. Na Tabela 5.1, as últimas duas colunas de cada técnica mostram o número de clusters produzidos e o número de planos obtidos pelo procedimento de detecção de picos.

Os tempos de execução das técnicas são afetados pelos parâmetros selecionados. Assim, foram escolhidos valores de parâmetros que otimizem o tempo de execução de cada algoritmo de forma individual para cada base de imagens, ao passo em que se tentava manter uma qualidade de detecção equivalente para ambas as técnicas. Além disso, con-

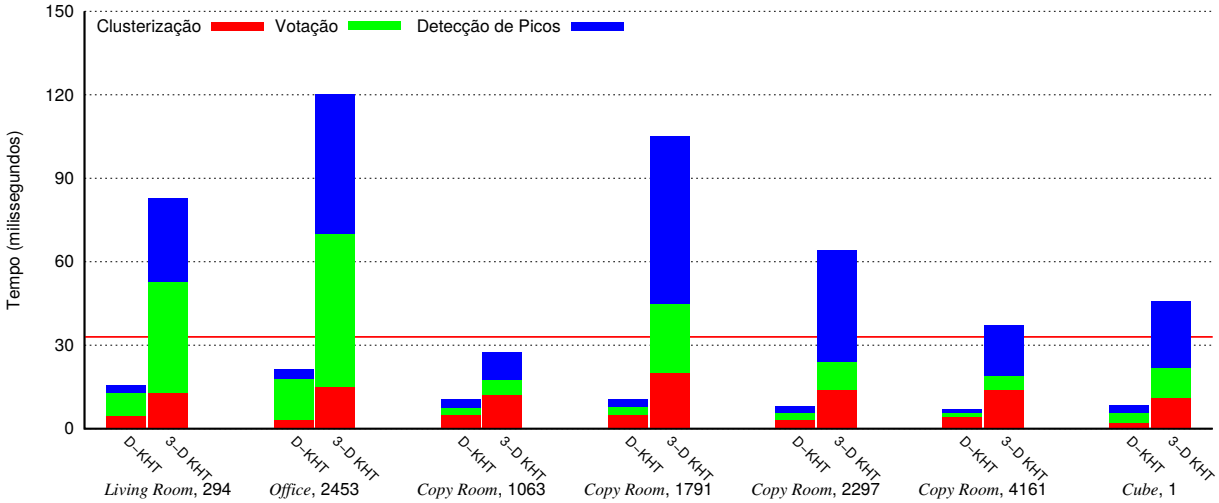
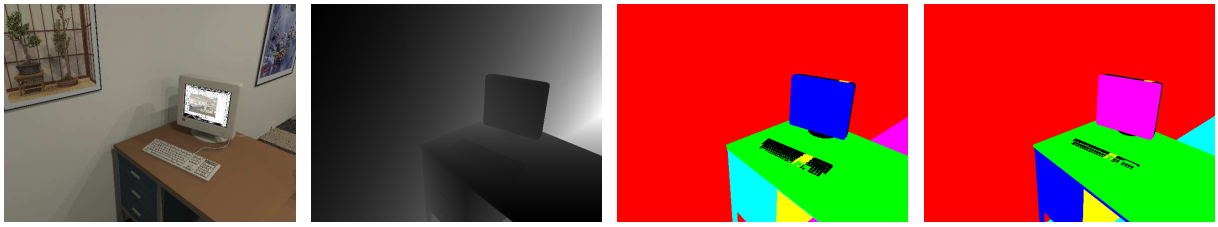


Figura 5.2: Gráfico comparativo dos tempos de execução da D-KHT e da 3-D KHT para os frames de bases contendo imagens de cenas reais e sintéticas. A linha vermelha define a marca de 33 ms. Tempo de processamento abaixo dessa linha representa execução em tempo real (30 fps ou mais).



(a) Imagem Colorida (b) Im. de profundidade (c) D-KHT (d) 3-D KHT

Figura 5.3: O frame 2453 da base de imagens *Office* ilustra a capacidade de detecção da D-KHT e da 3-D KHT em um cenário sem ruído. Da esquerda para a direita: (a) imagem colorida; (b) imagem de profundidade de entrada; (c) planos detectados em (b) pela D-KHT; e (d) planos detectados em (b) pela 3-D KHT. Perceba que ambas as técnicas são capazes de detectar os principais planos da cena. A D-KHT, no entanto, é $\sim$5 vezes mais rápida que a 3-D KHT nesse exemplo.

siderando que o espaço de parâmetros e o modelo de mapa de acumuladores é o mesmo para ambas as técnicas, foi possível comparar as duas técnicas sob o mesmo critério de tempo de execução. A Tabela 5.2 detalha os parâmetros assumidos nos experimentos. Os parâmetros N_ϕ, N_ρ, e s_{ms} da D-KHT correspondem, respectivamente, aos parâmetros ϕ_{max}, ρ_{max}, e s_{ms} da 3-D KHT. O último parâmetro da D-KHT é a espessura máxima do um plano ajustado (s_t). A 3-D KHT utiliza três outros parâmetros: s_α e s_β são as tolerâncias relativas para espessura e isotropia de plano, respectivamente, enquanto que s_{level} corresponde ao primeiro nível da *octree* para avaliação de coplanaridade aproximada. A D-KHT considera a avaliação de todos os níveis durante a subdivisão da *quadtree*. Todos os parâmetros da execução sobre os frames da base *Copy Room* foram mantidos no processamento das imagens visando a verificação da dependência de parâmetros em diferentes pontos de vista da mesma cena.

A Tabela 5.1 mostra que a abordagem proposta pode alcançar $\sim$119,4 fps para imagens de profundidade de cenas reais simples (*e.g.,* base *Cube*, frame 1), e de $\sim$94,2 a $\sim$142,6 fps para bases reais mais complexas (*e.g.,* base *Copy Room*, frames 1063 e 4161). A 3-D KHT, por outro lado, foi capaz de atingir performance de tempo real (*i.e.,* mais de 30 fps) apenas para o frame 1063 da base *Copy Room*, atingindo $\sim$9,5 a $\sim$26,9 fps para

Tabela 5.2: Parâmetros utilizados nos experimentos realizados.

Base	Ambas as Técnicas			D-KHT	3-D KHT		
	N_ρ	N_ϕ	s_{ms}	s_t	s_α	s_β	s_{level}
Living Room	78	180	150	0,0166	0,000066	0,73	5
Office	179	135	200	0,025	0,0003	0,70	5
Copy Room	55	70	150	0,21	0,009	0,06	2
Cube	80	30	200	0,36	0,006	0,00	3

outras imagens de profundidade de cenas reais.

O uso de SATs e de uma *quadtree* implícita faz o passo de clusterização da D-KHT de $\sim$2 a $\sim$5 vezes mais rápida que a estratégia de clusterização baseada em *octree* da 3-D KHT. Para cenas sintéticas, *i.e.,* onde todos os pixels possuem informação de profundidade válida, o procedimento de votação domina o tempo de ambas as técnicas. Para bases com imagens de cenas reais, a subdivisão da *quadtree* domina o tempo da D-KHT (exceto para os frames da base *Cube*), enquanto a detecção de picos é o passo mais lento da 3-D KHT (exceto para o frame 1063 da base *Copy Room*). Por fim, é bastante perceptível a diferença entre as performances de detecção de máximos locais em ambas as técnicas. A estratégia de subida no gradiente da D-KHT é cerca de 22 vezes mais rápida que a 3-D KHT nesse passo (*e.g., Copy Room*, frame 1791). Os dados de tempos de execução são melhor visualizados na Figura 5.2, onde é possível notar os tempos da D-KHT abaixo a linha vermelha, indicativa de 33 ms. Além disso, é possível notar também os menores tempos de execução sobretudo nas etapas de clusterização e detecção de picos.

Uma inspeção cuidadosa da Tabela 5.1 revela que apesar do tempo de votação para a D-KHT se relacionar ao número de clusters, o passo de clusterização para o frame da base *Cube* obteve, de longe, mais clusters que as outras imagens, mas o tempo de votação não foi afetado na mesma proporção. O número de clusters pode ser explicado pela quantidade de ruído presente nessa imagem de profundidade em particular, o que levou a uma segmentação mais detalhada das superfícies. Conforme a imagem era subdividida, a incerteza relativa nos clusters resultantes era reduzida. Como resultado, os kernels gaussianos utilizados no procedimento de votação tornaram-se menores permitindo, assim, um procedimento de votação mais rápido.

5.1 Discussão

Foi apresentada, ao longo deste Capítulo, uma análise dos resultados obtidos pela execução do algoritmo proposto sobre frames de bases de imagens de profundidade disponíveis na literatura. Foram analisados critérios como tempo de execução e quantidade de planos detectados e estabelecido um comparativo com o estado da arte.

Como qualquer outra abordagem baseada em HT que utiliza um mapa de acumuladores discreto, a performance e robustez da técnica proposta são restritas à discretização apropriada do espaço de parâmetros e à complexidade da cena. Além disso, a presença de superfícies não-planares na cena pode levar à detecção de planos espúrios. Felizmente,

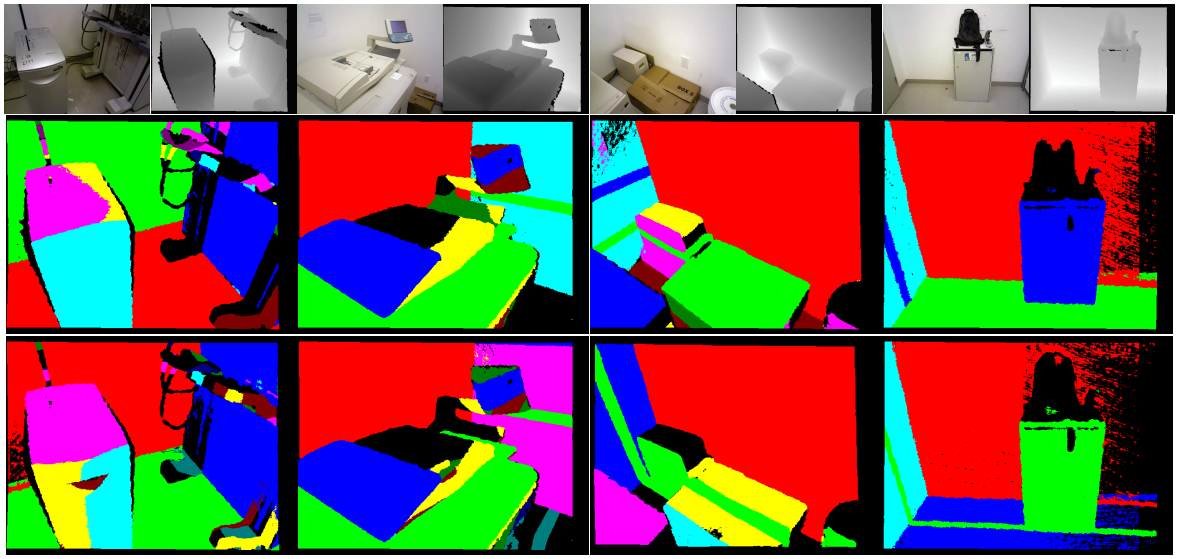


Figura 5.4: Comparação dos resultados da detecção de planos em cenas reais. Da esquerda para a direita, frames 1063, 1791, 2297, e 4161 da base *Copy Room*. A primeira linha mostra as imagens em cor e de profundidade. A segunda e terceira linhas apresentam os planos detectados (identificados por cores) pela D-KHT e pela 3-D KHT, respectivamente.

tais planos costumam ter baixa representatividade e podem ser descartados com a escolha apropriada de um limiar de supressão, recurso que não foi utilizado em nenhum dos resultados apresentados nesse trabalho. Planos paralelos próximos podem ser detectados como um único plano em bases ruidosas. Por fim, câmeras de profundidade restringem o conjunto de planos que podem ser observados na cena (*e.g.,* planos atrás da câmera não podem ser vistos). Assim, o espaço de parâmetros adotado acaba por incluir regiões que nunca são alteradas pelo procedimento de votação.

Capítulo 6

Estudo de Caso

A técnica apresentada nesse trabalho foi desenvolvida com base na necessidade prática de um algoritmo para detecção de planos no contexto de um projeto maior, visando a solução de problemas encontrados no decorrer do desenvolvimento desse projeto. Assim, se faz interessante discorrer sobre as características que envolvem a aplicação da técnica, os problemas encontrados e as soluções propostas.

6.1 Descrição do Projeto

Ambientes internos são tipicamente compostos por regiões planares. Assim, a D-KHT atua como o passo de entrada para um projeto de mais alto nível que visa a reconstrução da geometria de ambientes internos por planos, tendo como entrada imagens de profundidade capturadas por dispositivos móveis em tempo real. Dispositivos móveis recém inseridos no mercado, como o *Structure Sensor* [1] e o *Project Tango* [2] (Figura 6.1) permitem a captura de imagens de profundidade com boa resolução de profundidade (16 bits) e executam triangulação ativa com luz infravermelha. Considerando o baixo consumo de memória e complexidade do algoritmo proposto e a sua execução em tempo real, tais dispositivos devem ser suficientes para possibilitar a execução dos algoritmos ainda durante a captura. A D-KHT, no entanto, trabalha apenas com a entrada da técnica de reconstrução. Além disso, múltiplos frames são processados durante a captura e as partes do mesmo plano identificados em mais de um frame devem ser reconstruídos como um só. Assim, os demais passos da técnica envolvem o mapeamento de texturas da cena para a reconstrução, a determinação das matrizes de parâmetros extrínsecos de câmera entre os frames capturados e o mapeamento e reconstrução das estruturas não-planares, através de triangulação ativa. Nesse Capítulo será tratado o mapeamento de texturas, considerando

(a) Structure Sensor (b) Project Tango

Figura 6.1: Dispositivos utilizados para captura de imagens de profundidade dados e para a execução de todo o algoritmo de reconstrução.

que esse é o passo conectado diretamente com a D-KHT no projeto recebendo, como entrada, a saída da D-KHT.

6.2 Mapeamento de Texturas da Cena

Considerando o algoritmo para reconstrução proposto, uma vez detectados os planos em uma imagem de profundidade, é necessário mapear porções da textura da cena para cada plano. Assim, será descrito, em alto nível, o *pipeline* sugerido por Lúcio [35] para o mapeamento de texturas.

Inicialmente, a imagem é segmentada por plano identificado, gerando várias imagens binárias, cada uma com um plano em destaque. O passo seguinte diz respeito à determinação da curva de bordo da região pertencente a cada plano. Em seguida são triangulados tanto os nós com amostras coplanares quanto os nós com amostras não-coplanares da *quadtree*. Por fim, a triangulação inicial é simplificada e, com base nos vértices dos triângulos pertencentes a essa malha simplificada, são geradas as coordenadas de textura.

6.2.1 Segmentação da imagem

Com base na saída da D-KHT, é feita a segmentação da imagem por plano. Trata-se de um pré-processamento para a determinação da curva de bordo da região pertencente ao plano, que requer uma imagem binária como entrada. Assim, para cada plano detectado sobre a imagem de profundidade, é gerada uma nova imagem contendo apenas aquele plano. A Figura 6.2 mostra as várias imagens binárias geradas.

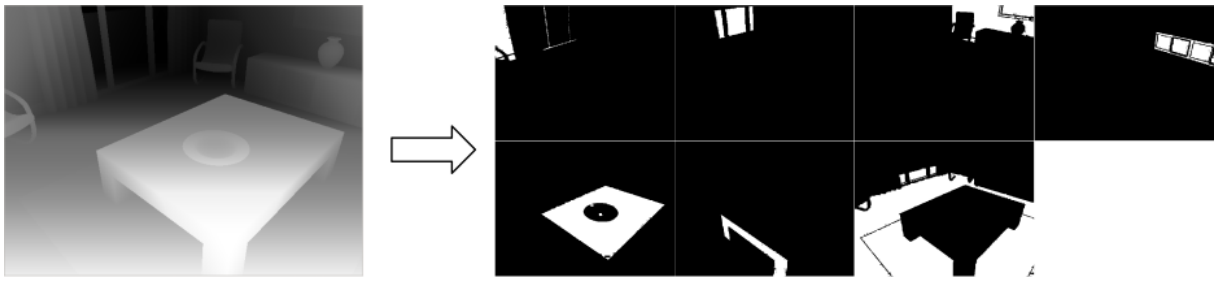


Figura 6.2: Imagem de saída do processo de limiarização por plano que tem como entrada os pontos pertencentes a cada plano detectado pela D-KHT

6.2.2 Determinação da curva de bordo

Nesse passo, a ideia principal é definir o bordo da região da imagem pertencente ao plano, isto é, determinar o polígono associado ao contorno da projeção dessa porção planar na imagem. Para isso, utilizou-se um algoritmo para a avaliação de regiões conexas, o *CrackCode* [36]. O algoritmo é capaz de identificar os contornos externos e internos de regiões. Contornos internos correspondem a buracos no polígonos, sendo indícios de regiões não-planares e que devem ser reconstruídas posteriormente.

6.2.3 Triangulação

Nessa etapa, é feita a triangulação dos nós recebidos como subproduto do processo de subdivisão da *quadtree* da D-KHT. Essa triangulação é bastante trivial. Considerando que cada nó possui formato retangular, a triangulação consiste em traçar uma das diagonais de cada nó, conforme observado na Figura 6.3. Apesar de sua simplicidade, esse passo é essencial para a posterior simplificação da malha triangular. Perceba que não apenas os nós contendo amostras coplanares são triangulados aqui. Nós contendo amostras não-coplanares podem corresponder a buracos na região pertencente ao plano e também precisam ser trianguladas.

6.2.4 Simplificação de malhas triangulares

Uma vez que a determinação da curva de bordo esteja concluída e os bordos e buracos do polígono estejam propriamente marcados, o passo seguinte consiste em simplificar a geometria da cena. O passo de triangulação gera uma malha bastante complexa no que diz respeito à quantidade de vértices e triângulos. Assim, o objetivo desse passo é reduzir essa quantidade e manter o mínimo de triângulos conservando a topologia (incluindo na região de buracos). Para isso, aplicou-se o algoritmo Douglas-Peucker [37]. A Figura 6.4

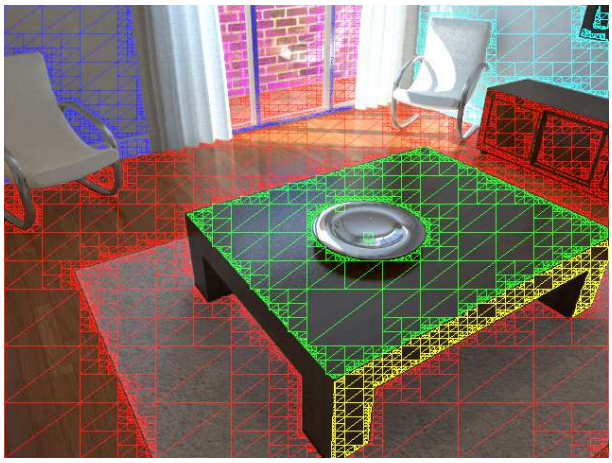


Figura 6.3: Imagem triangulada de acordo com a saída da D-KHT, para os nós contendo amostras aproximadamente coplanares.

exibe o resultado final da aplicação do algoritmo.

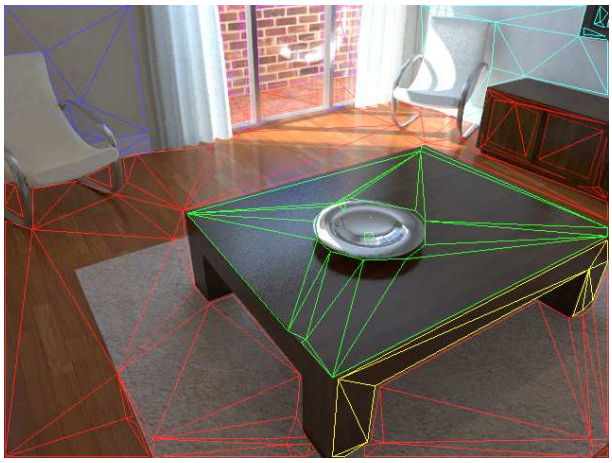


Figura 6.4: Saída do algoritmo Douglas-Peucker aplicado sobre a triangulação exibida na Figura 6.3.

Por fim, com base nos vértices gerados pela triangulação e posterior simplificação da malha de triângulos, são geradas as coordenadas de textura (Figura 6.5).

6.3 Discussão

Ao longo desse Capítulo foi comentado o *pipeline* para mapeamento de texturas, apresentado por Lucio [35] que, em conjunto com o algoritmo introduzido ao longo desse trabalho, é parte integrante de uma técnica para reconstrução da geometria de ambientes internos por aproximação de planos. O *pipeline* consiste na implementação de algoritmos com baixa complexidade computacional visando uma execução rápida, mesmo em dispositivos

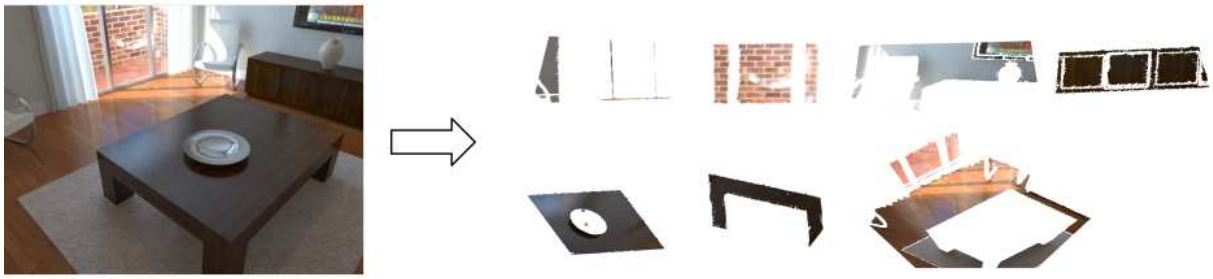


Figura 6.5: Imagens de saída do processo de mapeamento de texturas. À esquerda da seta é apresentada a imagem colorida e à direita é apresentada a textura para cada plano detectado na imagem pela D-KHT

.

com capacidade de processamento limitada. Atualmente, o projeto está em desenvolvimento em parceria entre o IMPA (Instituto Nacional de Matemática Pura e Aplicada) e o Instituto de Computação da UFF (Universidade Federal Fluminense) com previsão de conclusão em 2016.

Capítulo 7

Conclusões e Trabalhos Futuros

A detecção de entidades geométricas em imagens de profundidade e em tempo real possui várias aplicações práticas sendo, desse modo, de grande valia, principalmente se a detecção se dá em ambientes ruidosos sem perda de qualidade ou performance. Este trabalho apresentou uma abordagem em tempo real para a detecção de planos em imagens de profundidade. Neste último capítulo são enumerados os principais pontos apresentados ao longo dessa dissertação. Em seguida, é apresentada uma discussão sobre as vantagens e limitações da técnica apresentada. Por fim, são discutidas as possibilidades e sugestões para trabalhos futuros.

Para garantir o baixo custo computacional da técnica, tirou-se proveito de algumas restrições provenientes do modelo de captura e da estrutura regular da imagem, o que foi fundamental para o custo computacional assintótico linear na primeira etapa do algoritmo. De modo geral, essa etapa diz respeito à clusterização dos pontos de entrada no espaço 2,5-D em regiões aproximadamente coplanares utilizando, para isso, uma *quadtree* e análise de covariância. A definição de imagem de profundidade (ver Seção 3.1) possibilita a aplicação de SATs para o cálculo da matriz de covariâncias em tempo constante para cada cluster.

O passo seguinte é a votação no espaço de parâmetros do plano. Seguindo a abordagem de HT eficiente para planos [8], é possível utilizar cada cluster obtido na etapa anterior, associado à incerteza do ajuste dos pontos modelada como uma distribuição gaussiana, e utilizar essa distribuição para distribuir os votos sobre uma região restrita. Isso evita a criação de um mapa de acumuladores denso.

Por fim, é feita a detecção de máximos locais no mapa de acumuladores. A abordagem para esta etapa é baseada na observação de que os máximos locais não costumam estar

distantes dos parâmetros do plano médio de cada kernel. Assim, é possível utilizar esses parâmetros como semente para um algoritmo iterativo que, a cada iteração, analisa a vizinhança e avança na direção da célula do mapa de acumuladores com mais votos até se estabilizar em um máximo local. Essa subida no gradiente do mapa de acumuladores se mostrou eficiente, conforme demonstrado no Capítulo 5.

A eficácia do método foi demonstrada com base em sua comparação com o estado-da-arte. As entradas adotadas para os experimentos envolviam frames pertencentes à bases de imagens sintéticas e reais. Além disso, as bases analisadas também contavam com a presença de superfícies não-planares para que fosse analisada a resiliência à detecção de planos espúrios, problema comum nesse tipo de dado. A técnica apresentada se mostrou bastante eficiente para imagens de profundidade, chegando a ser executada entre $\sim$3 e $\sim$10 vezes mais rápida que o estado-da-arte.

7.1 Discussão

7.1.1 A influência da discretização nos resultados da técnica apresentada

Como toda técnica baseada em HT, a abordagem proposta é bastante sensível à discretização. Enquanto uma discretização mais grosseira provê um menor tempo de execução do algoritmo, porém com menor qualidade da detecção, discretizações mais finas geram o efeito oposto. Tal efeito pode ser observado ao comparar a Figura 1.1 e a Figura 5.1. A segunda, com discretização mais fina, foi executada em pouco mais de 28 ms, contra 15,34 ms da primeira. Isso leva a cerca de 65 fps para a imagem da Figura 1.1 contra apenas 34,9 fps para a imagem da Figura 5.1.

7.1.2 Influência do ruído na qualidade da detecção

De modo geral, o ruído natural presente em imagens de profundidade provenientes de cenas reais possui pouca ou quase nenhuma influência sobre a qualidade da detecção. Em imagens muito ruidosas, no entanto, a qualidade da detecção pode ser um pouco afetada, sobretudo em planos paralelos, finos e muito próximos. Planos com essas características tendem a ser detectados como se fossem apenas um. Essa característica não está restrita apenas à D-KHT. É bastante comum em técnicas baseadas em HT. Em imagens com mais de 50% de ruído impulsivo aleatório, a qualidade da detecção da D-KHT foi bem mais

afetada que a da 3-D KHT. Isso se deve ao fato de que, nesse tipo de ruído, ao converter as coordenadas de pixel da imagem para a unidade de medida padrão da câmera (ver Equação 4.1), acabava havendo certa sobreposição dos pontos ao longo do eixo z. Por não fazer divisões em profundidade, como a 3-D KHT faz, acabava havendo perda na qualidade da detecção nesses casos.

7.2 Trabalhos Futuros

Ao longo desse trabalho foram identificadas questões que servem como direção para a extensão ou melhoria das técnicas aqui discutidas.

7.2.1 Redução do espaço de parâmetros

Durante o desenvolvimento desse trabalho foi observado que algumas regiões do espaço de parâmetros esférico não são acessadas nunca, considerando restrições do *frustum* da câmera, como planos atrás da câmera, por exemplo. Assim, acredita-se que seja possível a redução desse espaço de parâmetros, reduzindo o consumo de memória no processo.

7.2.2 Relação entre parâmetros do algoritmo

Foi observado durante os experimentos que parece haver uma forte relação entre os parâmetros N_ρ e s_t da técnica apresentada. Como trabalho futuro, pretende-se descrever matematicamente essa relação visando a eliminação de um parâmetro da técnica.

Referências

[1] OCCIPITAL. *Structure Sensor*. 2015. Disponível em: <http://structure.io>.

[2] GOOGLE. *Project Tango*. 2015. Disponível em: <https://www.google.com/atap/project-tango/>.

[3] HARTLEY, R. I.; ZISSERMAN, A. *Multiple View Geometry in Computer Vision*. 2nd. ed. [S.l.]: Cambridge University Press, 2004.

[4] KAUCIC, R.; HARTLEY, R.; DANO, N. Plane-based projective reconstruction. In: *Proc. of ICCV*. [S.l.: s.n.], 2001. v. 1, p. 420–427.

[5] TRIGGS, B. Autocalibration from planar scenes. In: *Proc. of ECCV*. [S.l.: s.n.], 1998. v. 1, p. 89–105.

[6] HOLZ, D.; HOLZER, S.; RUSU, R. B.; BEHNKE, S. Real-time plane segmentation using RGB-D cameras. *Lecture Notes in Computer Science*, v. 7416, n. D, p. 306–317, 2012.

[7] SIMON, G.; FITZGIBBON, A. W.; ZISSERMAN, A. Markerless tracking using planar structures in the scene. In: *Proc. of ISAR*. [S.l.: s.n.], 2000. p. 120–128.

[8] LIMBERGER, F. A.; OLIVEIRA, M. M. Real-time detection of planar regions in unorganized point clouds. *Pattern Recognit.*, v. 48, n. 6, p. 2043–2053, 2015.

[9] FISCHLER, M. A.; BOLLES, R. C. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, v. 24, n. 6, p. 381–395, 1981.

[10] FISCHLER, M. A.; BOLLES, R. C. Perceptual organization and curve partitioning. *IEEE Trans. Pattern Anal. Mach. Intell.*, v. 8, n. 1, p. 100–105, 1986.

[11] CHEN, D. S. A data-driven intermediate level feature extraction algorithm. *IEEE Trans. Pattern Anal. Mach. Intell.*, v. 11, n. 7, p. 749–758, 1989.

[12] BESL, P. J.; JAIN, R. C. Segmentation through variable-order surface fitting. *IEEE Trans. Pattern Anal. Mach. Intell.*, v. 10, n. 2, p. 167–192, 2002.

[13] POPPINGA, J.; VASKEVICIUS, N.; BIRK, A.; PATHAK, K. Fast plane detection and polygonalization in noisy 3D range images. In: *Proc. of IROS*. [S.l.: s.n.], 2008. p. 3378–3383.

[14] XIAO, J.; ZHANG, J.; ZHANG, J.; ZHANG, H.; HILDRE, H. P. Fast plane detection for SLAM from noisy range images in both structured and unstructured environments. In: *Proc. of ICMA*. [S.l.: s.n.], 2011. p. 1768–1773.

[15] CHOI, S.; ZHOU, Q.-Y.; KOLTUN, V. Robust reconstruction of indoor scenes. In: *Proc. of CVPR*. [S.l.: s.n.], 2015. p. 5556–5565.

[16] HOUGH, P. V. C. Machine analysis of bubble chamber pictures. In: *Proc. of Int. Conf. on High Energy Accelerators and Instrum.* [S.l.: s.n.], 1959. p. 554–558.

[17] FERNANDES, L. A. F.; OLIVEIRA, M. M. Real-time line detection through an improved Hough transform voting scheme. *Pattern Recognit.*, v. 41, n. 1, p. 299–314, 2008.

[18] DUDA, R. O.; HART, P. E. Use of the Hough transformation to detect lines and curves in pictures. *Commun. ACM*, v. 15, n. 1, p. 11–15, 1972.

[19] VOSSELMAN, G.; GORTE, B. G. H.; SITHOLE, G.; RABBANI, T. Recognizing structure in laser scanner point clouds. In: *Proc. of Conference on Laser scanners for Forest and Landscape assessment and instruments*. [S.l.: s.n.], 2004. XXXVI, 8/W, p. 33–38.

[20] NGUYEN, H. H.; KIM, J.; LEE, Y.; AHMED, N.; LEE, S. Accurate and fast extraction of planar surface patches from 3D point cloud. In: *Proc. of Int. Conf. on Ubiquitous Information Management and Commun.* [S.l.: s.n.], 2013. p. 84.

[21] SCHNABEL, R.; WAHL, R.; KLEIN, R. Efficient RANSAC for point-cloud shape detection. *Comput. Graph. Forum*, v. 26, n. 2, p. 214–226, 2007.

[22] FERNANDES, L. A. F.; OLIVEIRA, M. M. A general framework for subspace detection in unordered multidimensional data. *Pattern Recognit.*, v. 45, n. 9, p. 3566–3579, 2012.

[23] FERNANDES, L. A. F.; OLIVEIRA, M. M. Handling uncertain data in subspace detection. *Pattern Recognit.*, v. 47, n. 10, p. 3225–3241, 2014.

[24] ADAMS, R.; BISCHOF, L. Seeded region growing. *IEEE Trans. Pattern Anal. Mach. Intell.*, v. 16, n. 6, p. 641–647, 1994.

[25] ZHU, J.; WANG, L.; YANG, R.; DAVIS, J. Fusion of time-of-flight depth and stereo for high accuracy depth maps. In: *Proc. of CVPR*. [S.l.: s.n.], 2008. p. 1–8.

[26] SAMET, H. The quadtree and related hierarchical data structures. *ACM Computing Surveys*, v. 16, n. 2, p. 187–260, 1984.

[27] LO, M. W.; HOCKNEY, G.; KWAN, B. Quad-tree visual-calculus analysis of satellite coverage. 2003.

[28] CROW, F. C. Summed-area tables for texture mapping. *SIGGRAPH Comput. Graph.*, v. 18, n. 3, p. 207–212, 1984.

[29] JOLLIFFE, I. T. Principal Component Analysis, Second Edition. *Encyclopedia of Statistics in Behavioral Science*, v. 30, n. 3, p. 487, 2002. ISSN 00401706.

[30] PARRATT, L. Probability and experimental errors in science. *J. Frankl. Inst.-Eng. Appl. Math.*, v. 272, n. 6, p. 527–528, 1961.

[31] BORRMANN, D.; ELSEBERG, J.; LINGEMANN, K.; NüCHTER, A. The 3D Hough transform for plane detection in point clouds: A review and a new accumulator design. *3D Research*, v. 2, n. 2, p. 1–13, 2011.

[32] BERTSEKAS, D. P.; TSITSIKLIS, J. N. *Introduction to Probability*. [S.l.]: Athena Scientific, 2002.

[33] DLIB. *dlib c++ library*. [S.l.]. Disponível em: <http://dlib.net>.

[34] ZHOU, Q.-Y.; KOLTUN, V. Dense scene reconstruction with points of interest. *ACM Trans. Graph.*, v. 32, n. 4, p. 112:1–112:8, 2013.

[35] LUCIO, D. *Uso de Estruturas Planares Extraídas de Imagens RGB-D em Aplicações de Realidada Aumentada*. Dissertação (Mestrado) — Departamento de Informática, PUC-Rio, Rio de Janeiro, Rio de Janeiro, Brasil, 2016.

[36] SHIH, F. Y. *Image Processing and Pattern Recognition: Fundamentals and Techniques*. [S.l.]: John Wiley & Sons, 2010. ISBN 9780470404614.

[37] DOUGLAS, D. H.; PEUCKER, T. K. Algorithms for the Reduction of the Number of Points Required to Represent a Digitized Line or its Caricature. In: *Classics in Cartography: Reflections on Influential Articles from Cartographica*. [S.l.]: John Wiley & Sons, 2011. p. 15–28. ISBN 9780470681749.