Gilberto Antonio Marcon dos Santos

Novas Abordagens para Segmentação de Nuvens de Pontos Aplicadas à Robótica Autônoma e Reconstrução 3D

Goiânia, Brasil

2016

Gilberto Antonio Marcon dos Santos

Novas Abordagens para Segmentação de Nuvens de Pontos Aplicadas à Robótica Autônoma e Reconstrução 3D

Trabalho apresentado como requisito parcial para a conclusão do Mestrado em Engenharia Elétrica e de Computação da Universidade Federal de Goiás.

Universidade Federal de Goiás – UFG

Escola de Engenharia Elétrica, Mecânica e de Computação – EMC

Engenharia Elétrica e de Computação

Orientador: Prof^{o} Dr. Gélson da Cruz Júnior

Coorientador: Prof^{o} Dr. Cássio Dener Noronha Vinhal

Goiânia, Brasil

2016

Ficha de identificação da obra elaborada pelo autor, através do
Programa de Geração Automática do Sistema de Bibliotecas da UFG.

Marcon dos Santos, Gilberto Antonio
Novas Abordagens para Segmentação de Nuvens de Pontos Aplicadas à Robótica Autônoma e Reconstrução 3D [manuscrito] / Gilberto Antonio Marcon dos Santos. - 2016.
CXII, 112 f.: il.

Orientador: Prof. Dr. Gelson Cruz junior; co-orientador Dr. Cassio Dener Noronha Vinhal.
Dissertação (Mestrado) - Universidade Federal de Goiás, Escola de Engenharia Elétrica (EEEC), Programa de Pós-Graduação em Engenharia Elétrica e de Computação, Goiânia, 2016.
Bibliografia.
Inclui siglas, abreviaturas, gráfico, tabelas, algoritmos, lista de figuras.

1. Segmentação de nuvens de pontos. 2. Extração de piso. 3. Consenso de amostragem. 4. Estratégias evolucionárias. 5. Redes neurais artificiais. I. Cruz junior, Gelson, orient. II. Título.

sistema de bibliotecas ufg

UFG

TERMO DE CIÊNCIA E DE AUTORIZAÇÃO PARA DISPONIBILIZAR AS TESES E DISSERTAÇÕES ELETRÔNICAS NA BIBLIOTECA DIGITAL DA UFG

Na qualidade de titular dos direitos de autor, autorizo a Universidade Federal de Goiás (UFG) a disponibilizar, gratuitamente, por meio da Biblioteca Digital de Teses e Dissertações (BDTD/UFG), regulamentada pela Resolução CEPEC nº 832/2007, sem ressarcimento dos direitos autorais, de acordo com a Lei nº 9610/98, o documento conforme permissões assinaladas abaixo, para fins de leitura, impressão e/ou *download*, a título de divulgação da produção científica brasileira, a partir desta data.

1. Identificação do material bibliográfico: **[X] Dissertação** **[ ] Tese**

2. Identificação da Tese ou Dissertação

Nome completo do autor: **Gilberto Antônio Marcon dos Santos**

Título do trabalho: **Novas Abordagens para Segmentação de Nuvens de Pontos Aplicadas à Robótica Autônoma e Reconstrução 3D**

3. Informações de acesso ao documento:

Concorda com a liberação total do documento [**X**] SIM [ ] NÃO[1]

Havendo concordância com a disponibilização eletrônica, torna-se imprescindível o envio do(s) arquivo(s) em formato digital PDF da tese ou dissertação.

Gilberto G M Santos

Assinatura do (a) autor (a) [2]

Data: 17 / 08 / 2016

[1] Neste caso o documento será embargado por até um ano a partir da data de defesa. A extensão deste prazo suscita justificativa junto à coordenação do curso. Os dados do documento não serão disponibilizados durante o período de embargo.

[2] A assinatura deve ser escaneada.

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DE GOIÁS
ESCOLA DE ENGENHARIA ELÉTRICA, MECÂNICA E DE COMPUTAÇÃO
COORDENAÇÃO DE PÓS-GRADUAÇÃO

Ata de Dissertação de Mestrado

Ata da sessão de julgamento da Dissertação de Mestrado em Engenharia Elétrica e de Computação, área de concentração Engenharia de Computação, do candidato **Gilberto Antonio Marcon dos Santos,** realizada em 12 de agosto de 2016.

Aos doze dias do mês de agosto de dois mil e dezesseis, às 14:00 horas, na Sala "Caryocar Brasiliensis" no bloco A da Escola de Engenharia Elétrica e de Computação (EMC), Universidade Federal de Goiás (UFG), reuniram-se os seguintes membros da Comissão Examinadora designada pela Coordenadoria do Programa de Pós-graduação em Engenharia Elétrica e de Computação: Os Doutores , Gelson da Cruz Junior – Orientador (EMC/UFG), Cassio Dener Noronha Vinhal – Coorientador (EMC/UFG), Carmelo José Albanez Bastos Filho – POLI/UPE, e Leonardo da Cunha Brito – EMC/UFG, para julgar a Dissertação de Mestrado de **Gilberto Antonio Marcon dos Santos,** intitulada **"Novas Abordagens para Segmentação de Nuvens de Pontos Aplicadas à Robótica Autônoma e Reconstrução 3D"**, apresentada pelo Candidato como parte dos requisitos necessários à obtenção do grau de Mestre, em conformidade com a regulamentação em vigor. O Professor Doutor Gelson da Cruz Junior, Presidente da Comissão, abriu a sessão e apresentou o candidato que discorreu sobre seu trabalho, após o que, foi arguido pelos membros da Comissão na seguinte ordem: Carmelo José Albanez Bastos Filho, Leonardo da Cunha Brito e Cassio Dener Noronha Vinhal. A parte pública da sessão foi então encerrada e a Comissão Examinadora reuniu-se em sessão reservada para deliberar. A Comissão julgou então que o Candidato, tendo demonstrado conhecimento suficiente, capacidade de sistematização e argumentação sobre o tema de sua Dissertação, foi considerado **aprovado** e deve satisfazer as exigências listadas na Folha de Modificação de Dissertação de Mestrado, em anexo a esta Ata, no prazo máximo de 60 dias, ficando o professor orientador responsável por atestar o cumprimento dessas exigências. Os membros da Comissão Examinadora descreveram as justificativas para tal avaliação em suas respectivas Folhas de Avaliação, anexas a esta Ata. Nada mais havendo a tratar, o presidente da Comissão declarou encerrada a sessão. Nos termos do Regulamento Geral dos Cursos de Pós-graduação desta Universidade, a presente Ata foi lavrada, lida e, julgada conforme, segue assinada pelos membros da Comissão supracitados e pelo Candidato. Goiânia, 12 de agosto de 2016.

Comissão Examinadora Designada:

Gélson da Cruz Júnior – Orientador (EMC/UFG) (Avaliação: APROVADO)

Carmelo José Albanez Bastos Filho – POLI/UPE (Avaliação: APROVADO)

Leonardo da Cunha Brito - EMC/UFG (Avaliação: Aprovado)

Cassio Dener Noronha Vinhal – EMC/UFG (Avaliação: Aprovado)

Candidato:

Gilberto Antonio Marcon dos Santos

Resumo

Métodos de sensoriamento de profundidade produzem nuvens de pontos que representam as superfícies vizinhas. Interpretar e extrair informações de nuvens de pontos é um campo estabelecido e repleto de desafios ainda não superados. Algoritmos de processamento de imagens clássicos não se aplicam ou têm de ser adaptados porque a estrutura organizada que se poderia supor em imagens bidimensionais não se faz presente. Este trabalho apresenta três contribuições ao campo de processamento e segmentação de nuvens de pontos. Tais contribuições são resultados da investigação realizada no Laboratório para Educação e Inovação em Automação – LEIA, com o fim de avançar os conhecimentos relacionados a aplicações de sensoriamento espacial para robótica autônoma. A primeira contribuição consiste de um novo algoritmo para extração de planos de nuvens de pontos, que se baseia em métodos evolutivos. Partindo do método proposto por Bazargani, Mateus e Loja (2015), esta contribuição consiste em utilizar estratégias evolucionárias no lugar de algoritmos genéticos, de forma a tornar o processo menos sensível aos parâmetros definidos pelo usuário. A segunda contribuição é um método para segmentação de piso e obstáculos em nuvens de pontos para navegação autônoma, que utiliza o algoritmo de extração de planos proposto. O uso de uma árvore quaternária para segmentação adaptativa de área permite classificar os pontos com elevada taxa de acerto de forma eficiente e com desempenho compatível com dispositivos embarcados de baixo custo. A terceira contribuição é uma variação do método de segmentação proposto que se faz mais robusta e tolerante a ruído através da agregação de um classificador neural. O uso do classificador neural no lugar da limiarização simples torna o processo menos sensível a ruídos e falhas nas nuvens de pontos, o tornando especialmente interessante para o processamento de nuvens de pontos obtidas por métodos de reconstrução estéreo de tempo real. Uma completa análise de sensibilidade, acurácia e eficiência é apresentada para cada algoritmo. A métrica de ângulo diedral (ângulo entre os planos detectados e os polígonos de referência que compartilham ao menos um ponto em comum) proposta por Bazargani, Mateus e Loja (2015) é utilizada para quantificar a acurácia do método de detecção de planos. A razão entre os pontos corretamente classificados e o número total de pontos é utilizada como métrica de acurácia para os métodos de segmentação de piso. Também são considerados os custos computacionais e o tempo de execução, comparados aos principais métodos estado-da-arte.

Palavras-chave: Segmentação de nuvens de pontos. Extração de piso. Extração de planos. Consenso de amostragem. Estratégias evolucionárias. Redes neurais artificiais. Visão estéreo. Robótica autônoma.

Abstract

Depth sensing methods yield point clouds that represent neighboring surfaces. Interpreting and extracting information from point clouds is an established field, full of yet unsolved challenges. Classic image processing algorithms are not applicable or must be adapted because the organized structure of 2D images is not available. This work presents three contribution to the field of point cloud processing and segmentation. These contributions are the results of investigations carried out at the Laboratory for Education and Innovation in Automation – LEIA, aiming to advance the knowledges related to applying spacial sensing to autonomous robotics. The first contribution consists of a new algorithm, based on evolutionary methods, for extracting planes from point clouds. Based on the method proposed by Bazargani, Mateus e Loja (2015), this contribution consists of adopting evolutionary strategies in place of genetic algorithms making the process less sensitive to user-defined parameters. The second contribution is a method for segmenting ground and obstacles from point clouds for autonomous navigation, that utilizes the proposed plane extraction algorithm. The use of a quadtree for adaptive area segmentation allows for classifying points with high accuracy efficiently and with a time performance compatible with low cost embedded devices. The third contribution is a variant of the proposed segmentation method that is more noise tolerant and robust by incorporating a neural classifier. The use of a neural classifier in place of simple thresholding makes the process less sensitive to point cloud noise and faults, making it specially interesting for processing point clouds obtained from real time stereo reconstruction methods. A through sensitivity, accuracy, and efficiency analysis is presented for each algorithm. The dihedral angle metric (angle between the detected plane and the reference polygons that share at least one point) proposed by Bazargani, Mateus e Loja (2015) is used to quantify the plane detection method accuracy. The ratio between the correctly classified points and the total number of points is utilized as an accuracy metric for the ground segmentation methods. Additionally, computing costs and execution times are considered and compared to the main state-of-the-art methods.

Keywords: Point cloud segmentation. Ground extraction. Plane extraction. Sample consensus. Evolutionary strategies. Artificial neural networks. Stereo vision. Autonomous robotics.

Lista de ilustrações

Lista de abreviaturas e siglas

ADALINE	Neurônio adaptativo linear, do inglês: *adaptive linear neuron*.
AE	Algoritmos evolucionários.
AG	Algoritmos genéticos.
ASIC	Circuito integrado específico de aplicação, do inglês: *application-specific integrated circuit*.
CE	Computação evolucionária.
EBP	Retro-propagação de erro, do inglês: *error backpropagation*.
EE	Estratégias evolucionárias.
FPGA	Arranjo de portas programáveis por campo, do inglês: *field-programmable gate array*.
GPU	Unidade de processamento gráfico, do inglês: *graphic processing unit*.
HL	Camada oculta, do inglês: *hidden layer*.
LUT	Tabela de referência, do inglês: *look-up table*.
MLP	Perceptron de múltiplas camadas, do inglês: *multi-layer perceptron*.
OL	Camada de saída, do inglês: *output layer*.
PE	Programação evolucionária.
PG	Programação genética.
RANSAC	Consenso de amostragem aleatória, do inglês, *random sample consensus*.
RNA	Redes neurais artificiais.
SBM	Correspondência estéreo de blocos, do inglês, *stereo block matching*.
SGBM	Correspondência estéreo semi-global de blocos, do inglês, *semi-global stereo block matching*.

Sumário

Introdução

Tecnologias de escaneamento de superfícies permitem a coleta de dados referentes à geometria tridimensional de superfícies vizinhas e geram a reconstrução digital detalhada de objetos e ambientes. Vários campos da ciência, engenharia, arquitetura, artes e entretenimentos têm se beneficiado das técnicas de escaneamento disponíveis atualmente. Entre os exemplos de aplicação, tem-se reconstrução de fósseis, modelagem de partes de maquinário legado, modelagem de velhas construções e sítios arqueológicos, preservação de esculturas antigas e criação de modelos virtuais para filmes e comerciais. Avanços na tecnologia tornaram os dispositivos de escaneamento gradualmente mais baratos ao longo das décadas recentes. A crescente disponibilidade de tais dispositivos vem acompanhada da demanda por algoritmos eficientes para o processamento dos dados que coletam. Esta crescente diversificação de escaneadores permite expandir os limites de várias áreas que dependem de sensoriamento espacial (MASUDA et al., 2015; GUO; ZHU; MORDOHAI, 2015; ABAYOWA; YILMAZ; HARDIE, 2015; OCHMANN et al., 2016).

Percepção de profundidade é uma fonte de sensoriamento poderosa para veículos autônomos e robôs. Escaneadores de detecção de profundidade por luz (do inglês, *light detection and ranging* – lidar) oferecem dados de superfície acurados por tempo de voo. Escaneadores lidar tridimensionais direcionaram a maior parte dos projetos recentes de veículos autônomos (LEVINSON et al., 2011). Entretanto, o tamanho, o consumo de energia e o custo dos dispositivos lidar ainda os tornam proibitivos para projetos de robótica de médio e pequeno porte. Diferente dos dispositivos lidar, as câmeras de tempo de voo não têm partes móveis e oferecem dados de profundidade a baixos custos. Porém, problemas de distorção, baixa resolução e limitações relacionadas a objetos em movimento e objetos com concavidades nas cenas ainda limitam seu uso (FOIX; ALENYA; TORRAS, 2011). Tecnologias de reconstrução por estereoscopia e luz estruturada (do inglês, *structured light*) têm aplicações demonstradas e são realizadas por dispositivos de baixo custo (WEBER; HäNSCH; HELLWICH, 2015). Sensores de luz estruturada oferecem informação de profundidade a custos baixos, mas seu desempenho é severamente degradado sob iluminação ambiente intensa, como ao ar livre (GUPTA; YIN; NAYAR, 2013). Fusão de dados a partir de diferentes dispositivos sensores também tem sido assunto de estudo (ZHANG; SINGH, 2015).

Em contrate a estas técnicas, a visão estéreo oferece uma abordagem passiva ao sensoriamento de profundidade. Usando luz ambiente, requer menos energia para operar e oferece informações de profundidade de alta resolução a baixos custos. A reconstrução estéreo densa requer mais poder de processamento que os demais métodos, mas algoritmos modernos já são capazes de executar em tempo real no hardware atual (MARK; GAVRILA,

2006). Plataformas embarcadas de processamento de alto desempenho tornaram possível a coleta de dados tridimensionais em tempo real a partir de pares de câmeras de baixo custo usando métodos de reconstrução estéreo (HUMENBERGER et al., 2010). Esta abordagem estima profundidade com base em disparidade estéreo, i.e., diferenças entre as imagens esquerda e direita. Limitações físicas do processo estabelecem os limites de acurácia. Usando o modelo de câmera estenopeica, o erro de profundidade oferecido por um par de câmeras é aproximado por

$$| \delta Z | \approx \frac{Z^2 \cdot \mu}{b \cdot f}. \tag{1}$$

onde Z representa a distância da câmera, μ representa o erro de disparidade, b representa a distância entre as câmeras e f representa a distância focal das câmeras (CHANG; CHATTERJEE, 1992). A incerteza δZ aumenta quadraticamente com a distância, degradando a acurácia rapidamente e, assim, limitando dramaticamente o alcance. Esta é a maior desvantagem da reconstrução estéreo como escaneador de profundidade. Entre outras desvantagens, tem-se a dependência de fatores ambientais, tais como a suavidade das superfícies, a iluminação ambiente e a presença de texturas repetitivas e oclusões. Entretanto, câmeras estéreo, se comparadas a sensores ativos, são mais baratas, leves, não têm partes móveis e consomem menos energia.

Os métodos de sensoriamento de profundidade produzem nuvens de pontos que representam as superfícies vizinhas. Interpretar e extrair informações de nuvens de pontos é um campo estabelecido. Extrair atributos e segmentar nuvens de pontos ainda é uma tarefa desafiadora. Algoritmos de processamento de imagens clássicos não se aplicam ou têm de ser adaptados. A estrutura organizada que se poderia supor em imagens bidimensionais não se faz presente. Nuvens de pontos gerais consistem de uma lista não ordenada de coordenadas xyz.

A maior parte dos estudos foca em processamento de nuvens de pontos de sensores lidar para navegação, localização e mapeamento. Algoritmos para segmentação e interpretação de nuvens de pontos de câmeras de tempo de voo e de luz estruturada também são comuns.

Interpretação, que comumente envolve segmentação e classificação, tem recebido muita atenção tanto das iniciativas de pesquisa em computação gráfica quanto em robótica. A comunidade de computação gráfica está geralmente interessada no campo para o desenvolvimento de métodos de compressão (BAUER et al., 2003; MERRY; MARAIS; GAIN, 2006), técnicas de renderização e visualização rápidas (WAHL; GUTHE; KLEIN, 2005) e de engenharia reversa de estruturas e partes mecânicas (CHEN; CHEN, 2008; SCHNABEL; WAHL; KLEIN, 2007). A literatura de robótica foca principalmente na interpretação de nuvens de pontos como fonte primária de sensoriamento para robótica móvel e veículos autônomos. Diversas tarefas, tais como planejamento de rotas, navegação,

localização mapeamento e reconhecimento de objetos se beneficiam das informações ricas oferecidas pelos dados de profundidade (XIAO et al., 2013). Atenção especial tem sido dada, nos anos recentes, à questão do sensoriamento de terreno para robôs terrestres autônomos, tanto *off-road* (LALONDE et al., 2006; KONOLIGE et al., 2009) quanto de propósito geral (MOOSMANN; PINK; STILLER, 2009; HIMMELSBACH; HUNDELSHAUSEN; WÜNSCHE, 2010; CHEN et al., 2014).

Abordagens baseadas em modelos, tais como a de Schnabel, Wahl e Klein (2007), aplicam modelos de Monte Carlo na tentativa de corresponder formas predefinidas às nuvens de pontos. Métodos baseados na transformada Hough têm gerado algoritmos com complexidade O(n log n), entretanto, requerem a calibração de vários parâmetros empiricamente (HULIK et al., 2014; LIMBERGER; OLIVEIRA, 2015). Métodos baseados em normais de superfície têm de lidar com o peso de processamento de cálculo dos vetores normais para cada um dos pontos da nuvem (MOOSMANN; PINK; STILLER, 2009; STROM; RICHARDSON; OLSON, 2010; SCHOENBERG; NATHAN; CAMPBELL, 2010). Propor e avaliar métodos de estimação de normais para nuvens de pontos ainda fazem parte de um contínuo esforço de pesquisa (JORDAN; MORDOHAI, 2014).

Para melhorar o desempenho de execução da interpretação de nuvens de pontos, a maior parte das abordagens recentes se beneficia da realização de discretização ou de redução de dimensionalidade por meio da projeção das nuvens tridimensionais em um espaço bidimensional. Algumas abordagens realizam projeção ortogonal sobre um plano horizontal xy, paralelo ao piso (HERNANDEZ; MARCOTEGUI, 2009; HIMMELSBACH; LUETTEL; WUENSCHE, 2009). Outras realizam projeção perspectiva para tornar a densidade de pontos aproximadamente uniforme sobre a superfície de projeção (HAMPP; BORMANN, 2013; BEWLEY; UPCROFT, 2013; PARK; CHOI; YU, 2014). Projeção e discretização permitem acelerar a segmentação, porém ao custo de perda de informação.

Métodos baseados em grafos permitem adaptar algoritmos de segmentação de imagens bidimensionais *bottom-up* para nuvens de pontos sem exigir qualquer projeção. Pontos se transformam em nós dos grafos e conexões entre pontos geram vértices. Algoritmos de crescimento de região que operam sobre grafos renderam resultados bem-sucedidos utilizando apenas dados geométricos (KLASING; WOLLHERR; BUSS, 2008; MOOSMANN; PINK; STILLER, 2009) e também dados geométricos combinados com dados de tonalidade (STROM; RICHARDSON; OLSON, 2010; SCHOENBERG; NATHAN; CAMPBELL, 2010). Entretanto, para alcançar desempenho em tempo real, as implementações contam com a ordenação dos dados dos sensores ou com a fusão de dados RGB de câmeras para a construção dos grafos. Para nuvens de pontos gerais, para as quais ordenação não é esperada, uma lenta busca de vizinhança seria necessária. A busca de vizinhança, um processo chave em vários métodos de segmentação bidimensional, requer estruturas complexas e métodos avançados para executar rapidamente em nuvens de pontos (BEHLEY;

STEINHAGE; CREMERS, 2015).

Nuvens de pontos obtidas por métodos *estéreo*, diferentemente das nuvens obtidas via lidar, estão sujeitas a vários problemas, tais como elevado ruído e dados esparsos. Assim, aplicar a nuvens de pontos estéreo métodos feitos para sensores ativos se torna frequentemente inviável. Alguns algoritmos para segmentação de nuvens de pontos estéreo também estão sujeitos a restrições adicionais, tais como operar apenas sobre nuvens de pontos organizadas ou sobre nuvens de pontos geradas por métodos *off-line*. A Figura 1 apresenta um exemplo de nuvem de pontos estéreo. Fotografia em tons de cinza da cena original apresentada em (a). Nuvem de pontos com intensidades luminosas apresentada em (b). Informações de profundidade apresentadas em (c): pontos em vermelho estão mais distantes. Em (d), piso marcado manualmente (em verde) para teste de acurácia e treinamento da MLP (Capítulo 7).

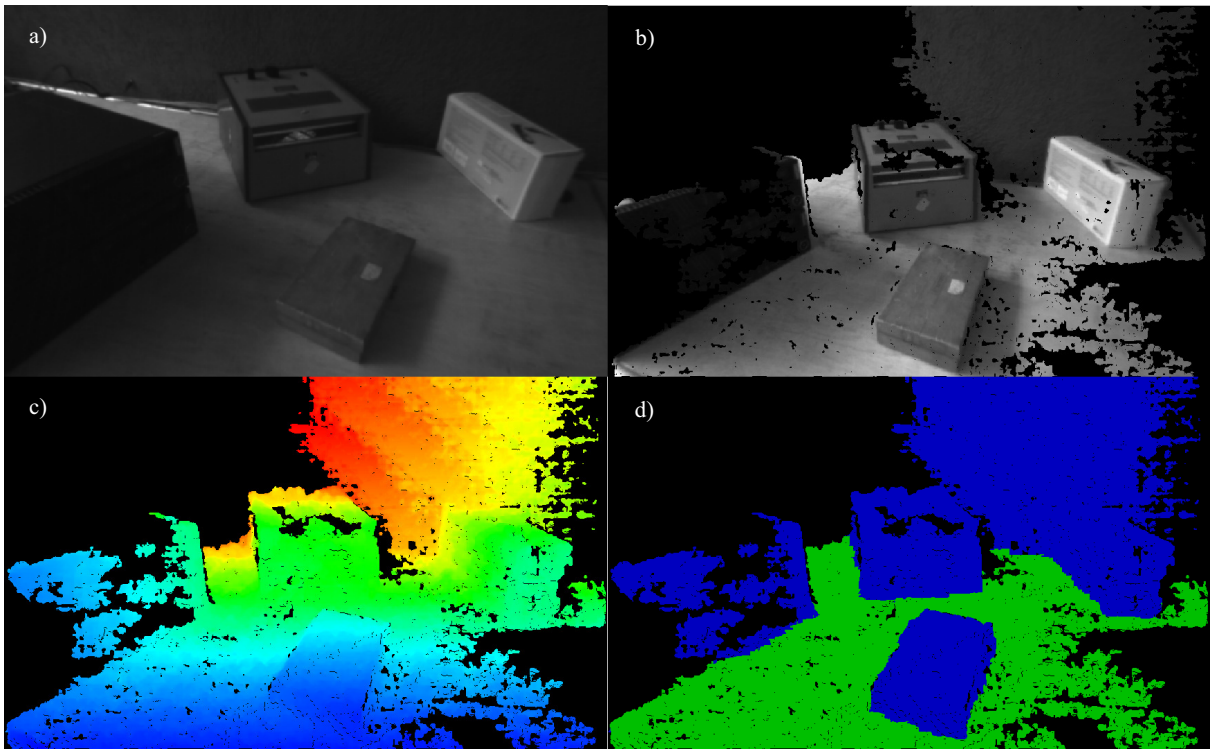

Figura 1 – Nuvem de pontos obtida por reconstrução estéreo em tempo real. Esta nuvem representa uma parede ao fundo e objetos espalhados pelo piso. (a) Fotografia em tons de cinza da cena original. (b) Nuvem de pontos com intensidades luminosas. (c) Informações de profundidade: pontos em vermelho estão mais distantes. (d) Piso marcado manualmente (em verde) para teste de acurácia e treinamento da MLP (Capítulo 7).

Este trabalho apresenta três contribuições ao campo de processamento e segmentação de nuvens de pontos. Tais contribuições resultam da investigação realizada no Laboratório para Educação e Inovação em Automação – LEIA. As três contribuições surgiram como ramificações da pesquisa que culminou no desenvolvimento de uma plataforma

para pesquisa em robótica móvel autônoma (MARCON; FERRAO, 2016). Como a visão e a interpretação de informações visuais têm um papel central na tomada de decisão de robôs móveis autônomos, vários problemas relacionados surgiram durante o desenvolvimento da plataforma. O estudo das soluções existentes na literatura e a necessidade de novas soluções resultou na proposição dos métodos apresentados no presente trabalho.

A primeira contribuição, apresentada pelo Capítulo 5, é um novo algoritmo para detecção e extração de formas planas de nuvens de pontos. A extração de formas planas de nuvens de pontos é um componente importante de vários métodos de interpretação. Assim, este método tem um escopo de aplicação muito mais amplo que apenas à robótica móvel. O algoritmo proposto se mostrou um método interessante para representação compacta de nuvens de pontos ricas em estruturas artificiais, tais como peças mecânicas e edificações.

A segunda contribuição, apresentada pelo Capítulo 6, detalha um novo algoritmo para extração de piso de nuvens de pontos não-organizadas obtidas por lidar, cuja aplicação principal é na detecção de obstáculos para navegação autônoma. Este algoritmo resulta da aplicação do algoritmo proposto no Capítulo 5 aliado à segmentação adaptativa usando árvores quaternárias. Esta abordagem foi publicada recentemente pelo autor (MARCON et al., 2016).

A terceira contribuição, apresentada pelo Capítulo 7, resulta da aplicação de uma rede MLP ao problema de extração de piso em nuvens de pontos. O algoritmo consiste de uma versão simplificada do publicado (MARCON et al., 2016), incrementada com um classificador MLP. O uso de uma rede MLP viabiliza sua simplificação, o torna mais robusto e permite a redução dos parâmetros definidos pelo usuário. Este método é mais apropriado para segmentar nuvens de pontos estéreo não-organizadas, geradas por reconstrução em tempo real.

Os algoritmos propostos neste trabalho se baseiam em métodos de inteligência computacional e conceitos clássico da computação, tais como abordagens de geometria computacional e de estruturas de dados. Uma breve introdução aos tópicos de redes neurais artificiais e de computação evolucionária é apresentada nos capítulos 1 e 2. A visão computacional estéreo é responsável pela obtenção das nuvens de pontos a partir de pares estéreo e uma breve introdução ao tópico é apresentada no Capítulo 3. Entretanto, seu papel neste trabalho tem uma função auxiliar, de prestar o serviço de obtenção de nuvens de pontos. O escopo do trabalho consiste na interpretação das nuvens de pontos, portanto são apresentados apenas os mínimos fundamentos de visão computacional necessários para a contextualização e a compreensão dos métodos de interpretação propostos.

Parte I

Fundamentação Bibliográfica

1 Redes neurais artificiais

As redes neurais artificiais (RNAs) são estruturas capazes de representar e aproximar funções com grande generalidade. Como tal, têm um diverso portfólio de aplicações para solução de problemas de classificação, análise de dados e controle de sistemas. A existência de métodos simples, efetivos e eficientes para treinamento de RNAs as fazem especialmente eficazes em gerar modelos contínuos a partir de amostras pontuais. O método proposto no Capítulo 7 tem seu grande diferencial derivado da aplicação de uma RNA para tornar mais robusta e geral a classificação de pontos de uma nuvem para extração de piso para navegação robótica. Este capítulo apresenta uma breve introdução ao tema de RNAs.

1.1 Biomimética

Antes de introduzir os temas de RNAs e algoritmos evolucionários, é importante destacar o conceito que define a origem de ambas teorias, a biomimética. Biomimética é o uso e a adaptação de processos e mecanismos encontrados na natureza para a construção de sistemas e obras humanas. Muitos processos biológicos podem ser vistos como o resultado de um longo processo de otimização. O estudo de tais processos é comprovadamente uma fonte extensiva de inspiração para a solução de variados problemas de engenharia.

Várias áreas da computação e da engenharia se baseiam em bio-inspiração. O desenho de estruturas aero e hidrodinâmicas, novos materiais e estruturas são aplicações típicas. Áreas de mais alto nível de abstração, tais como a engenharia de sistemas de controle e de sistemas de classificação e de tomada de decisão também têm se beneficiado. Dente estas, se pode citar o desenvolvimento de teorias tais como as relacionadas ao aprendizado de máquina ou à computação evolucionária (PASSINO, 2005).

1.2 Introdução a RNAs

Redes neurais artificiais são estruturas computacionais inspiradas em redes neurais biológicas. São um representante clássico das estruturas computacionais para aprendizado de máquina. Adequadas e comumente utilizadas para aprendizado supervisionado, estas estruturas também permitem aprendizado por reforço e outras técnicas indiretas de aprendizado. O aprendizado supervisionado, assim como o aprendizado humano, se dá por meio de exemplos. O sistema é alimentado com diversos exemplos e o aprendizado é alcançado quando este se torna capaz de gerar saídas válidas para entradas ainda não apresentadas (generalização).

De maneira geral, o aprendizado em RNAs ocorre da seguinte maneira: um conjunto de entradas e saídas correspondentes válidas é apresentado à RNA. Este conjunto, também chamado de conjunto de treinamento, é a base para o aprendizado e constitui o conjunto de exemplos válidos para o treinamento. Através de um algoritmo de aprendizagem, a rede vai se adaptar segundo os exemplos. Após a aprendizagem, espera-se que as relações inerentes entre as entradas e as saídas apresentadas tenham sido captadas pela rede. Por meio do processo de aprendizagem, a RNA vai de fato aproximar estas relações.

Em outras palavras, uma RNA é um aproximador de funções. Suponha que há um conjunto de entradas $X = \{x_1, x_2, \ldots, x_p\}, x \in \mathbb{R}^I$ e um conjunto de saídas correspondentes $D = \{d_1, d_2, \ldots, d_p\}, d \in \mathbb{R}^K$. Suponha também que há uma relação inerente entre X e D representada por $f(x) : \mathbb{R}^I \to \mathbb{R}^K$. Através do processo de aprendizagem, a rede neural vai aproximar a função $f(x)$. Assim, pode ser usada para prever valores de saída a partir de entradas arbitrárias. RNAs são comumente usadas como aproximadores universais de funções de múltiplas entradas e múltiplas saídas (KECMAN, 2001).

1.3 Origens: neurônios artificiais lineares

A teoria de aprendizado supervisionado de máquina por meio de estruturas inspiradas em redes neurais tem suas origens no início da década de 1960. Foi nesse período que surgiram, a partir de campos distintos, dois modelos bio-inspirados capazes de aproximar funções lineares quaisquer.

1.3.1 ADALINE

O primeiro, publicado em 1960, foi batizado de neurônio adaptativo linear (ADALINE, do inglês: *adaptive linear neuron*) (WIDROW; HOFF et al., 1960). Oriundo do campo de processamento de sinais, sua função inicial era de proporcionar um mecanismo para obtenção de filtros lineares para cancelamento de ruído. A rede, constituída de uma camada de nós de processamento cuja função de ativação é proporcional, permite modelar qualquer função (transformação) linear de $\mathbb{R}^N \to \mathbb{R}^M$. A Figura 2 apresenta o modelo original de uma ADALINE de N entradas e M saídas.

Note que o elemento $N + 1$ não é de fato uma entrada, mas sim um valor constante unitário. Este elemento é chamado de fator de viés, compensação ou limiar e permite modelar hiperplanos que não necessariamente passam pela origem do espaço em questão. Como exemplo, no espaço $\mathbb{R}^3$, um plano é a transformação $f(x) : \mathbb{R}^2 \to \mathbb{R}^1$ definido por

$$z = w_1 x + w_2 y + w_3. \tag{1.1}$$

Assim, o elemento w_3 é o peso correspondente à entrada constante, 1, enquanto w_1

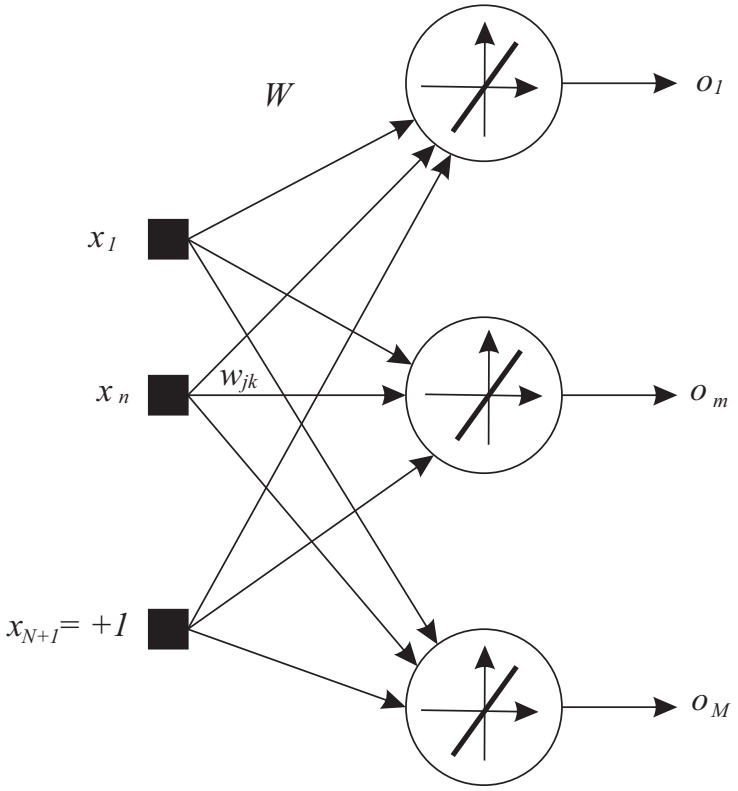

Figura 2 – Rede ADALINE.

e w_2 correspondem às entradas factuais, x e y. Sem a entrada constante e seu peso, w_3, a capacidade de representação estaria restrita a planos que passam pela origem do espaço. É esse fator constante que permite que a transformação efetuada por $W = \{w_1, w_2, w_3\}$ represente qualquer plano em $\mathbb{R}^3$.

Uma rede ADALINE também pode ser representada matricialmente:

$$Wx = o. \tag{1.2}$$

Observe que a relação representada pela Equação 1.2 é a representação geral para uma transformação linear. De fato, é do que se trata uma ADALINE. Ao solucionar sistemas lineares, porém, dispõe-se comumente dos elementos W e o e procura-se determinar x. No caso do treinamento em ADALINE, dispõe-se de um conjunto de $X = \{x_1, x_2, \ldots, x_p\}, x \in \mathbb{R}^N$ e um conjunto $D = \{d_1, d_2, \ldots, d_p\}, d \in \mathbb{R}^M$ e deseja-se encontrar o conjunto de pesos ideais W^*, que melhor aproxima a relação linear entre as entradas X e as saídas D, resolvendo

$$WX = D. \tag{1.3}$$

Como apresentado por Kecman (2001), W^* é obtido por

$$W^* = (XX^T)^{-1}XD = X^+D. \tag{1.4}$$

Onde X^+ é a pseudo-inversa de X. Este método permite a obtenção imediata dos pesos ótimos, porém é um processo monolítico. Uma abordagem mais gradual associada ao uso de redes ADALINE é o chamado método de descida gradiente, um processo de otimização comum, não apenas para problemas lineares. Aplicado à ADALINE, há garantia de convergência para W^*, dado que a otimização ocorre em uma superfície convexa.

Este método iterativo consiste então em ajustar continuamente os pesos segundo o erro δ – a diferença entre as saídas esperadas, d, e as saídas fornecidas pelo sistema, o. Por esse motivo, o método também é chamado de regra delta. O passo de ajuste é definido por um escalar arbitrário η, também chamado de taxa de aprendizado. Dessa forma, os pesos são obtidos iterativamente:

$$W_{i+1} = W_i + \eta(d_i - o_i)x_i, \tag{1.5}$$

$$W_{i+1} = W_i + \eta\delta x_i. \tag{1.6}$$

1.3.2 Perceptron

O segundo modelo, chamado de perceptron, foi publicado por Rosenblatt (1962). Voltado para a percepção visual, foi desenvolvido para tratar do reconhecimento de padrões e classificação. Perceptrons são capazes de classificar padrões quaisquer, desde que estes sejam linearmente separáveis. A questão da separabilidade linear é representada pela Figura 3. Enquanto uma rede baseada em perceptrons seria capaz de classificar os elementos dos grupos de (a) com total acurácia, o mesmo não pode ser esperado de (b). A reta que minimiza os erros de classificação é também apresentada na figura.

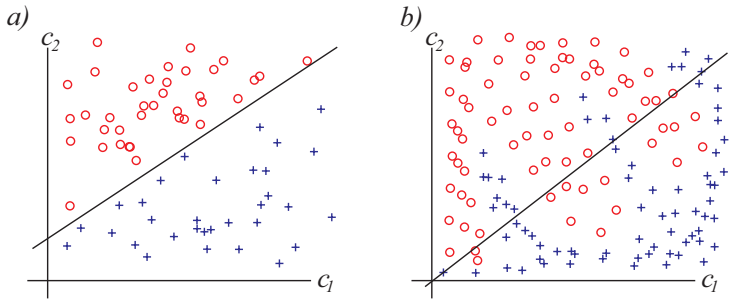

Figura 3 – Separabilidade de conjuntos. Os conjuntos em (a) são linearmente separáveis, enquanto os conjuntos em (b) não são.

O que ocorre, de fato, é que o processo de aprendizagem aplicado a uma rede linear de perceptrons é uma minimização dos erros, assim, espera-se que o sistema vai encontrar uma linha que melhor divide os dois grupos de modo a minimizar os erros de classificação. O fator que diferencia redes de perceptrons, como definidas originalmente por Rosenblatt e redes de ADALINE é o uso de funções de ativação *signum*. As funções *signum* se assemelham a uma função degrau, porém, são funções ímpares, i.e., $f(x) = -f(-x)$. São chamadas de *signum* porque para se gerar a saída leva-se em consideração apenas o sinal do valor de entrada:

$$signum(x) = \begin{cases} +1, \text{ se } x \geq 0 \\ -1, \text{ se } x < 0 \end{cases}. \tag{1.7}$$

De maneira semelhante à ADALINE, as saídas de um perceptron também podem ser representadas matricialmente:

$$o = signum(Wx). \tag{1.8}$$

Em sua publicação original, Rosenblatt mostrou que a convergência de aprendizado também ocorre para redes de perceptrons e definiu um teorema de convergência. Em resumo, o teorema afirma que, dado um conjunto de treinamento com entradas e saídas cujo padrão é linearmente separável, o processo de aprendizado vai necessariamente classificar os dados em tempo finito. Coincidentemente, este referido processo de aprendizado é exatamente o mesmo associado à ADALINE e definido pela Equação 1.5. A Figura 4 apresenta um modelo de rede clássica de perceptrons.

1.4 Perceptrons de múltiplas camadas

Enquanto as redes de ADALINE conseguem aproximar qualquer função linear e as redes clássicas de perceptrons conseguem classificar qualquer conjunto linearmente separável, ambas se limitam ao domínio linear. Por esse motivo, a área de aprendizagem supervisionada ficou relativamente estagnada durante as duas décadas que seguiram após sua origem.

O potencial de uso de redes de múltiplas camadas com funções de ativação não lineares para representação de funções quaisquer e classificação de grupos quaisquer, independente de linearidade, já era conhecido. Entretanto, não se dispunha de um algoritmo para efetuar o treinamento de maneira generalizada em trais estruturas. Foi apenas com a publicação do método de retro-propagação de erro (EBP, do inglês, *error backpropagation*) por Rumelhart, em seu artigo revolucionário de 1985, que o ramo voltou a se dinamizar e toda uma nova geração de investigação foi desencadeada (RUMELHART; HINTON; WILLIAMS, 1985).

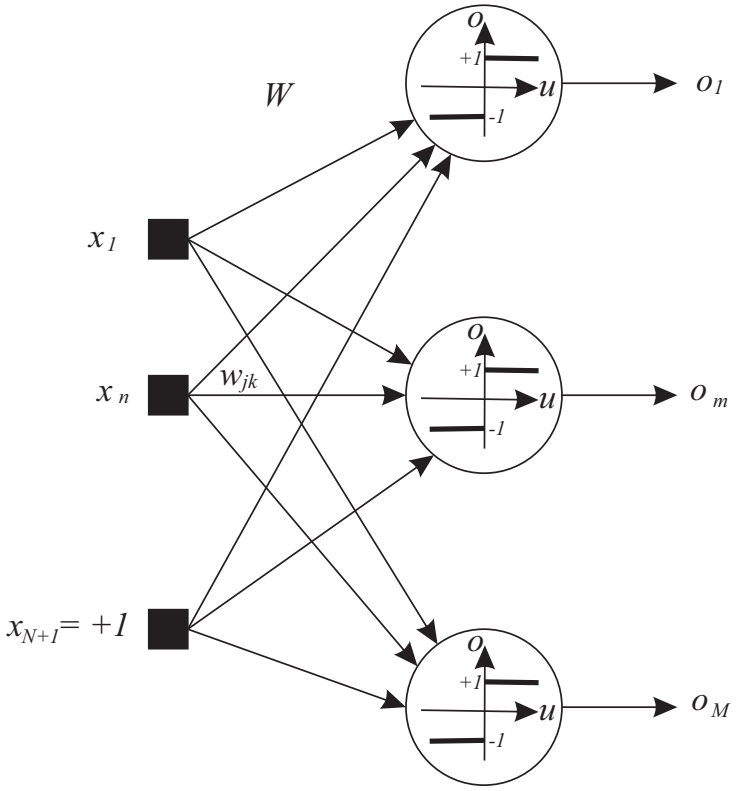

Figura 4 – Rede clássica de perceptrons.

O método de Rumelhart, também chamado de regra delta generalizada, posteriormente expandido em seu livro de 1988, permite o aprendizado em redes de perceptrons com múltiplas camadas e funções de ativação não lineares, desde que estas sejam diferenciáveis (RUMELHART; HINTON; WILLIAMS, 1988). Apesar de ter causado um grande avanço para a área de aprendizado de máquina na década de 1980, o método EBP já era há muito utilizado em outras áreas, como em controle ótimo, e já havia sido investigado por diversos outros pesquisadores (KECMAN, 2001). Este método é uma versão do método de descida gradiente apresentado pela Equação 1.5 e será detalhado nas seções a seguir.

1.4.1 Funções de ativação

Para que uma rede seja um aproximador global, é necessário, que ao menos uma camada da mesma seja constituída por neurônios cuja função de ativação seja não linear. Diversas funções de ativação foram estudadas desde a publicação do EBP. A classe mais comum é a das funções sigmoidais, caracterizadas por um formato que se assemelha à letra S e cuja derivada é positiva para todos os valores reais: $\frac{d}{dx}f(x) > 0, \forall x \in \mathbb{R}$. Como exemplo de função sigmoidal se pode citar a função logística unipolar:

$$f(u) = \frac{1}{1 + e^{-u}}. \tag{1.9}$$

Função logística bipolar:

$$f(u) = \frac{2}{1 + e^{-u}} - 1. \tag{1.10}$$

A tangente hiperbólica:

$$f(u) = tanh(u) = \frac{e^{u} - e^{-u}}{e^{u} + e^{-u}}. \tag{1.11}$$

A função de erro gaussiana:

$$f(u) = erf(u) = \frac{2}{\sqrt{\pi}} \int_{0}^{u} e^{-t^2 dt}. \tag{1.12}$$

Entre outras. Redes neurais de múltiplas camadas e funções de ativação sigmoidais são chamadas de perceptron de múltiplas camadas (MLP, do inglês, *multi-layer perceptron*). Um esquemático básico de MLP é apresentado pela Figura 5 e um exemplo de função sigmoidal é apresentado pela Figura 6 (a).

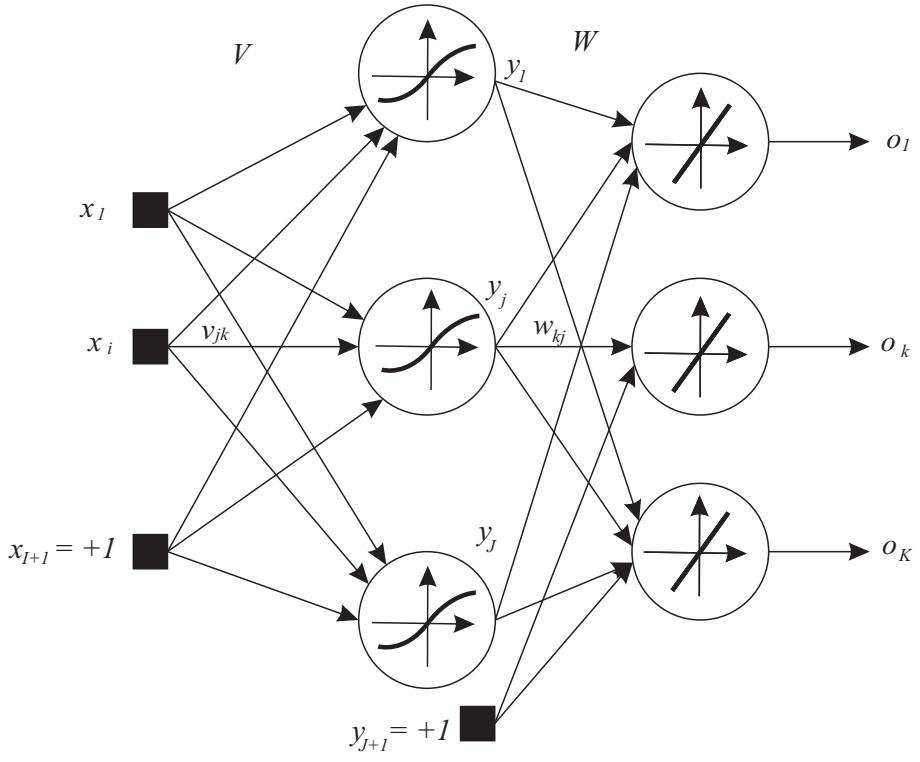

Figura 5 – Rede de perceptrons de múltiplas camadas.

Outra classe de funções de ativação muito importante é a classe das funções baseadas em raio (RBF, do inglês, *radial basis funciton*). Um exemplo de RBF é a função

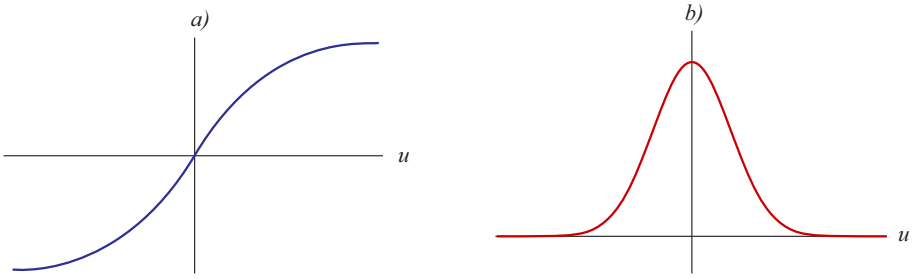

Figura 6 – Funções de ativação mais comuns e relevantes na área de RNAs. (a) Função de ativação sigmoidal. (b) Função de ativação gaussiana.

gaussiana:

$$f(u, \mu, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}. \tag{1.13}$$

onde μ representa a média e σ representa o desvio padrão da distribuição gaussiana.

Como exemplificado pela gaussiana da Figura 6 (b), as funções RBF são caracterizadas por curvas contínuas, suaves e simétricas. Redes neurais baseadas em funções RBF também são referenciadas pelo termo RBF. Assim como redes MLP, redes RBF também podem ser treinadas pelo método EBP. Adicionalmente, redes RBF se destacam por sua fundamentação teórica, segundo Kecman (2001), baseada na chamada teoria da regularização, que trata da aproximação de funções de múltiplas variáveis a partir de dados esparsos.

1.4.2 Topologia de rede

Em se tratando de redes ADALINE ou perceptron com funções de ativação lineares, cada camada da rede pode ser representada por uma transformação linear. Neste contexto, uma rede ADALINE de múltiplas camadas é chamada de MADALINE. Assim, uma rede MADALINE com três camadas lineares, A, B e C, com entradas x e saídas o pode ser representada como

$$ABCx = o. \tag{1.14}$$

Por essa notação, fica evidente que pode-se adotar uma matriz W capaz de representar qualquer transformação linear efetuada por ABC:

$$W = ABC, \tag{1.15}$$

$$Wx = o. \tag{1.16}$$

Portanto, o uso de múltiplas camadas de perceptrons cuja função de ativação é linear é uma redundância. Em se tratando de camadas não lineares, o assunto é mais controverso. Muita discussão no tópico ocorreu no final dos anos 1980 / início dos anos 1990, tanto de pesquisas que evidenciavam vantagens do uso de múltiplas camadas não lineares, quanto do oposto (CYBENKO, 1989), (HORNIK; STINCHCOMBE; WHITE, 1989), (KUURKOVA, 1992), (HUSH; HORNE, 1993), (FUNAHASHI, 1989), (CHESTER, 1990). Em suma, é matematicamente comprovado que tanto redes com apenas uma camada não linear quanto redes com múltiplas camadas não lineares funcionam como aproximadores universais. Assim, é comum o uso de redes com apenas uma camada não linear pelo princípio de se encontrar a solução mais simples e rápida para o problema em questão.

A rede MLP mostrada pela Figura 5 é um exemplo típico para aplicações de aproximação. Tem-se a camada interna, também chamada de camada oculta (HL, do inglês, *hidden layer*), composta por neurônios cuja função de ativação é não-linear e a camada externa, também chamada de camada de saída (OL, do inglês, *output layer*) composta por neurônios com funções de ativação linear. Para fins de classificação é mais comum o uso de redes MLP nas quais as funções da OL são funções degrau ou *signum*.

1.4.3 Quantidade de neurônios na camada interna

A questão referente ao número ideal de neurônios na HL, J, é um dilema importante na teoria de RNAs, também chamado de dilema da variância-viés (do inglês, *bias-variance dilemma*). Trata-se de um balanço entre obter algo mais próximo de uma aproximação ou algo mais próximo de uma interpolação.

É um fenômeno análogo ao que ocorre em interpolação polinomial. Quando a ordem do polinômio aproximador iguala a quantidade de pontos, obtém-se uma interpolação de fato, ou seja: a curva obtida passa necessariamente por cada um dos pontos tratados. Ao diminuir a ordem do polinômio, a curva resultante passa a ser uma curva de aproximação apenas, que não passa necessariamente pelos pontos.

O mesmo ocorre em redes neurais com funções de ativação não lineares. Em geral, ao elevar o número de elementos na HL, a curva resultante vai se aproximar de uma interpolação. Ao reduzir o número de elementos, a curva torna-se uma aproximação cada vez mais suave – e cada vez menos sujeita à influência de cada ponto individualmente. Em outras palavras, a redução faz com que a curva aproximadora passe a ignorar mais as características individuais de cada amostra. Isso a torna menos sujeita a ruídos – é como um filtro passa-baixas que elimina variações de alta frequência, gerando curvas mais suaves. Parte do sinal, contudo, também será ignorado juntamente com o ruído.

Quando se eleva excessivamente a ordem da HL, há uma tendência de se modelar o ruído pela aproximação. Em outras palavras, a curva está aproximando componentes

espúrios dos dados. Esse fenômeno acarreta na elevação da variância da solução: a cada vez que dados são coletados e aproximados, uma nova curva é obtida porque o ruído está sendo agregado à curva modelada. Quanto mais ruído modelado, mais variação é esperada entre uma iteração e outra. Quando a ordem da HL é excessivamente baixa, por outro lado, a variância é também tipicamente baixa, de forma que a mesma curva é obtida sempre a cada nova coleta de dados. Entretanto, há um viés (do inglês, *bias*), um deslocamento na curva modelada, de forma que esta difere da função inerente que relaciona os dados. Busca-se então um balanço entre variância e *bias* ao determinar a quantidade de neurônios da HL, como apresentado pela Figura 7, adaptada de Kecman (2001). Este balanço depende inerentemente dos dados modelados, a quantidade de entradas, de saídas, a largura de banda do sinal e relação sinal-ruído.

1.4.4 A taxa de aprendizagem e inicialização dos pesos

A taxa de aprendizado η, também chamada de largura de passo, é um escalar de grande importância na descida gradiente. Valores demasiado elevados para η podem levar à instabilidade do algoritmo, impedindo que o mesmo venha a convergir. Por outro lado, valores muito pequenos de η podem tornar o tempo de convergência excessivamente longo, ao ponto de fazer com que o algoritmo pareça não convergir após longo tempo de execução. Para diferentes aplicações, η pode diferir em várias ordens de magnitude, assim, uma regra básica ao encontrar problemas de convergência com o EBP é variar os valores de η.

Uma prática comum na inicialização do EBP é definir pesos aleatórios para W e V no intervalo $(-1, 1)$. Dependendo da aplicação, porém, outros intervalos podem favorecer uma convergência mais rápida.

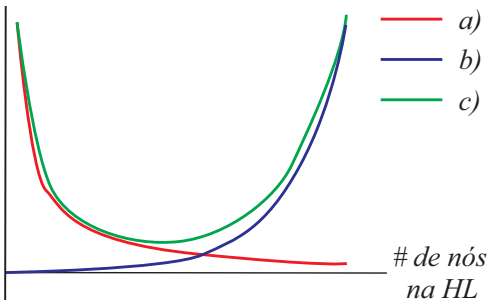

Figura 7 – Representação gráfica do dilema da variância-viés. (a) Viés ou *bias*. (b) Variância. (c) Soma das influências negativas de *bias* e variância. Busca-se encontrar um balanço entre os dois problemas. Adaptado de Kecman (2001).

1.4.5 O algoritmo Backpropagation

O princípio fundamental do EBP é de se determinar os erros nas saídas da OL e, a partir deste erro, determinar os erros referentes às saídas das camadas internas, propagando

o erro. Seguindo o princípio da Equação 1.5, os erros são usados para iterar os pesos de cada camada. No caso das redes MLP, como apresentado pela Figura 5, tem-se que o delta da OL é

$$\delta_o = d - o. \tag{1.17}$$

E que o delta da HL é

$$\delta_h = (1 - y)y\delta_o W. \tag{1.18}$$

Por fim, os pesos da camada interna (W) e da camada externa (V) devem ser iterados segundo:

$$W_{i+1} = W_i + \eta\delta_o y, \tag{1.19}$$

$$V_{i+1} = V_i + \eta\delta_h x. \tag{1.20}$$

O EBP trata a função de erro médio quadrático como uma função objetiva a ser minimizada, da mesma forma que o método de otimização gradiente, caminhando no sentido oposto ao gradiente da função objetiva. Assim, uma maneira de avaliar o progresso da aprendizagem é observar a função de erro quadrado definida por

$$E = \frac{1}{2}\delta_o^2. \tag{1.21}$$

No caso em que o número de saídas da OL é maior que um, $K > 1$, faz-se a soma dos erros quadrados de cada saída. O erro quadrado E pode ser utilizado como condição de parada, estabelecendo um valor de erro almejado E_a, porém, para comparações prefere-se estabelecer um limite de iterações e avaliar os resultados empiricamente. Cada par $\{x_p, d_p\}$ dos P pares dos conjuntos X e D são apresentados ao MLP, efetuando P iterações. A cada vez que todo o conjunto de treinamento é iterado, diz-se que se passou uma época (do inglês, *epoch*) de aprendizado. Comumente usam-se várias épocas, repetindo-se o ciclo várias vezes. Quanto mais épocas, melhor o aprendizado, porém mais longo o tempo de treinamento. Assim, procura-se experimentalmente encontrar o melhor balanço entre número de épocas e tempo de treinamento. O algoritmo 1 apresenta o processo EBP.

1.5 Aplicações de RNAs

Dada sua característica de atuar como aproximador universal, as RNAs encontram aplicações em diversos ramos científicos, da indústria e da saúde. Em geral, pode-se dividir

Dados:
Matriz de entradas $X_{P,I}$,
Matriz de saídas $D_{P,K}$,
número de nós HL J,
taxa de aprendizado η e
erro almejado E_a.
Resultado:
Matriz de pesos HL $V_{J,I+1}$ e
matriz de pesos OL $W_{K,J+1}$

Inicialização dos pesos W e V.
repeat
  $u = VX$;
  $Y = sigmoid(u)$;
  $O = WY$;
  $\delta_o = D - O$;
  $\delta_h = (1 - y)y\delta_o W$;
  $W_{i+1} = W_i + \eta\delta_o Y$;
  $V_{i+1} = V_i + \eta\delta_h X$;
  $E = \frac{1}{2}\delta_o^2$;
until $E > E_a$;

Algoritmo 1: Algoritmo EBP.

as aplicações de RNAs em diversas classes:

Aplicações de regressão incluem casos em que a RNA é utilizada para modelar e prever padrões em séries temporais, funções de aptidão em otimização ou qualquer outra aplicação que tenha regressão como elemento central.

Aplicações de classificação que se caracterizam por reconhecimento de padrões e sequencias.

Processamento de dados que envolvem filtragem, agrupamento e compressão.

Robótica, onde RNAs encontram as mais variadas aplicações, desde controle dinâmico, reconhecimento de padrões, até interface entre próteses e sistemas orgânicos.

Sistemas de controle, onde RNAs são aplicadas em controles adaptativos, cognitivos e inteligentes.

Dentro das categorias citadas pode-se mencionar controle de processos, veículos, gerenciamento de recursos e tomada de decisões, reconhecimento de alvos, reconhecimento de padrões em processamento de imagens, visão computacional, processamento de áudio e sinais diversos, reconhecimento de voz, texto, diagnósticos médicos, previsões e sistemas de negociação financeira, mineração de dados e filtragem de e-mail.

1.6 Desenvolvimentos recentes de RNAs e computação paralela

A partir do momento em que o algoritmo EBP foi publicado, grande esforço foi realizado no sentido de explorar o paralelismo característico das redes neurais. Diversas implementações em hardware foram estudadas e desenvolvidas. Os chamados neuro-computadores, surgidos nesse movimento, são uma proposta de sistemas de hardware voltado para a implementação paralela de redes neurais. Muito foi desenvolvido usando circuitos integrados específicos de aplicação (do inglês, *application-specific integrated circuit*, ASIC), porém os custos e a rigidez do ciclo de desenvolvimento de ASIC invariavelmente tornava os chips obsoletos antes mesmo de chegarem ao mercado – o desempenho de implementações de software em processadores atuais se mostrava um rival insuperável dos neuro-computadores em termos de desempenho. Em termos de flexibilidade, as implementações em hardware dificilmente apresentam vantagens sobre as de software (OMONDI; RAJAPAKSE, 2006).

Com o avanço e a popularização de sistemas de arranjo de portas programáveis por campo (do inglês, *field-programmable gate array*, FPGA), novas abordagens de RNAs em hardware passaram a ser estudadas nos anos 1990 e mais intensamente após 2000. A facilidade com que se pode reprogramar um sistema FPGA permite testar diferentes arquiteturas em um mesmo dispositivo a um ritmo muito semelhante ao de desenvolvimento de software. Adicionalmente, dispositivos FPGA recentes têm apresentado arquiteturas híbridas que contém centenas de nós aritméticos de ASIC distribuídos pela matriz. Com tais dispositivos pode-se utilizar a FPGA como uma rede programável e reconfigurável, explorando a flexibilidade de projeto de FPGA e ainda com desempenho comparável ao obtido em implementações ASIC (OMONDI; RAJAPAKSE, 2006).

Em geral, são três as principais dificuldades encontradas em implementações de redes neurais em FPGA: limitações de precisão numérica, representação de funções de ativação e limitação do tamanho do hardware. A questão da precisão numérica está associada à limitação do tamanho do hardware. Trabalhar com elevado número de bits para a representação numérica exige uma grande expansão no mesmo, que no contexto de RNAs, é uma expansão de ordem polinomial. Ao elevar o número de bits de precisão acarreta-se em uma elevação no número de unidades aritméticas de cada nó da rede e também na complexidade do roteamento entre nós (ZHU; SUTTON, 2003).

Para muitas aplicações, adicionalmente, o número de nós da rede inviabiliza uma implementação totalmente paralelizada, exigindo que uma mesma unidade aritmética seja usada para realizar o processamento referente a vários perceptrons. Essa circunstância exige um escalonamento e multiplexação por divisão de tempo, de tal forma que requer uma fila para ordenar o uso de cada unidade, elevando o *overhead* de controle. Muitas implementações estudadas buscam, assim, encontrar um balanço entre paralelismo de hardware e divisão de tempo. Poucas aplicações práticas apresentam redes pequenas o

suficiente para possibilitar implementações completamente paralelas. Além da limitação de processamento, outra limitação comum no hardware atual é a limitação de memória. Enquanto bancos de memória são abundantes e baratos atualmente, o acesso aos mesmo é serial, o que inviabiliza, muitas vezes, o acesso simultâneo de centenas de unidades aritméticas. Sistemas FPGA atuais também dispõem de pequenos blocos de memória RAM distribuídos pela matriz. Assim, como as unidades aritméticas, porém, sua quantidade é demasiado limitada para muitas aplicações (OMONDI; RAJAPAKSE, 2006).

A questão da representação das funções de ativação também é de grande relevância, uma vez que a convergência do EBP depende intrinsecamente das mesmas. Problemas relacionados à precisão da representação e do *overhead* gerado têm sido tratados em pesquisas recentes. Uma abordagem comum é o uso de tabelas de referência (LUT, do inglês *look-up table*) para a representação das funções, tanto alocadas em registradores *flip-flop* quanto em memória RAM. Gomperts, Ukil e Zurfluh (2010) propõe um paradigma de implementação paramétrico que visa explorar os recursos híbridos oferecidos em sistemas FPGA contemporâneos. Para tal, sugere uma representação de funções de ativação em LUT diferenciada, que permite uma acurácia superior às representações em LUT simples através de interpolação dinâmica sobre os elementos da tabela. Enquanto esta proposta permite uma economia dramática no uso e acesso à memória, causa um grande *overhead* de hardware. É digno de nota que sua implementação se restringe à representação numérica a ponto fixo, uma vez que o uso de ponto flutuante consumiria uma parcela maior de unidades aritméticas por nó.

Um caminho que tem sido explorado de forma intensa atualmente é o oferecido pelos sistemas de GPU. Beneficiada pelos investimentos massivos do dinâmico mercado de jogos eletrônicos, a tecnologia proporciona uma plataforma de pesquisa e desenvolvimento consolidada de processamento massivamente paralelo via software. A existência de interfaces de programação refinadas e estáveis e uma comunidade bem estabelecida favorece projetos na área. O uso de tais sistemas de computação paralela programáveis tem pavimentado o caminho para ramos de pesquisa em redes neurais de alta densidade e elevado número de unidades neurais. Aplicações voltadas ao processamento de imagens, visão computacional, classificação e reconhecimento de padrões são as áreas mais beneficiadas (CIRESAN et al., 2011). Muito tem se proposto com relação às chamadas redes neurais convolucionais, que são inspiradas em estruturas neurais biológicas cuja função é o reconhecimento de padrões (KRIZHEVSKY; SUTSKEVER; HINTON, 2012).

2 Computação evolucionária

A computação evolucionária é uma área da computação flexível inspirada na evolução de organismos vivos. Tendo como moto a "sobrevivência do mais forte"ou a "sobrevivência do mais bem adaptado", esta área explora a busca por soluções para problemas complexos através de processos aleatórios controlados. Em contrate aos métodos de busca estocástica de Monte Carlo, os métodos evolucionários visam restringir dinamicamente o espaço de busca para focalizar seu processo de otimização e minimizar seu custo computacional. Os Capítulos 5 e 6 apresentam métodos propostos baseados em estratégias evolucionárias como solução para problemas de interpretação de nuvens de pontos. Este capítulo introduz os conceitos fundamentais da computação evolucionária.

2.1 Definições

Atualmente há um consenso da comunidade científica sobre a definição de computação evolucionária, (CE). O termo se refere à adaptação de conceitos relacionados à biologia evolucionária para a implementação de sistemas evolutivos artificiais voltados à otimização. As origens da teoria que fundamenta a CE datam da primeira metade do século XX (JONG, 2006). Diversas famílias de abordagens surgiram desde então e esta ciência passou por vários processos de diferenciação e fusão ao longo das décadas que se seguiram. Este capítulo se propõe a apresentar uma breve síntese de seu desenvolvimento e as principais linhagens que surgiram durante este processo.

O primeiro ponto a ser notado em relação à CE é seu propósito. Seu objetivo é encontrar soluções com base em conceitos biológicos, mas não necessariamente seguir tais conceitos de maneira fiel. Em muitos casos, apenas uma pequena fração das características de um sistema biológico é modelada. Em geral, muito é ignorado dos sistemas evolutivos naturais pelo bem da simplicidade e previsibilidade de implementação. Resumidamente, enquanto o propósito da biologia evolucionária é estudar e modelar sistemas evolutivos naturais, o propósito da CE é obter inspiração em tais sistemas para a solução de problemas de computação e de engenharia (JONG, 2006).

Antes de tratar de detalhes históricos ou técnicos, é importante destacar os principais conceitos biológicos que fundamentam a CE. Em uma população, o indivíduo mais bem adaptado tem maiores chances de sobrevivência e de propagação de seu material genético. Neste contexto, **seleção natural** é o processo através do qual certas características tendem a sobressair em certas populações por garantir aos indivíduos maior adaptabilidade ao meio. **Reprodução por cruzamento** ocorre quando a reprodução envolve mais de um indivíduo parental e assim os descendentes incorporam uma combinação dos materiais genéticos dos

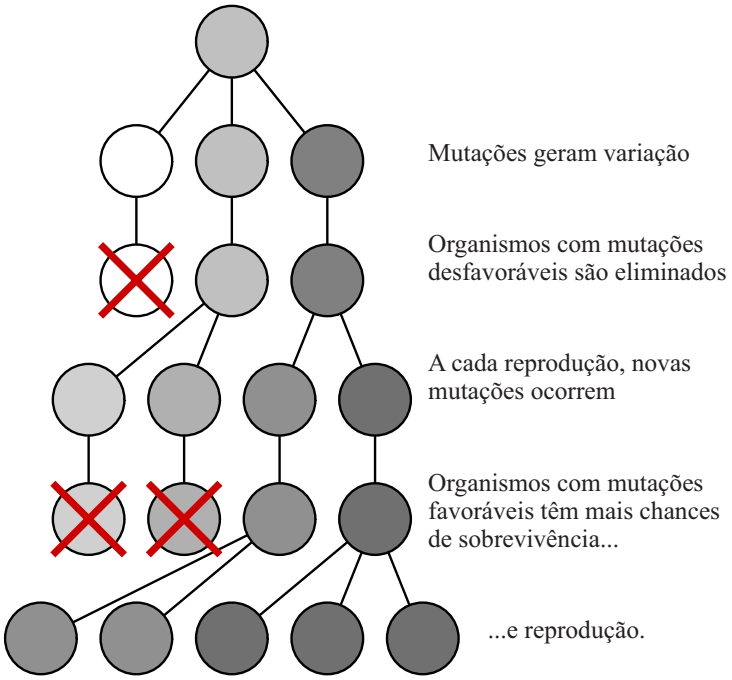

Figura 8 – Esquemático da noção básica de evolução.

pais. Este fenômeno tende a homogeneizar as características de uma população. **Mutação** é caracterizada pela distorção aleatória do conteúdo genético passado de ascendente para descendente. Seu papel é fundamental na variabilidade genética de uma população. Essa variabilidade estocástica é o fator necessário para que haja evolução em uma população. A **evolução** é a adaptação dinâmica de uma população ao meio em que esta se encontra através de processos de mutação e seleção natural. Além destes conceitos, as definições de **genótipo** e **fenótipo** são de grande importância. O primeiro se refere ao código genético de um indivíduo, enquanto o segundo diz respeito à manifestação de características observáveis de indivíduos como resultado da interação do genótipo com o ambiente ao qual está inserido. Por fim, o conceito de **aptidão** (do inglês ***fitness***) diz respeito ao grau de adequação de um indivíduo ao meio atual (PASSINO, 2005). A Figura 8 apresenta um esquemático do ciclo básico da evolução. Mutações geram variação e organismos com mutações desfavoráveis são eliminados. A cada reprodução, novas mutações ocorrem e organismos com mutações favoráveis têm mais chances de sobrevivência e reprodução.

Para esta análise é válido também destacar a definição de dois termos opostos usados em sistemas de otimização. **Busca intensiva** (do inglês ***exploitative search***) e **busca extensiva** (do inglês ***explorative search***). A primeira é mais mais lenta e localizada. Apesar de ser mais detalhada, tem mais chances de se prender a mínimos locais. A segunda é mais rápida e abrange áreas maiores em menor tempo. Apesar de ser menos susceptível a se prender a mínimos locais, tem maior tendência a ignorar pontos importantes do espaço de busca (JONG, 2006).

O termo **elitismo** é a última definição deste capítulo. Este termo é caracterizado

pelo favorecimento agressivo de genótipos imediatamente superiores. Algoritmos que eliminam extensivamente indivíduos menos aptos são ditos mais *elitistas* que aqueles que decidem mais brandamente sobre tais. Como exemplo de elitismo pode-se citar um sistema onde apenas o indivíduo mais apto sobrevive e gera descendência e sua prole sobrepõe todos os demais indivíduos. Em geral, métodos mais elitistas tendem a realizar busca mais **intensiva** (JONG, 2006). O elitismo é marcado por um ponto positivo e um ponto negativo:

- Preserva e garante a permanência mais duradora de genes aptos;
- Diminui a variabilidade e torna a busca mais susceptível a se prender a mínimos locais.

Nas Seções a seguir serão definidas as abordagens canônicas relacionadas à CE, juntamente com breve descrição histórica.

2.2 Algoritmos evolucionários

Algoritmos evolucionários (AE) compreendem uma série de métodos que envolvem o uso de conceitos biológicos e estocásticos para a otimização. Há várias analogias entre os conceitos biológicos de AE e os conceitos de otimização tradicionais. O espaço definido por todos os possíveis genótipos é análogo ao espaço de busca. O grau de aptidão de um determinado indivíduo é análogo à avaliação da função objetiva em um ponto do espaço de busca. Há máximos/mínimos locais e globais em curvas de aptidão assim como em funções objetivas.

Por se tratar de métodos estocásticos, os AE são particularmente úteis na otimização de sistemas não-lineares e que possuem múltiplos mínimos locais. Têm também uma natureza intrinsecamente paralela, avaliando a aptidão de múltiplas soluções em diversos pontos do universo de busca paralelamente. As origens destes algoritmos são diversas e cada uma delas é devida à busca pela solução de um problema distinto. Esta diversidade das origens fez com que cada abordagem explorasse certas particularidades dos AE.

A ideia central de AE é modelar uma solução para um problema como um indivíduo. Um conjunto de regras determinísticas/estocásticas é definido claramente para o ambiente onde se encontra o indivíduo. Em simulação, o ambiente é implementado e se cria uma população de tais indivíduos. Inicialmente atribui-se a cada indivíduo genótipos gerados de maneira aleatória segundo uma distribuição de probabilidades uniforme, não havendo nenhuma informação *a priori*. Segundo as regras do ambiente, certos indivíduos são selecionados para gerar novos indivíduos a partir de seu material genético de acordo com seus fenótipos. Da mesma forma, outros indivíduos são eliminados para dar lugar às novas

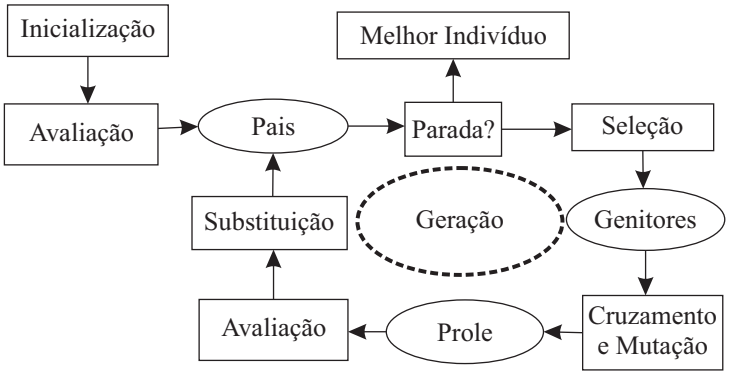

Figura 9 – Algoritmo geral de CE.

gerações. Dada a efetividade das regras de ambiente, espera-se que haja uma convergência da população em se adaptar ao ambiente. Pode-se assim, desenhar indiretamente o indivíduo ao definir o ambiente. A Figura 9 apresenta um rascunho do ciclo básico que rege o comportamento dos AE.

Três abordagens precursoras definiram o cenário de AE. Todas surgiram nos anos 1960 e desde então têm sido expandidas e aprimoradas. Algoritmos genéticos (AG), estratégias evolucionárias (EE) e Programação Evolucionária (PE) serão discutidas em maior detalhe nas seções a seguir. Uma quarta abordagem, que também tratamos aqui, difere tanto em origem quanto estruturalmente das demais. Se trata de programação genética (PG), que segue princípios semelhantes mas tem diferenças intrínsecas que a separa das demais e surgiu duas décadas mais tarde, nos anos 1980 (KOZA, 1992).

2.3 Algoritmos genéticos (AG)

Algoritmos Genéticos constituem uma das linhagens originárias de AE. Sua principal característica é o foco na codificação e decodificação de material genético. Esta característica distingue claramente genótipos e fenótipos. Um indivíduo é definido por um código que representa seu genótipo. Decodificando este código obtém-se o fenótipo, que constitui as características mensuráveis do indivíduo. Estas características podem então ser avaliadas através das regras do ambiente e assim sua aptidão é calculada.

Originalmente descritos por Holland (1975), AGs foram desenvolvidos para ser independentes de aplicação. Com esse intuito é que se diferenciou genótipos e fenótipos através de codificação. Tal codificação foi definida inicialmente como uma cadeia binária, mas abordagens posteriores expandiram-na para números reais e outros códigos (JONG, 2006).

Uma característica clássica dos AGs é a "geracionalidade". Em AGs clássicos, a cada geração a população é substituída por completo. A geração dos descendentes substitui a geração dos pais, de tal forma que não há competição direta entre gerações.

Outra característica típica das implementações pioneiras de AG é seleção proporcional à aptidão. Como que em uma roleta, a cada pai é atribuída uma probabilidade de produzir prole. Esta probabilidade é proporcional à sua aptidão. Ao gerar um novo descendente, o sistema determina estocasticamente seu pai "girando a roleta". Dessa forma, os pais que têm aptidão maior geram mais descendentes e os pais com aptidão menor geram menos descendentes. Mais especificamente, pais que estão acima da média gerarão, em média, mais que um descendente e aqueles que estão abaixo da média gerarão, em média, menos de um descendente (JONG, 2006).

2.4 Estratégias evolucionárias (EE)

Estratégias Evolucionárias surgiram com um caráter mais voltado à solução de problemas de otimização e foram inicialmente descritas por Schwefel (1975). Mais especificamente, seu foco era em otimizar funções a valores reais. Com relação à estrutura das populações e da seleção parental, é notável a notação (μ+λ), onde μ denota a população de pais e λ a população de filhos. Assim, em um sistema EE (μ+λ) com uma população de μ+λ indivíduos, selecionam-se os μ mais aptos como pais. Estes μ pais então geram λ filhos, que tomam o lugar dos λ menos aptos e assim sucessivamente. Um caso específico notável é (1+λ), onde apenas um indivíduo, o mais apto, de uma população é selecionado como pai e seus λ filhos substituem todo o restante da população atual. Variações da teoria de EE também incluem o caso de "geracionalidade", onde os pais são descartados ao gerar a prole e não são passados para as iterações seguintes.

Em EE, o indivíduo é modelado como uma estrutura com duas cadeias de valores. A primeira cadeia, s, contém os n argumentos da função objetiva. Esta cadeia representa o genótipo do indivíduo. A segunda cadeia, σ, contém n valores de desvio padrão. Estes valores definem o raio de atuação da mutação sobre cada elemento do genótipo. Assim, cada valor de σ diz respeito à mutação de um dos elementos de s.

Durante a geração de descendentes, a operação de mutação é efetuada sobre os elementos de s. Esta operação estocástica segue uma distribuição de probabilidade gaussiana de média zero e desvio padrão definido por σ. Assim, a cadeia σ define a variabilidade de mutação de cada um dos genes do indivíduo.

Tal estruturação foi elaborada para permitir que não apenas os genes mudem dinamicamente durante o processo de evolução, mas que também mude sua taxa de mutação. Ou seja, a cadeia σ, que define a taxa de mutação, também está sujeita à mudança e seleção como se fosse parte do genótipo. Espera-se assim fazer com que haja convergência mais rápida longe dos mínimos e mais lenta próximo destes.

Dada esta definição, sabe-se que σ define a mutação de s, mas o que define então a mutação de σ? A questão que esta estrutura gera é: com que taxa variar σ? Resultados

teóricos sobre a taxa de convergência ótima resultaram na regra de auto-adaptação de 1:5. Segundo Jong (2006) ela estabelece que:

Se mais de 20 % das mutações foram bem-sucedidas usando um dado σ, então o passo é demasiado ***intensivo****. Deve-se incrementar σ.*

Se menos de 20 % das mutações foram bem-sucedidas usando um dado σ, então o passo é demasiado ***explorativo****. Deve-se decrementar σ.*

As questões que permanecem são: (1) Em quanto se deve incrementar/decrementar σ? (2) Com que frequência deve-se realizar esta checagem? É notado que as respostas para tais perguntas podem depender muito fortemente da aplicação em questão. Outro ponto importante a ser notado é que fatores de "sucesso de mutação"devem ser considerados para cada dimensão da função objetiva. Em outras palavras, é necessário manter um registro para cada elemento do genoma para determinar o sucesso ou insucesso das mutações passadas. Esta análise de evolução da evolução é também chamada de **coevolução** (JONG, 2006).

2.5 Programação evolucionária (PE)

Descrita inicialmente por Fogel, Owens e Walsh (1966), a ideia inicial do algoritmo de Programação Evolucionária surgiu para a síntese automática de Máquinas de Estado Finitas (MEF). Em sua implementação a mutação assume o papel de acrescentar/subtrair arcos e estados e a recombinação de MEF em novas MEF. Dado uma população de tamanho N, a cada indivíduo é permitida a geração de um descendente, dobrando o tamanho da população. Os N mais aptos são então selecionados para gerar novos filhos e estes substituem os N menos aptos (JONG, 2006). Esta implementação é marcada assim, pela ausência de "geracionalidade".

2.6 Programação genética (PG)

Programação Genética é um ramo bastante distinto dos demais ramos de CE. Herda todos os conceitos fundamentais da biologia evolutiva para CE, porém tem seu foco em síntese automática de código. Sua estrutura é voltada para códigos onde dados e instruções são representados da mesma maneira, como em Lisp (STEELE, 1990).

Lisp é também a linguagem mais comumente adotada nos estudos de PG. Dado que a linguagem usa notação polonesa pré-fixada, seu código gera árvores gramaticais de maneira direta. A Figura 10 apresenta um exemplo simples de árvore gerada. A operação de mutação em PG, então, trata de acrescentar/remover nós e de modificar o conteúdo dos nós da árvore. A reprodução por cruzamento envolve, assim, a composição da árvore

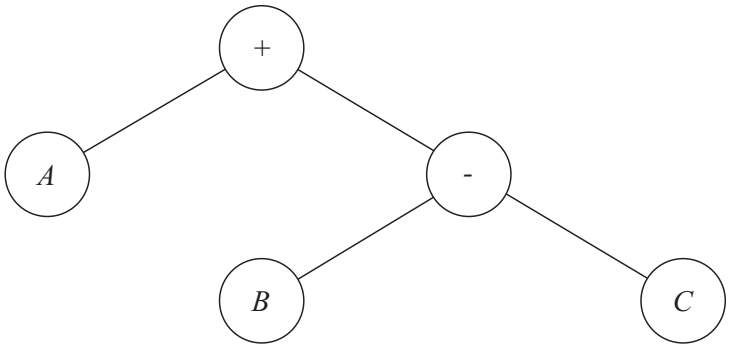

Figura 10 – Exemplo de árvore gramatical de Lisp.

descendente como a combinação de sub-árvores dos pais, dado que cada sub-árvore é por si só um programa válido (KOZA, 1992).

2.7 Diferenças entre AG, EE e PE

Apesar das diferentes origens e contextos em que surgiram AG, EE e PE, os três algoritmos têm muito em comum. Em todas as implementações iniciais foi abordado o cruzamento entre indivíduos na reprodução e a mutação teve um papel central na evolução. Em se tratando das diferenças de propósito, se pode ressaltar que, a princípio:

- AG visava criar uma estrutura de CE mais geral e independente de aplicação;
- EE visava tratar de otimização de funções a valores reais; e
- PE visava tratar da geração automática de máquinas de estado.

O nível de elitismo é outro fator diferenciador. Neste quesito, o AG clássico é tido como o menos elitista. Como não mantém os pais da geração anterior, o algoritmo força a variabilidade. Adicionalmente, o sistema de "roleta"e chances de procriação proporcionais à aptidão permitem que eventuais soluções não tão aptas procriem, garantindo a diversidade. Tudo isso garante uma maior robustez a mínimos locais, mas tende a dificultar o refinamento da solução quando a população já se encontra muito próxima a um mínimo absoluto. Abordagens EE e PE, por outro lado, tendem a ser mais elitistas.

Outro fator diferenciador importante é a codificação/decodificação característica de AG. Genótipos são decodificados (se manifestam em) fenótipos e vice-versa. Enquanto esta clara distinção entre os dois garante maior generalidade e menos dependência da implementação, a codificação traz uma série de dificuldades. A implementação inicial de AG usava codificação binária e as mutações consistiam em inverter certos bits do genoma aleatoriamente. Ao se inverter certos bits em um número binário pode-se acarretar variações muito grandes ou muito pequenas. O número 0b1101 (binário) = 0d13 (decimal),

por exemplo está a apenas um bit de distância de 0b1100 = 0d12. Este mesmo número, porém, também está a um bit de distância de 0b0101 = 0d05. Assim, para números binários com mais casas, uma inversão de bit pode ocasionar tanto em uma pequena quanto em uma grande mudança no valor resultante. Este fato tem vantagens e desvantagens. Ao passo que permite intrinsecamente que haja variabilidade no tamanho do passo da busca, faz com que essa variabilidade não seja controlável. Sistemas EE, por outro lado, têm grande enfoque no controle e otimização das taxas de mutação. Em algumas aplicações esta característica dos AGs pode inviabilizar a convergência para uma solução válida. Como resposta a isso, alterações sugeridas ao algoritmo original incluem o uso de código Gray ou o uso de codificação decimal. Outros problemas relacionados à codificação foram notados nas décadas que sucederam a publicação original de AGs e muitas soluções alternativas foram sugeridas (JONG, 2006).

Durante o desenvolver dessa ciência nas décadas finais do século XX, porém, houve uma integração dos grupos de pesquisa relacionados. Vantagens e desvantagens de cada abordagem foram trazidas à tona e confrontadas em conferências da área. Esta integração levou ao surgimento de cruzamentos e variações das teorias de AG, EE e PG, amadurecendo EC como ciência.

3 Visão computacional estéreo

A visão computacional estéreo permite extrair informação de profundidade a partir de pares de fotografias. Tal informação é comumente registrada na forma de uma lista de pontos, chamada de nuvem de pontos por sua semelhança a uma nuvem de partículas quando plotadas graficamente. Estas nuvens portam informações ricas sobre a geometria do espaço em que as fotografias foram registradas. Permitem o levantamento de mapas tridimensionais detalhados de ambientes, além da representação de objetos de formas e tamanhos variados para fins diversos. Os métodos apresentados neste trabalho representam contribuições ao processamento e à interpretação de nuvens de pontos, e por esse motivo, o presente capítulo apresenta uma breve introdução à visão computacional estéreo e noções básicas sobre como se obtêm nuvens de pontos a partir de pares estéreo.

3.1 Estereoscopia

A visão humana permite percepção de distância e profundidade. Tal percepção é possível devido à característica binocular de nossa visão. As imagens captadas, uma de cada olho, possuem diferenças que são causadas pela distância relativa entre os olhos, sendo cada imagem obtida de uma perspectiva ligeiramente diferente. O par de imagens é processado e fundido pelo cérebro em apenas uma imagem, que contém informações de profundidade, como ilustrado pela Figura 11. Este processo de obtenção de dados de profundidade a partir de imagens bidimensionais é chamado de estereoscopia. A estereoscopia também pode ser implementada computacionalmente para o mesmo efeito. Com base nas diferenças entre um par de imagens obtidas de posições ligeiramente distintas, obtêm-se informações de profundidade de um espaço tridimensional.

A estereoscopia é obtida computacionalmente combinando métodos de processamento de imagens e de reconhecimento de padrões e permite calcular posição, tamanho e velocidade de objetos no espaço tridimensional a partir de um par de imagens bidimensionais. Conhecendo as características intrínsecas de cada câmera (distância focal, formato do sensor e distorções da lente), é possível estimar com precisão as posições e as formas dos objetos no espaço.

O par estéreo consiste de duas imagens de uma mesma cena obtidas de posições distintas. Assim, um ponto específico deverá aparecer em coordenadas distintas no referencial de cada imagem. A distância absoluta entre cada ponto, ao sobrepor as imagens, é chamada de disparidade binocular do ponto, como ilustrado pela Figura 12.

A partir das características da câmera, é possível inferir a profundidade de um dado

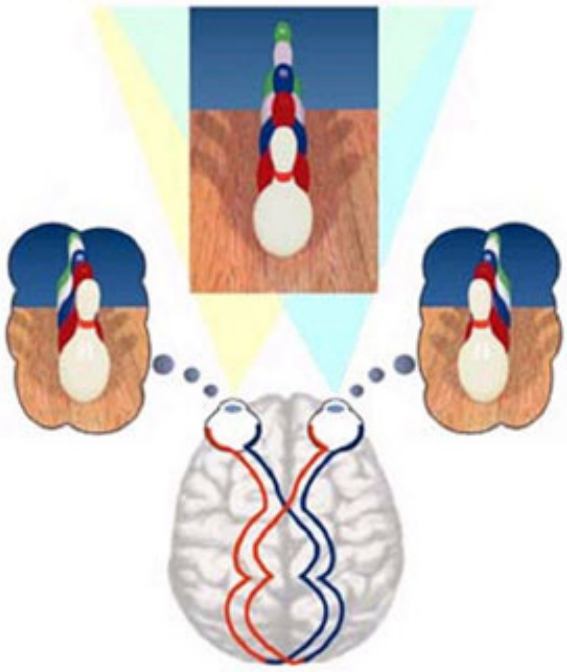

Figura 11 – Visão Binocular (MUNRO; GERDELAN, 2009)

ponto, dada sua disparidade. Para encontrar a disparidade entre dois pontos correspondentes, é necessário antes identificar tal correspondência, isto é, é necessário determinar quais pontos da imagem esquerda correspondem a quais pontos da imagem direita. Este processo de encontrar pontos correspondentes em pares estéreo é facilitado pela retificação das imagens, que as coloca no mesmo plano de projeção. O produto final da reconstrução estéreo é a nuvem de pontos, que consiste da lista dos pontos cuja posição no espaço tridimensional foi recuperada. As seções a seguir introduzem os conceitos fundamentais e detalham cada passo do procedimento para recuperação dos pontos tridimensionais.

3.2 Retificação

No caso geral, a busca de pontos correspondentes entre duas imagens de um par estéreo é uma busca bidimensional de elevado custo computacional. Quando se pode admitir alinhamento perfeito das câmeras, de forma a garantir a coplanaridade entre os planos de formação de imagem, a busca se torna unidimensional porque correspondências se restringem a pontos situados em linhas equivalentes. Em outras palavras, quando se tem câmeras idênticas perfeitamente alinhadas, um ponto que aparece na linha l da imagem esquerda aparecerá na linha l da imagem direita, tornando o escopo de busca por correspondências unidimensional. Como o alinhamento mecânico perfeito é dificilmente obtido e mantido, é comum a adoção do alinhamento virtual, via software.

Este alinhamento via software, chamado de retificação, se utiliza das características

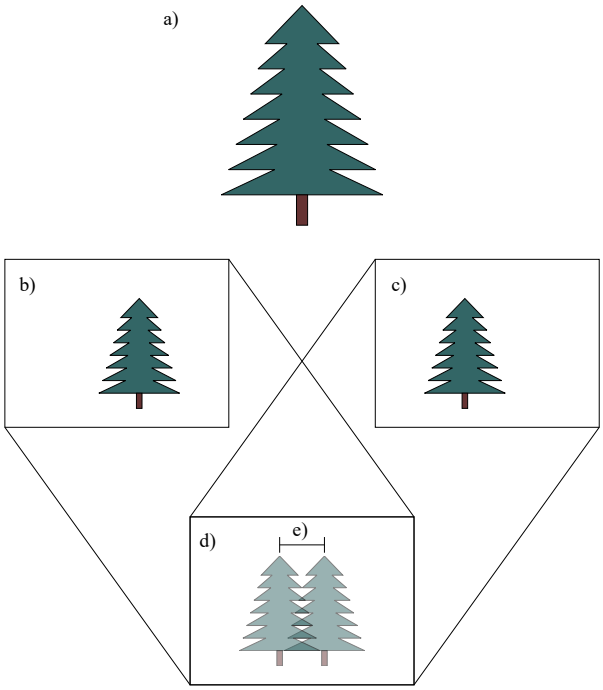

Figura 12 – Disparidade binocular. (a) Objeto tridimensional real. (b) Imagem bidimensional esquerda. (c) Imagem bidimensional direita. (d) Sobreposição das imagens esquerda e direita. (e) Disparidade binocular no ponto correspondente ao topo do pinheiro.

intrínsecas e extrínsecas do par estéreo. As características intrínsecas são comprimento focal f, formato do sensor (m_x e m_y são fatores de escala horizontal e vertical e γ representa uma distorção cisalhante) e ponto principal (u_0, v_0) de cada câmera. As características intrínsecas são representadas matricialmente pela matriz K:

$$K = \begin{bmatrix} f \cdot m_x & \gamma & u_0 \\ 0 & f \cdot m_y & v_0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{3.1}$$

As características extrínsecas derivam das diferenças de posição $\vec{T}$ e orientação R entre as câmeras, formando a matriz de coeficientes extrínsecos: $\begin{bmatrix} R & \vec{T} \end{bmatrix}$. O vetor $\vec{T}$ representa a posição da origem do sistema de coordenadas global expressa em coordenadas do sistema de coordenadas centralizado na câmera. R representa uma matriz de rotação, tal que $C = -R^T\vec{T}$, sendo C a posição do centro da câmera expressa em coordenadas globais (FUSIELLO; TRUCCO; VERRI, 2000). Com este modelo de câmera, a transformação projetiva que leva pontos do espaço tridimensional para o espaço bidimensional (do mundo real para a imagem) é dada, em coordenadas homogêneas, por:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = K \begin{bmatrix} R & \vec{T} \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}. \tag{3.2}$$

O processo de retificação tem interpretação geométrica demonstrável por meio da geometria epipolar, como ilustrado pela Figura 13. Duas câmeras distintas, cujos pontos focais são representados por c_1 e c_2, obtêm imagens de um ponto p do espaço tridimensional. A imagem de p aparece no ponto p_1 da imagem 1 e no ponto p_2 da imagem 2. O ponto p_1 é determinado pela intersecção entre o plano da imagem 1 e a reta que liga p a c_1. p_2 é determinado de maneira análoga.

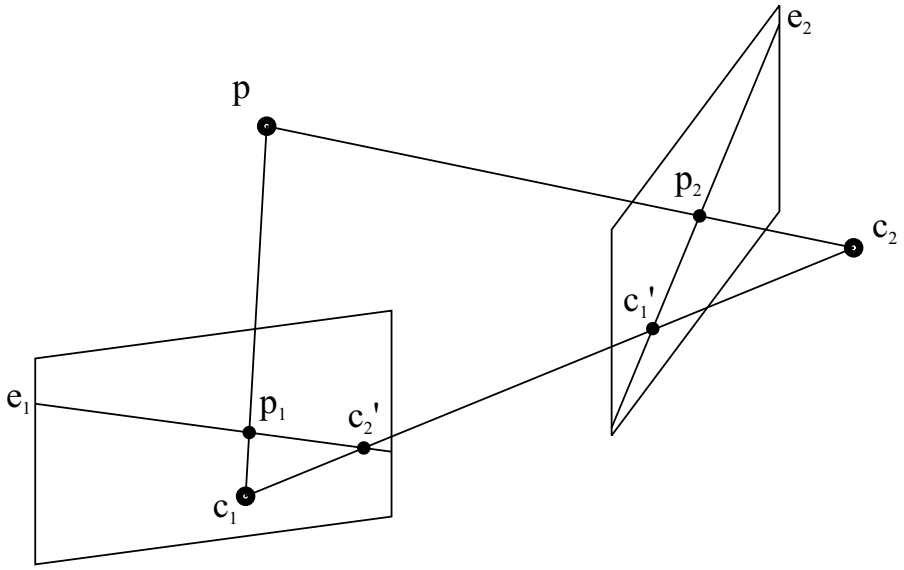

Figura 13 – Elementos fundamentais da geometria epipolar. Imagem 1 à esquerda e imagem 2 à direita.

A projeção do ponto c_2 sobre a imagem 1, determinada pela intersecção entre o plano da imagem 1 e a reta que liga c_2 a c_1, é chamada de epipolo da imagem 1. O epipolo da imagem 2 é determinado analogamente. A linha e_1, contida no plano da imagem 1, liga o epipolo da imagem 1 ao ponto p_1 e é chamada de linha epipolar. Há infinitas linhas epipolares e todas elas se encontram no epipolo. O processo de retificação leva os epipolos das duas imagens para o infinito e faz com que todas as linhas epipolares sejam horizontais e paralelas. Como consequência, um ponto arbitrário p no espaço tridimensional, que tem sua imagem no pixel p_1 da imagem 1, terá sua imagem no pixel p_2 da imagem 2, sendo que p_1 e p_2 se encontram em linhas equivalentes. Em outras palavras, por exemplo, se p_1 está na linha 137 da imagem 1, p_2 estará na linha 137 da imagem 2. Entretanto, devido ao ruído e à oclusão, esta asserção pode não ser exatamente válida, mesmo em pares estéreo de câmeras perfeitamente calibradas. Para imagens reais, a asserção é apenas aproximadamente válida, oferecendo uma boa estimativa de onde buscar por pontos correspondentes. Por isso, os métodos mais utilizados buscam também em áreas vizinhas à linha epipolar.

3.3 Obtenção do mapa de disparidades

Após a retificação, o próximo passo do processo de reconstrução tridimensional é determinar os pontos correspondentes entre as duas imagens. Este processo permite a criação do mapa de disparidades binoculares, que permite a estimação das profundidades. Esta seção apresenta os dois métodos mais comuns para determinação de correspondências e geração do mapa de disparidades.

3.3.1 Correspondência estéreo de blocos

A correspondência estéreo de blocos (SBM, do inglês, *stereo block matching*) é um algoritmo simples que utiliza da diferença dos níveis de intensidade entre duas imagens para obter a informação de distância. Neste método as imagens são percorridas por meio de uma janela N x N pixels, onde N é um número ímpar. Para cada janela de uma imagem, a outra imagem é percorrida por uma janela de mesmo tamanho sobre determinada linha epipolar. Os valores de intensidade dos pixels então são comparados por meio da soma das diferenças absolutas ou soma das diferenças quadradas entre as janelas. A menor diferença absoluta é escolhida, determinando o ponto da correspondência. A Figura 14 apresenta graficamente o processo de varredura das janelas sobre a linha epipolar.

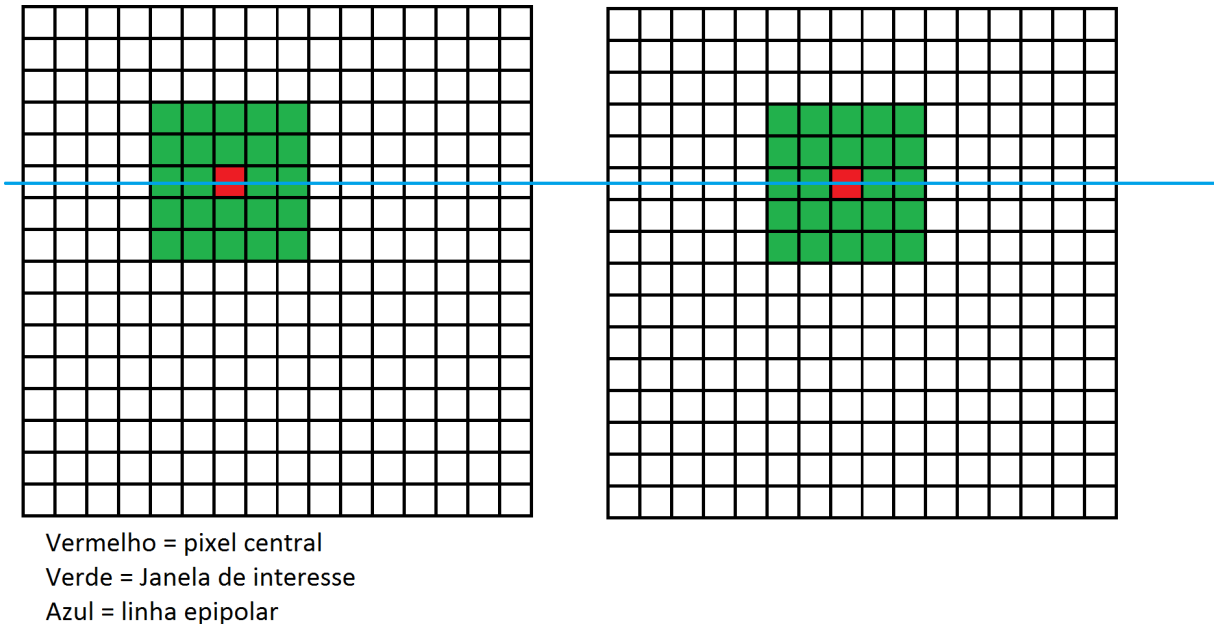

Figura 14 – Exemplo gráfico do algoritmo SBM.

3.3.2 Correspondência estéreo semi-global de blocos

O método da correspondência estéreo semi-global de blocos (SGBM, do inglês, *semi-global stereo matching*) é um processo mais complexo que utiliza de heurísticas para

gerar reconstruções mais densas e precisas. Em vez de utilizar apenas a linha epipolar, como no SBM, SGBM faz o cálculo da função custo do pixel por meio de vários caminhos, como mostrado pela Figura 15, para encontrar a melhor correspondência. A função custo para um pixel p e disparidade D_p é dada pela Equação 3.3, onde $h_{I_1}(p)$ é a entropia (HIRSCHMÜLLER, 2005) do pixel p da imagem esquerda, $H_{I_2}(D_p)$ é a entropia da imagem direita do pixel correspondente e H_{I_1,I_2} é a entropia das duas imagens sobrepostas do pixel correspondente.

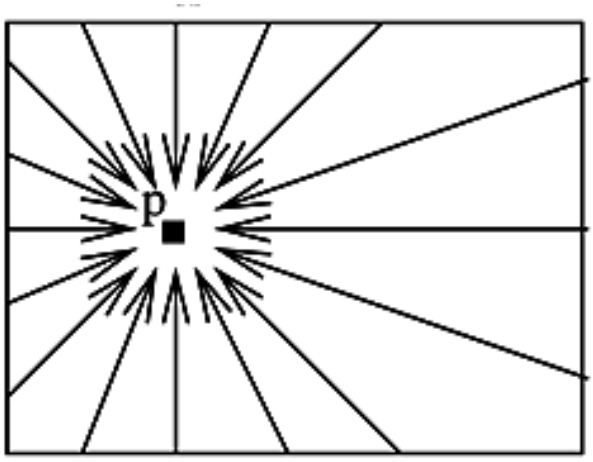

Figura 15 – Caminhos utilizados pelo SGBM.(HIRSCHMÜLLER, 2008)

$$-C(p, D_p) = h_{I_1}(p) + H_{I_2}(D_p) - H_{I_1,I_2}(p, D_p). \tag{3.3}$$

Para cada caminho, é calculado o custo para chegar a um pixel com uma certa disparidade. Para cada pixel e para cada disparidade, os custos são somados sobre todos os caminhos. Depois disso, para cada pixel, a disparidade com o menor custo é escolhida.

3.4 Obtenção da nuvem de pontos a partir do mapa de disparidades

Uma nuvem de pontos é um conjunto de pontos em um sistema de coordenadas euclidiano obtido a partir de reconstrução estéreo ou outras formas de sensoriamento tridimensional. Os pontos são comumente definidos em coordenadas cartesianas *xyz* e podem ser acompanhados de informações adicionais, tais como cor ou intensidade luminosa. A Figura 16 apresenta um exemplo de nuvem de pontos obtida via lidar.

Para construir a nuvem de pontos a partir de um mapa de disparidades, é necessário determinar a correspondência pixel-ponto, ou seja, cada pixel irá gerar um ponto na nuvem de pontos com a sua coordenada específica. Para conseguir essa correspondência, é utilizada a matriz de transformação de perspectiva, ou matriz Q. Esta matriz é obtida a partir dos

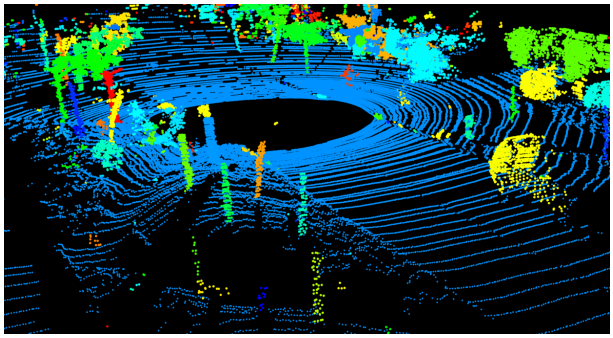

Figura 16 – Exemplo de nuvem de pontos (DOUILLARD et al., 2011).

parâmetros intrínsecos e extrínsecos da câmera e representa a transformação projetiva inversa, que leva pontos do espaço bidimensional para o espaço tridimensional.

$$Q = \begin{bmatrix} 1 & 0 & 0 & -u_0 \\ 0 & 1 & 0 & -v_0 \\ 0 & 0 & 0 & f \\ 0 & 0 & \frac{-1}{T_x} & \frac{u_0 - u_0'}{T_x} \end{bmatrix}. \tag{3.4}$$

As coordenadas do ponto principal da câmera esquerda são u_0 e v_0. A coordenada x do ponto principal da câmera direita é dada por u_0', f é a distância focal da câmera e T_x é a distância absoluta entre as duas câmeras no plano de projeção. A obtenção do ponto $P(x, y, z)$ se dá utilizando a seguinte equação

$$\begin{bmatrix} x' \\ y' \\ z' \\ w' \end{bmatrix} = Q \cdot \begin{bmatrix} u \\ v \\ disparidade(u, v) \\ 1 \end{bmatrix}, \tag{3.5}$$

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} \frac{x'}{w'} \\ \frac{y'}{w'} \\ \frac{z'}{w'} \end{bmatrix}. \tag{3.6}$$

Parte II

Abordagens propostas

4 Processamento de nuvens de pontos e métodos propostos

Nuvens de pontos organizadas, também chamadas de imagens de profundidade 2,5D, são imagens que representam profundidade, onde a intensidade de cada pixel indica a distância até a câmera. Nuvens de pontos organizadas permitem várias suposições geométricas que facilitam processamento e segmentação. A adjacência de pixels geralmente implica na adjacência de pontos tridimensionais, tornando a busca por vizinhos muito mais eficiente, por exemplo. Métodos recentes, tais como os métodos propostos por Holz et al. (2011) e Zhang, Wang e Yang (2010), alcançaram elevada acurácia e reduzidos tempos de processamento, mas podem operar apenas sobre nuvens de pontos organizadas.

Nuvens de pontos não-organizadas, por outro lado, são listas não ordenadas de pontos no espaço euclidiano e nenhuma informação de vizinhança está implícita. Nenhuma informação geométrica pode ser inferida da estrutura de dados e não se pode supor que as amostras são espaçadas ordenadamente. Os algoritmos para nuvens de pontos não-organizadas devem então operar sob condições muito mais desafiadoras.

Uma distinção clara também deve ser feita entre nuvens de pontos estéreo obtidas por métodos de tempo real e métodos *off-line*. Os primeiros oferecem dados geralmente mais densos e acurados que os segundos. Algoritmos de reconstrução de nuvens de pontos com desempenho de tempo real oferecem nuvens que são tipicamente mais esparsas, como a nuvem apresentada pela Figura 1, impondo mais restrições sobre os algoritmos de segmentação.

Várias abordagens de segmentação de nuvens de pontos bem estabelecidas são precisas e robustas, mas computacionalmente caras e lentas. Abordagens baseadas em modelos, que aplicam processos de Monte Carlo na tentativa de encontrar correspondências com formas predefinidas, requerem demasiado tempo de processamento para executar em tempo real (SCHNABEL; WAHL; KLEIN, 2007). Abordagens *top-down* dependem comumente de árvores *octree*, o que torna sua operação lenta (BEHLEY; STEINHAGE; CREMERS, 2015).

Algoritmos estado-da-arte rápidos se beneficiam de suposições específicas da aplicação ou de fusão de dados. Outros requerem dados muito precisos, tais como de lidar, ou dependem de ordenação de dados. Abordagens *bottom-up* são comumente baseadas em grafos: os pontos geram nós e relações entre os pontos geram as arestas dos grafos. Similaridades entre os atributos dos nós são representadas pelos pesos das arestas e usam-se algoritmos de crescimento de região baseados nos pesos. Como exemplo de métodos

bottom-up rápidos baseados em grafos que dependem apenas de informações geométricas tem-se Klasing, Wollherr e Buss (2008) e Moosmann, Pink e Stiller (2009), enquanto Strom, Richardson e Olson (2010) e Schoenberg, Nathan e Campbell (2010) usam tanto dados de cor quanto dados geométricos. Para a construção destes grafos em tempo real, as implementações usam ordenações de dados específicas do sensor ou fusão de dados de câmeras RGB. Para nuvens de pontos não-organizadas gerais, a lenta busca de vizinhança seria necessária, tornado a abordagem demasiada lenta para operar em tempo real.

Algoritmos clássicos de processamento de imagens são maduros e apresentam tempo de execução eficiente, mas requerem redução de dimensionalidade e discretização para se aplicar a nuvens de pontos. Uma estratégia comum envolve realizar projeção ortogonal sobre o plano horizontal xy e discretizar os dados da nuvem lidar sobre grades de ocupação, tal como realizado por Hernandez e Marcotegui (2009) e Himmelsbach, Luettel e Wuensche (2009). Oniga e Nedevschi (2010) propõe um método para detecção de obstáculos em estradas pavimentadas a partir de nuvens de pontos estéreo com desempenho de tempo real. Este método consiste em discretizar o plano horizontal em uma grade e então julgar os nós com base na densidade e altura dos pontos. Entretanto, o método depende de várias suposições, tais como a pose do veículo, a posição da câmera, a densidade dos dados e o formato da estrada; assim, o mesmo não é adequado à navegação robótica em ambientes heterogêneos.

Outras abordagens realizam projeção perspectiva ou cilíndrica para obter densidade de pontos uniforme sobre a superfície de projeção como em uma imagem (BEWLEY; UPCROFT, 2013) e em seguida aplicam técnicas clássicas de segmentação de imagens *bottom-up* ou *top-down*. O uso de métodos de segmentação *bottom-up* sobre estas superfícies, tal como crescimento de região, ainda é restrito a nuvens de pontos obtidas por métodos de reconstrução estéreo *off-line* devido às numerosas descontinuidades encontradas em nuvens de pontos obtidas por métodos de reconstrução estéreo em tempo real. Abordagens de segmentação *top-down*, tais como aquelas que usam árvores quaternárias (HAMPP; BORMANN, 2013; PARK; CHOI; YU, 2014; MARCON et al., 2015), produziram resultados bem-sucedidos. Árvores quaternárias são mais simples que árvores *octree*, permitem execução mais rápida e também permitem segmentação *top-down*. O Capítulo 6 propõe um novo método, que usa árvores quaternárias e projeção ortogonal sobre o plano horizontal xy, recentemente publicado pelo autor (MARCON et al., 2016), para a segmentação de piso e detecção de obstáculos em nuvens de pontos não-organizadas obtidas via lidar. O Capítulo 7 propõe uma versão mais robusta deste método, que é melhorado e simplificado pela adoção de uma rede MLP e sua aplicação a nuvens de pontos obtidas por métodos estéreo de tempo real é demonstrada.

Detecção e extração de formas planas de nuvens de pontos exercem um papel central em uma variedade de abordagens de engenharia reversa e visão para robótica. Planos

são as estruturas geométrica tridimensionais mais simples, onipresentes em ambientes artificiais. Este atributo os torna apropriados para interpretação de cenas, permitindo implementações simples e ampla capacidade de representação. Bauer et al. (2003), Chen e Chen (2008) e Bazargani, Mateus e Loja (2015) aplicam reconhecimento de planos para criar modelos compactos de construções. Kida et al. (2004) utiliza extração de planos para remover pontos de piso e então estimar atributos humanos, tais como altura e comprimento dos membros. Vários outros usam detecção de planos para localização e mapeamento em robótica (GUTMANN; FUKUCHI; FUJITA, 2008; RUSU et al., 2009; TREVOR et al., 2012; XIAO et al., 2013; HORNUNG et al., 2014). O Capítulo 5 propõe um novo método para a segmentação de planos em nuvens de pontos, que é utilizado pelo método de extração de piso publicado pelo autor (MARCON et al., 2016).

5 Detecção e extração de formas planas em nuvens de pontos

Durante o desenvolvimento da plataforma de pesquisa em robótica móvel proposta pelo autor (MARCON; FERRAO, 2016), notou-se que para garantir autonomia suficiente para um robô móvel se localizar, mapear ambientes e reconhecer objetos sem a interferência humana são necessários sensores de profundidade. Levantamentos da literatura recente mostram que várias tecnologias atuais de sensoriamento tridimensional produzem nuvens de pontos representadas por dados geralmente esparsos e de organização irregular. O primeiro desenvolvimento relacionado ao sensoriamento espacial realizado no projeto foi a proposta de um método para detecção e extração de formas planas em nuvens de pontos. Devido à sua generalidade e presença frequente nos mais variados ambientes e objetos, as formas planas detectadas são comumente usadas na detecção de objetos, mapeamento de ambientes e compressão de nuvens de pontos.

O método desenvolvido e proposto para esse fim é apresentado neste capítulo e sua aplicação vai além dos fins para o qual foi inicialmente desenvolvido, a navegação para robótica móvel. De fato, mostrou-se especialmente interessante para o reconhecimento e a representação de peças mecânica e estruturas a partir dos planos que as compõem. Isso ocorre porque o método em questão apresenta maior eficiência computacional quando os requerimentos de acurácia são elevados e é aceitável o processamento *off-line* das nuvens. Após uma breve contextualização e histórico sobre algoritmos para reconhecimento de planos em nuvens de pontos, o algoritmo proposto é apresentado e avaliado.

5.1 Métodos para extração de planos em nuvens de pontos

A abordagem mais simples e amplamente usada para reconhecimento de planos em nuvens de pontos é o método de consenso de amostragem aleatória (do inglês, *random sample consensus* – RANSAC) introduzido por Fischler e Bolles (1981) como um algoritmo de regressão robusta. O RANSAC tem sido aplicado com sucesso à extração de planos tanto puro (WAHL; GUTHE; KLEIN, 2005; SCHNABEL; WAHL; KLEIN, 2007; YANG; FÖRSTNER, 2010) quanto integrado a diferentes *frameworks* de otimização (GOTARDO et al., 2004; BAZARGANI; MATEUS; LOJA, 2015). Neste capítulo é apresentada uma nova abordagem para segmentação de planos que adota o consenso de amostragem com estratégias evolucionárias (do inglês, *evolution strategy sample consensus* – ESSAC) (TODA; KUBOTA, 2014) no lugar do RANSAC. Ao aprimorar o RANSAC clássico com paradigmas de estratégias evolucionárias, ESSAC provê uma solução mais geral, garantido os benefícios

de acurácia de outras abordagens de otimização, porém menos dependente de parâmetros e limiares definidos pelo usuário. Uma análise detalhada de desempenho é apresentada e a acurácia é quantitativamente avaliada usando a métrica proposta por Bazargani, Mateus e Loja (2015).

Como enfatizado por Xiao et al. (2013), a maior parte das metodologias de extração de planos encontradas na literatura são variações da transformada Hough, de RANSAC ou de crescimento de região. Diferentes metodologias, tais como o método determinístico apresentado por Sabov e Krüger (2008) e o algoritmo não-iterativo de Suganthan, Coleman e Scotney (2008) também existem, ainda que com grandes limitações. Abordagens de crescimento de região dependem de busca de vizinhança, e, portanto, são ou restritos a dados de profundidade organizados, ou carecem de eficiência aceitável. Um trabalho recente que contorna este problema foi proposto por Xiao et al. (2013), fazendo um crescimento de região baseado em árvores *octree* (do inglês, *octree*) com cache. Entretanto, o mesmo apresenta limitações de escalabilidade e se provou mais lento que outros algoritmos por meio de testes comparativos empíricos.

Métodos baseados na transformada Hough estão emergindo como o principal rival para o RANSAC devido a seu comportamento determinístico. Trabalhos passados demonstraram sua aplicabilidade (DUBE; ZELL, 2011; BORRMANN et al., 2011) e um estudo recente de Limberger e Oliveira (2015) propôs um algoritmo caracterizado por uma complexidade O(n log n) bem definida. Entretanto, este método exige a definição de vários parâmetros empiricamente, o que impõe que muitas suposições sejam feitas *a priori* sobre as nuvens de pontos.

5.1.1 Abordagens baseadas em RANSAC

A tarefa de estimar formas planas a partir de nuvens de pontos recebeu atenção considerável nos últimos anos. Uma forma comum de abordar este problema é usando o RANSAC clássico (WAHL; GUTHE; KLEIN, 2005; SCHNABEL; WAHL; KLEIN, 2007; YANG; FÖRSTNER, 2010). RANSAC também foi aplicado com sucesso para a extração de planos integrado a diferentes *frameworks* de otimização (GOTARDO et al., 2004; BAZARGANI; MATEUS; LOJA, 2015). Para a abordagem apresentada neste capítulo, a adoção dos paradigmas de estratégias evolucionárias com o *framework* de consenso de amostragem é proposta, semelhante à abordagem apresentada por Toda e Kubota (2014).

A equação geral de um plano é

$$ax + by + cz + d = 0. \tag{5.1}$$

Onde ao menos um dos parâmetros a, b e c deve ser não nulo. Supondo que um dado ponto $\vec{q} = [\ x \ \ y \ \ z\ ]$ no espaço euclidiano pertence ao plano definido pelo vetor de coeficientes

da Equação 5.1, $\vec{p} = [\ a \quad b \quad c \quad d\ ]$, tem-se, em notação vetorial, que

$$\left[\ \ a \quad b \quad c \quad d\ \right] \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = 0. \tag{5.2}$$

$$\vec{p} \cdot \begin{bmatrix} \vec{q} \\ 1 \end{bmatrix} = 0. \tag{5.3}$$

A distância euclidiana d_i entre qualquer ponto $\vec{q_i}$ e um plano $\vec{p}$ é dada por

$$d_i = \left| \vec{p} \cdot \begin{bmatrix} \vec{q_i} \\ 1 \end{bmatrix} \right|. \tag{5.4}$$

Se o ponto $\vec{q_i}$ pertence ao plano $\vec{p}$ então $d_i = 0$, obedecendo à Equação 5.3.

Usando a Equação 5.4, se pode determinar se um dado ponto $\vec{q_i}$ está dentro ou fora de um limiar de distância t do plano $\vec{p}$. Então, o número de pontos da nuvem $C = \{\vec{q_1}, \vec{q_2}, \ldots, \vec{q_n}\}$ que estão dentro do limiar distância t do plano $\vec{p}$ pode ser computado usando

$$f\left(\vec{p}, C\right) = \sum_{i=1}^{n} \left(\left| \vec{p} \cdot \begin{bmatrix} \vec{q_i} \\ 1 \end{bmatrix} \right| < t \right). \tag{5.5}$$

Esta é a função de aptidão para o RANSAC clássico quando aplicado à extração de planos em nuvens de pontos. Ela dá o número de pontos da nuvem C que estão a uma distância menor que t do plano candidato $\vec{p}$. O limiar t é uma tolerância usada para compensar por irregularidades nos planos e ruído nos dados. O objetivo é maximizar esta função ao encontrar o plano que cobre a maior parte dos pontos em C, dada a tolerância de distância t. O valor ideal de t varia de acordo com os dados disponíveis, portanto deve ser definido empiricamente para um dado conjunto de dados. Ruído e irregularidades nas superfícies planas alvo são os fatores mais importantes a considerar ao definir t.

Gotardo et al. (2004) propõe um algoritmo que envolve uma variante de RANSAC chamado MSAC (TORR; ZISSERMAN, 2000) para a segmentação de superfícies de imagens de profundidade 2,5D. Esta abordagem consiste de uma fase de processamento que realiza segmentação de imagens baseada em arestas, seguida por uma extração de planos por segmento. A fase de extração de planos consiste do uso de um algoritmo genético como ferramenta de otimização e a função custo MSAC como função objetivo. Cada cromossomo é uma lista não ordenada de três elementos que referencia pontos da nuvem e representa um plano hipotético. Após a otimização por AG, os pontos pertencentes ao plano são extraídos e uma nova iteração ocorre. O processo se repete até que todos os pontos no conjunto de dados sejam extraídos. Os resultados mostram que o algoritmo supera outros algoritmos

de segmentação estado-da-arte em número de segmentos corretamente detectados sobre um conjunto de dados popular para teste. Entretanto, o estágio de pré-processamento se baseia na estrutura de imagem 2,5D para segmentação de arestas. Portanto, não pode ser aplicado a nuvens de pontos gerais não organizadas.

Wahl, Guthe e Klein (2005) constrói um modelo octree dos dados e aplica RANSAC a cada nó da árvore, detectando e extraindo planos passo-a-passo. Este método simplifica a nuvem de pontos para armazenamento eficiente e renderização híbrida, porém sua eficiência em tempo de execução não é competitiva. A decomposição espacial impõe a fragmentação de segmentos de plano, que são estimados separadamente em cada célula. Tal redundância reduz o desempenho e resulta em um modelo fragmentado.

O algoritmo de Yang e Förstner (2010) particiona previamente a nuvem em uma grade de tamanho fixo e aplica o RANSAC a cada bloco da grade, supondo haver no máximo três planos por bloco. A teoria da descrição de comprimento mínimo é então usada para estimar quantos planos de fato existem. A divisão da nuvem em uma grade permite que um objeto plano ocupe múltiplos blocos, tendo sua representação fragmentada. Um processo de crescimento de região funde planos vizinhos, encontrados em cada bloco, para contornar o problema da fragmentação. Nenhum dado quantitativo de desempenho é apresentado e nenhuma comparação com outros métodos é disponibilizada.

Em contraste, o método de Schnabel, Wahl e Klein (2007) não impõe nenhuma decomposição espacial antes da aplicação de RANSAC para a identificação de planos e também identifica outras formas geométricas primitivas, tais como esferas, cilindros, cones e toróides. O algoritmo toma amostras por toda a nuvem, avaliando quantos pontos se enquadram em cada modelo candidato e decide sobre extrai-los de acordo com uma estimativa probabilística de erro. As formas são extraídas iterativamente até que um critério de parada é atingido. Este critério vem de outra estimativa probabilística e estas são baseadas em distâncias espaciais entre amostras calculadas a partir de uma estrutura hierárquica de octree. Complexidade e eficiência em tempo de execução não foram comparadas a nenhum outro método patamar ou estado-da-arte.

Seguindo a mesma essência do algoritmo proposto por Gotardo et al. (2004), Bazargani, Mateus e Loja (2015) aplica um AG na otimização de uma função objetiva robusta logarítmica para a extração de superfícies planas de nuvens de pontos. Este método não é restrito a nuvens organizadas e nenhum passo de pré-processamento é necessário. Os cromossomos são definidos a valores reais e o espaço de busca cobre todos os planos representados por $ax + by + cz = d$. Em contraste ao método anterior, este não exige que os planos candidatos passem por quaisquer pontos da nuvem de pontos, teoricamente melhorando a busca em nuvens ruidosas e de baixa densidade. A busca é finalizada quando o número de pontos restantes fica abaixo de um limiar fixo. O ângulo diedral (ângulo entre os planos detectados e os polígonos de referência que compartilham ao menos um ponto

em comum) é considerado para a avaliação da detecção. Se o ângulo ficar abaixo de um determinado limiar, a detecção é dita bem-sucedida.

Propõe-se melhorar este método usando estratégias evolucionárias (EE) no lugar de AG para melhorar a eficiência e a robustez da busca. A adoção de EE tem o objetivo de tornar o método menos sensível e dependente de parâmetros definidos pelo usuário. O ângulo diedral é também adotado como métrica de qualidade, porém nenhuma limiarização é realizada, permitindo que os níveis de acurácia sejam comparados no lugar de um indicador binário de sucesso/falha. Adicionalmente, uma análise abrangente de sensibilidade de parâmetros é apresentada.

5.2 Estimadores robustos e consenso de amostragem

RANSAC se originou no campo de processamento de imagens como uma solução para ajuste de dados em cenários sujeitos a *outliers*, sem exigir filtragem manual prévia (FISCHLER; BOLLES, 1981). Vários problemas foram resolvidos usando RANSAC em vários campos da engenharia, especialmente em visão computacional e no reconhecimento de padrões (TORR; ZISSERMAN, 2000; CHUM; MATAS, 2005; SCHNABEL; WAHL; KLEIN, 2007). RANSAC é, entretanto, um caso especial de uma teoria mais abrangente chamada de estimação robusta e estudos mostraram que outros estimadores robustos podem apresentar melhores resultados quando aplicados no lugar de RANSAC (CHOI; KIM; YU, 2009). Esta seção introduz brevemente o tópico para que o leitor possa melhor apreciar como nossa abordagem e outras abordagens recentes diferem na detecção de planos em nuvens de pontos.

5.2.1 Estimadores robustos

Regressão baseada em mínimos quadrados é altamente sensível a *outliers*. Um único *outlier* pode comprometer todo o processo de estimação porque esta abordagem amplifica a influência de amostras distantes. Cenários ricos em *outliers* são comuns em várias aplicações de visão computacional, onde várias amostras vêm de falsas estimativas de correspondência e correlação de pontos. Funções de influência, também chamadas de funções de perda (CHOI; KIM; YU, 2009), foram estudadas ao longo das décadas recentes (TUKEY, 1960; ROUSSEEUW, 1984; HUBER, 2011). O método robusto mais popular, o método dos estimadores -m, foi aplicado com sucesso na solução de vários problemas de visão computacional e habilitou vários métodos a operar confiavelmente sobre dados reais (ZHANG, 1998).

Sob a óptica de estimadores robustos, a regressão é um processo de otimização que minimiza as influências dos erros de um determinado modelo, encontrando os parâmetros

mais adequados para tal. O mesmo pode ser descrito de maneira geral por

$$min \sum_i \rho(e_i). \tag{5.6}$$

onde $\rho(e_i)$ representa a função de influência do estimador. Neste *framework*, a função de influência dos mínimos quadrados (L_2) é $\rho_{L_2}(e_i) = e_i^2$ e para o RANSAC clássico é

$$\rho_{RANSAC}(e_i) = \begin{cases} 0 & \text{se } |e_i| < t \\ 1 & \text{se } |e_i| \geq t. \end{cases}. \tag{5.7}$$

onde t é o limiar de tolerância definido pelo usuário. A Figura 17 apresenta gráficos de funções de influência comuns. L_1 é chamado de mínimos absolutos, tendo $\rho(e_i) = |e_i|$. Note que para L_1 e L_2 a influência cresce indefinidamente com o erro, em contraste às funções robustas, que são constantes após determinado limiar. A função de perda robusta proposta por Tukey (1960) oferece uma curva suave, porém sua fórmula não é trivial:

$$\rho_{Tukey}(e_i) = \begin{cases} \frac{t^2}{6}(1 - (1 - (\frac{e_i}{t})^2)^3) & \text{se } |e_i| < t \\ \frac{t^2}{6} & \text{se } |e_i| \geq t. \end{cases}.. \tag{5.8}$$

A necessidade de definir limiares é uma das dificuldades de usar estimadores -m. A mediana mínima dos quadrados (do inglês, *least median of squares* – LMedS) consiste de substituir a somatória pelo operador mediana, tal que o processo se torna *min mediana* e_i^2. Este estimador não requer a definição de nenhum parâmetro e é robusto a *outliers*, porém se torna inválido para taxas de *outliers* superiores a 50% (ROUSSEEUW, 1984). A extração de planos por consenso de amostragem se baseia na suposição de que os pontos de um plano são *inliers* e que todos os demais pontos em uma cena são *outliers*. Assim, LMedS não pode ser aplicado para detecção de planos em nuvens de pontos em geral. Na maior parte dos casos, cada plano constitui apenas uma fração de todos os pontos.

A função de influência MSAC, proposta por Torr e Zisserman (2000) para estimar geometria epipolar e projetividades, foi aplicada com sucesso por Gotardo et al. (2004) para reconhecimento de planos em nuvens de pontos organizadas. O método utiliza MSAC para estimação robusta por combinar a robustez do RANSAC clássico com o refinamento de L_2. Como apresentado pela Figura 17, MSAC se comporta como L_2 para $|e_i| < t$ e é constante para $|e_i| \geq t$:

$$\rho_{MSAC}(e_i) = \begin{cases} e_i^2 & \text{se } |e_i| < t \\ t^2 & \text{se } |e_i| \geq t. \end{cases}. \tag{5.9}$$

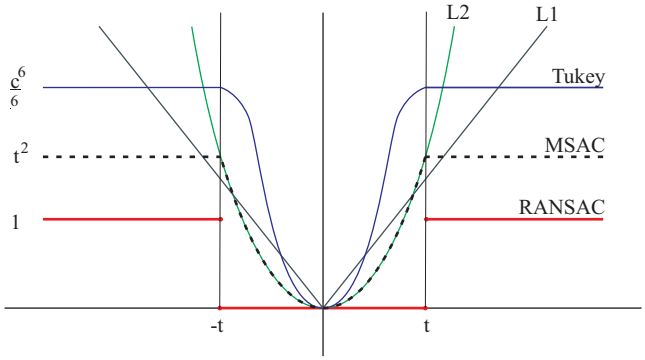

Figura 17 – Funções de influência comuns.

5.2.2 Consenso de amostragem

RANSAC é um método de Monte Carlo que inicializa tomando k parâmetros aleatórios de um conjunto de n candidatos. Estes k parâmetros são usados para construir um modelo que é avaliado conta todos os demais parâmetros no conjunto de dados. Este processo itera até que uma certa combinação de parâmetros satisfaça uma medida de consenso, segundo um critério de parada.

A avaliação do modelo no RANSAC clássico consiste em aferir quantas amostras se adequam ao modelo, dado um limiar de tolerância t , como na Equação 5.7. Variações de RANSAC utilizam outros estimadores -m no lugar. k corresponde ao número mínimo de parâmetros necessários para determinar o modelo. Assim, avaliar todas as possibilidades exigiria $\binom{n}{k} = \frac{n!}{k!(n-k)!}$ iterações, mas para a maior parte das aplicações, apenas uma pequena fração deste total é necessário para encontrar uma solução próxima do ótimo.

Estudos propuseram grande variedade de abordagens de amostragem e de critérios de parada, porém muitos exigem fazer suposições *a priori*, tais como sobre a proporção de *outliers* e considerações específicas do modelo (TORR; ZISSERMAN, 2000; MATAS; CHUM, 2005; CHUM; MATAS, 2005; HAST; NYSJÖ; MARCHETTI, 2013). Algumas abordagens focaram em tornar o processo mais robusto, mais acurado ou mais rápido adotando avaliação parcial, critérios de parada adaptativos, amostragem guiada, seleção de modelos e otimização local. Uma análise abrangente da família RANSAC é apresentada por Choi, Kim e Yu (2009).

No caso de extração de nuvens de pontos, três pontos distintos são tomados de uma nuvem de n pontos. O plano definido por estes três pontos é avaliado calculando a distância euclidiana entre si e os demais pontos da nuvem. Se a distância absoluta é maior que t, um *outlier* é contado. Quando o processo termina, o plano com menos *outliers* é retornado. Inferência estatística é comumente utilizada para determinar previamente o número de iterações a realizar, mas tais inferências requerem fazer suposições específicas que causam perda de generalidade e robustez.

Wahl, Guthe e Klein (2005) realiza um número fixo de iterações dependendo de quantos planos se espera encontrar em cada célula. Schnabel, Wahl e Klein (2007) estabelece um critério de parada baseado no número de pontos estimados para cada forma. Yang e Förstner (2010) supõe que os pontos se distribuem uniformemente sobre cada plano, além de limitar em três o número de planos a encontrar em cada nó da grade. Nenhum critério geral, livre de suposições iniciais ou considerações sobre as cenas representadas, foi proposto ainda. Nossa contribuição principal é propor um método menos sensível aos valores dos parâmetros. O mesmo requer uma estimativa do desvio padrão do sensor, mas nenhuma suposição sobre o conteúdo de cada cena é necessária.

5.3 Algoritmos evolutivos e consenso de amostragem

Algoritmos evolutivos combinam características de descida gradiente e métodos de Monte Carlo. Soluções candidatas são corrompidas (mutadas) estocasticamente, avaliadas e selecionadas a cada iteração. As três principais categorias diferem na maior parte dos aspectos de implementação (SPEARS et al., 1993; FOGEL, 1995; JONG, 2006).

Indivíduos baseados em valores reais permitem representação mais contínua de soluções e utilização efetiva de mutação gaussiana. A mutação adaptativa pode melhorar a velocidade de convergência sem comprometer a acurácia e a estabilidade da busca. Adicionalmente, estas técnicas podem também reduzir o número de parâmetros predefinidos, tornando a busca mais robusta.

Rodehorst e Hellwich (2006) propôs aplicar AG a problemas de consenso de amostragem, cunhando o termo GASAC. GASAC substitui tentativas aleatórias por uma busca sistemática usando operações AG padrão. Diferentes estimadores -m são avaliados como funções de aptidão para GASAC e LMedS é considerada a melhor opção para casos com proporções de *outliers* abaixo de 50%. Aproximações GASAC se mostraram mais acuradas – alcançando até 14% mais acurácia no teste proposto por Choi, Kim e Yu (2009). Nenhuma inferência prévia sobre os dados é exigida. Entretanto, provou-se que GASAC é computacionalmente mais caro que RANSAC – seu tempo de processamento médio foi cinco vezes mais longo (CHOI; KIM; YU, 2009). A aplicação de GASAC para extração de planos resultaria na abordagem proposta por Gotardo et al. (2004), publicado três anos mais cedo.

Toda e Kubota (2014) introduz ESSAC com EE no lugar de AG para consenso de amostragem para melhorar o balanço entre custo computacional e estabilidade da busca. Adotando o paradigma clássico de EE, ESSAC utiliza cromossomos a valores reais, mutação gaussiana com média zero e nenhum cruzamento. A mutação adaptativa é proposta para que cada indivíduo possua seus próprios parâmetros de variância de mutação, que também estão sujeitos à mutação.

A regra de sucesso de 1/5 é adotada, consistindo em incrementar ou decrementar a variância da população de acordo com a proporção de mutações bem-sucedidas a cada geração. O ESSAC proposto adota a função de perda do RANSAC clássico. Uma estimação de homografia matricial é utilizada para comparar ESSAC e RANSAC. Em seus testes, Toda e Kubota (2014) mostra que ESSAC converge rapidamente para o mínimo global, eficientemente concentrando os esforços de busca. A Tabela 1 apresenta uma comparação dos principais atributos dos três diferentes métodos de detecção de planos.

Tabela 1 – Métodos evolucionários de detecção de planos.

Método	*Framework*	Representação	Capaz de tratar nuvens de ponto não-organizadas
Gotardo et al. (2004)	GASAC	Inteiro	não
Bazargani, Mateus e Loja (2015)	GA	Real	sim
Proposto	ESSAC	Real	sim

5.4 Algoritmo proposto

Dada uma nuvem de entrada $C = \{\vec{q}_1, \vec{q}_2, \ldots, \vec{q}_i, \ldots, \vec{q}_n\}$, o algoritmo aplica ESSAC para iterativamente detectar e extrair formas planas. Seguindo a mesma essência dos algoritmos de Gotardo et al. (2004) e Bazargani, Mateus e Loja (2015), o método encontra e extrai plano após plano até que um critério de parada é encontrado. Enquanto o primeiro itera até que todos os planos tenham siso removidos, o segundo estabelece um limiar para o número mínimo de pontos por plano. Para o algoritmo proposto, em contraste, se define um parâmetro g de objetivo de extração que define qual proporção dos pontos deve ser extraída. Este parâmetro varia de $0,0$ – o processo para assim que o primeiro plano é encontrado – até 1,0 – o processo itera até que todos os pontos sejam extraídos. Quando a aplicação é, por exemplo, estimar o plano do piso, o valor de $g = 0,0$ permite encontrar o maior plano da cena, geralmente o piso. Quando o objetivo é extrair os principais planos de uma estrutura artificial, onde nem todos os elementos da cena são planos, um valor intermediário de g deve ser definido para refletir a quantidade de pontos de plano na cena. Isso torna o algoritmo configurável para diferentes aplicações.

Em notação de EE, E(μ+λ) indica que há μ pais e λ filhos, cada um representado por $\vec{p} = \begin{bmatrix} a & b & c & d \end{bmatrix}$ e $\vec{\sigma} = \begin{bmatrix} \sigma_n & \sigma_d \end{bmatrix}$. O vetor $\vec{p}$ contém os parâmetros do plano e o vetor $\vec{\sigma}$ contém meta-parâmetros que permitem a coevolução, definindo o desvio padrão da mutação do indivíduo. Assim, opera-se sobre a população de candidatos $P = \{\langle\vec{p}_1, \vec{\sigma}_1\rangle, \langle\vec{p}_2, \vec{\sigma}_2\rangle, \ldots, \langle\vec{p}_\lambda, \vec{\sigma}_\lambda\rangle, \ldots, \langle\vec{p}_{\lambda+\mu}, \vec{\sigma}_{\lambda+\mu}\rangle\}$

O erro e_i é calculado como a distância absoluta d_i entre o plano candidato $\vec{p}$ e cada ponto $\vec{q}_i$ usando a Equação 5.4. A influência do erro é avaliada usando MSAC como

Tabela 2 – Parâmetros para o algoritmo de extração de planos proposto.

Parâmetro	Descrição
g	Depende do objetivo da aplicação
μ	Parâmetro de busca
λ	Parâmetro de busca
t	Erro médio do sensor
ε	Aproxima zero

na Equação 5.9. O parâmetro limiar de MSAC, t, é definido igual ao erro esperado do sensor, que é frequentemente bem especificado para um dado sensor. Este é um processo de minimização, portanto, a função de aptidão, como na Equação 5.6, é de fato uma função de custo e mede o quão ruim é o candidato.

Três pontos distintos da nuvem são selecionados aleatoriamente para inicializar cada indivíduo, seguindo o padrão clássico de RANSAC. Os pais são avaliados e ordenados. Os μ melhores produzem descendentes para substituir os λ candidatos remanescentes. Cópias mutadas dos pais são criadas corrompendo cópias perfeitas com erro gaussiano de média zero. O parâmetro σ_n define o desvio padrão para a mutação dos componentes do vetor normal, a, b e c e é inicializado como unitário. O parâmetro σ_d define o desvio padrão para a mutação do parâmetro de deslocamento do plano d e é inicializado como t. σ_n e σ_d definem seus próprios desvios padrão de mutação. O vetor $\vec{p}$ é normalizado como unitário após a geração de descendentes.

O processo de otimização finaliza quando ambos os valores de σ_n e σ_d ficam abaixo de um parâmetro limiar ε, indicando que um plano foi encontrado. Este limiar deve aproximar zero. Adotou-se $\varepsilon = 0,001$ para este estudo. Esta condição de parada torna o processo menos dependente de parâmetros específicos de cada nuvem, tais como o número de pontos esperados por plano ou a densidade de pontos. A Tabela 2 apresenta todos os parâmetros necessários para este algoritmo.

A regra de sucesso de 1/5 é aplicada a cada geração. A mesma consiste em manter um registro de aptidão do pai de cada indivíduo e compara-lo com a aptidão do próprio indivíduo para estimar a taxa de sucesso. Se a taxa de sucesso está abaixo de 20%, todos os desvios padrão do indivíduo são reduzidos em 20%. O Algoritmo 2 apresenta a operação EE em alto nível. Esta rotina se repete até que a proporção objetiva de pontos, definida por g, tenha sido extraída, como apresentado pelo algoritmo 3.

5.5 Validação e teste

A Figura 18 (a) apresenta uma nuvem sintetizada para avaliar a sensibilidade de parâmetros e validar o método proposto. A mesma representa um galpão de 12 m x 8

Data: Nuvem de pontos C
Result: Plano $\vec{p}$ que aproxima uma forma plana extraída da nuvem.
1 Inicialize P como no RANSAC;
2 **repeat**
3 | calcule a aptidão para todos os $\lambda + \mu$ indivíduos;
4 | ordene todos os $\lambda + \mu$ indivíduos segundo aptidão;
5 | selecione os μ melhores indivíduos como pais;
6 | gere λ cópias mutadas dos μ pais;
7 | substitua os λ velhos com as λ cópias;
8 **until** $\sigma_n < \varepsilon$ *e* $\sigma_d < \varepsilon$;
9 extraia os pontos correspondentes ao melhor indivíduo $\vec{p}$;

Algoritmo 2: ESSAC para extração de planos.

Data: Nuvem de pontos C
Result: Conjunto de planos que aproximam todas as formas planas extraídas de C.
1 **repeat**
2 | Detectar e extrair plano usando algoritmo 2.
3 **until** $\frac{\textit{número de pontos extraídos de } C}{\textit{número total de pontos de } C} > g$;

Algoritmo 3: Processo iterativo de extração de planos.

m sobre um plano de 20 m x 20 m. É uma boa representante de estruturas artificiais, com um conjunto variado de polígonos em diferentes formas e orientações. Os pontos são gerados estocasticamente, uniformemente distribuídos a uma concentração de cem pontos por metro quadrado e corrompidos por ruído gaussiano de média zero com desvio padrão de 1 cm nos eixos x, y e z, que aproxima o ruído médio obtido em sensores lidar. Os polígonos usados para gerar e testar a nuvem são representados por diferentes cores na Figura 18 (b). Os planos encontrados pelo processo de segmentação proposto são representados por diferentes cores na Figura 18 (c). Note uma faixa horizontal nas portas do galpão. Uma vez que esta é uma implementação básica do método proposto, nenhum pós-processamento limita os planos encontrados. Estes são considerados infinitos, e, assim, as faixas representam pontos cobertos pelo plano aproximador do topo do cubo que está fora do galpão.

A Figura 19 apresenta passos do processo em um código de cores diferente. Pontos verdes já foram extraídos e os pontos azuis ainda não foram. Tais resultados foram obtidos ao definir diferentes valores para g. Note que para $g = 0,00$, o algoritmo finalizará quando um plano for detectado, enquanto para $g = 0,99$ o mesmo finaliza quando a maior parte (ou todos) dos planos for detectada. Em algumas aplicações, supõem-se que todos os pontos constituem planos – tal como na engenharia reversa de certas estruturas e peças mecânicas. Em outras aplicações, tais como em percepção robótica, não se supõe que todos os pontos constituem planos porque a cena observada pelo robô pode conter elementos não planares, tais como vegetação e formas curvas. Dessa forma, g pode ser ajustado para retornar a proporção esperada de pontos de superfícies planas.

A Figura 20 apresenta gráficos do processo evolutivo para diferentes valores de μ e λ. O custo converge para um mínimo e os desvios aproximam zero. Para estes testes, $\varepsilon = 0,0001$, indicado pela linha roxa na coluna do meio. A evolução finaliza quando $\sigma_n < \varepsilon$ e $\sigma_d < \varepsilon$.

5.5.1 Resultados obtidos com dados sintéticos

Nossa implementação de MSAC foi adotada como o método-patamar para comparação com o ESSAC proposto. Para MSAC, um número fixo de candidatos é avaliado e o melhor é tomado como plano detectado. Ambos são executados como rotinas chamadas de dentro do laço do algoritmo 3. Os gráficos da Figura 21 foram gerados ao definir o critério de parada g para $0,70$, $0,90$ e $0,99$, $\frac{\mu}{\mu+\lambda} = 1/16$ e com tamanho de população variando de 16 a 8192. A Figura 21 (a) apresenta o número de planos detectados e a Figura 21 (b) apresenta o ângulo diedral médio entre os planos detectados e os planos de referência. Cada ponto nestes gráficos foi obtido fazendo a média de mil execuções para obter validade estatística. MSAC e ESSAC convergem para o mesmo número de planos detectados para qualquer valor de g, dada uma população de tamanho suficientemente grande. ESSAC converge mais rápido porque itera várias vezes sobre a mesma população, enquanto o MSAC considera cada candidato apenas uma vez. O número elevado de planos detectados para pequenas populações é uma anomalia observada para tamanhos de população abaixo de aproximadamente 128. Este comportamento resulta de uma detecção equivocada de planos que estão distantes do plano de referência, causando elevados valores de ângulo diedral. Entretanto, populações de tamanho superior a 128 mitigam o problema.

Gráficos tais como da Figura 21 não oferecem uma comparação justa entre MSAC e ESSAC porque o custo de processamento reside nas avaliações da função de aptidão, não no tamanho da população. Para oferecer uma comparação quantitativa válida, é necessário analisar as métricas para um número fixo de funções custo. O número de avaliações de custo varia ao executar ESSAC, mesmo mantendo os mesmos valores de parâmetros, portanto, os gráficos apresentam amostragem irregular como na Figura 22. Tais gráficos são gerados ao incrementar o tamanho da população e medir o número resultante de avaliações da função custo. A Figura 22 mostra que menos avaliações da função custo são necessárias para obter o mesmo ângulo diedral para menores razões de pai/tamanho da população ($\frac{\mu}{\mu+\lambda}$). Entretanto, razões menores que 1/16 não apresentaram nenhum benefício observável para as faixas de parâmetros estudadas.

A Figura 23 apresenta uma comparação entre MSAC e ESSAC com $\frac{\mu}{\mu+\lambda} = 1/16$, com o número de avaliações da função variando de 256 a 65536. Note que o ESSAC não apresenta dados para números baixos de avaliações da função custo. Ao reduzir o tamanho da população, o número de avaliações da função custo satura em um mínimo, não gerando resultados abaixo de 512. Este fenômeno ocorre porque, ao decrementar o tamanho da

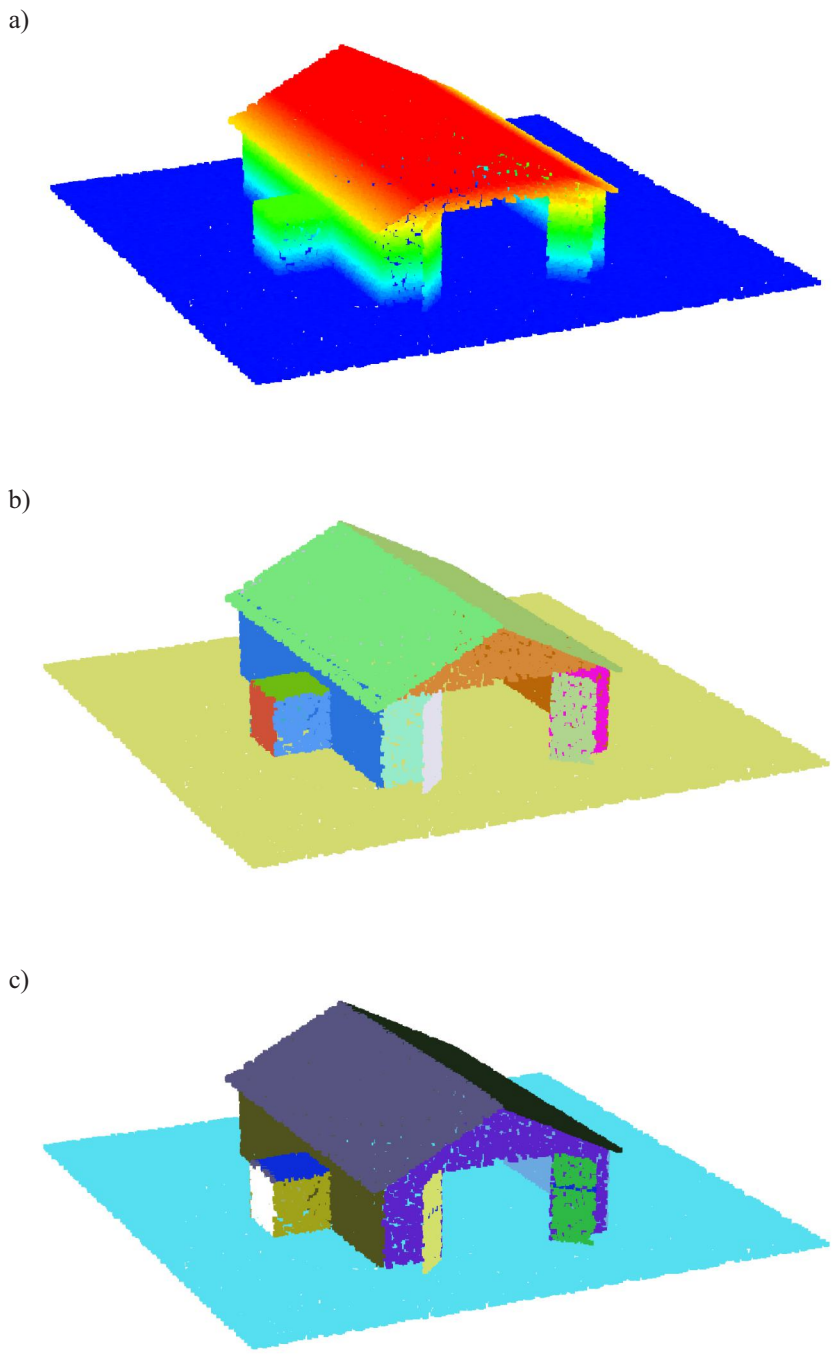

Figura 18 – Resultados de execução com $g = 0,99$ para uma nuvem de pontos sintetizada. Os pontos são apresentados em tamanho aumentado para facilitar a visualização: (a) apresenta a altura e acordo com o código de cores – de 0 m a 5 m, de azul a vermelho; em (b) diferentes cores correspondem a diferentes polígonos de referência; e em (c) diferentes cores correspondem a diferentes planos detectados.

população, o número de gerações automaticamente cresce para compensar. Em outras palavras, são necessárias mais gerações para a convergência do algoritmo de busca, o que

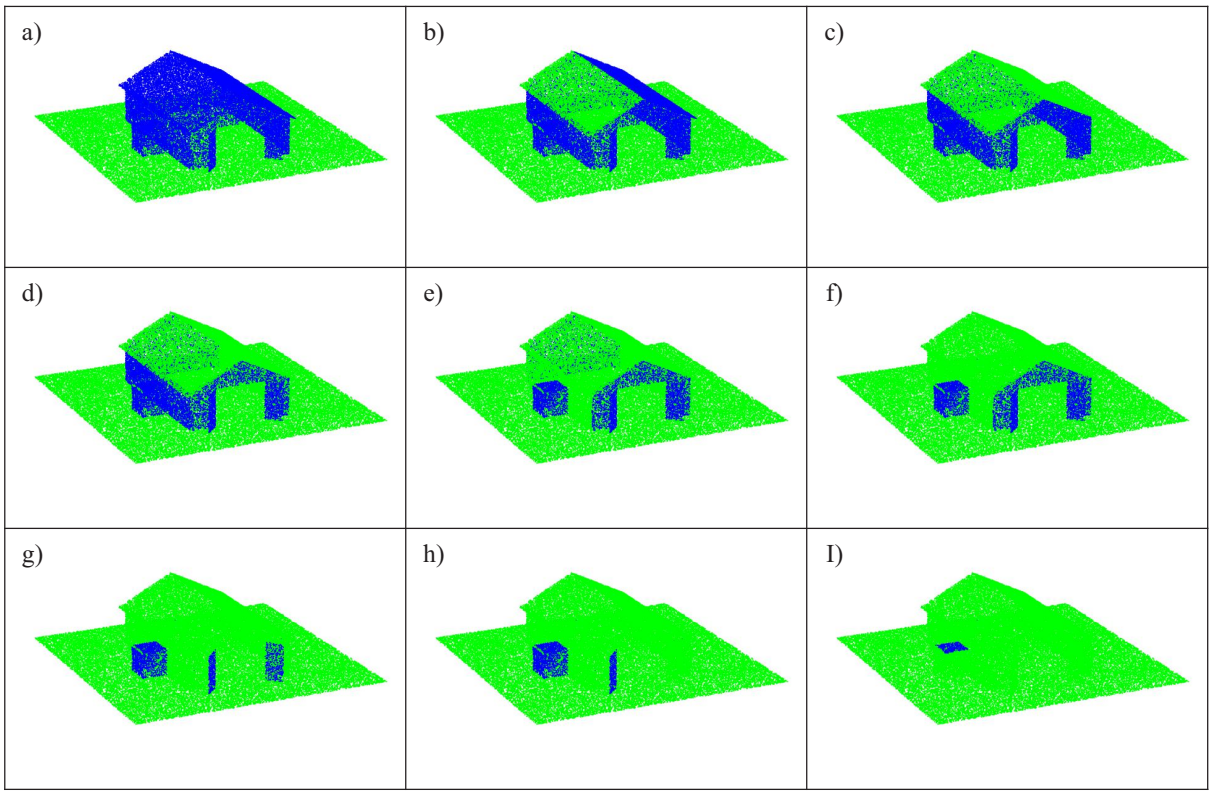

Figura 19 – Seqüência de detecção de planos. Pontos verdes foram extraídos e pontos azuis não foram. (a) $g = 0,00$; (b) $g = 0,60$; (c) $g = 0,70$; (d) $g = 0,80$; (e) $g = 0,85$; (f) $g = 0,90$; (g) $g = 0,95$; (h) $g = 0,97$; (i) $g = 0,99$.

incrementa o número de avaliações da função custo. Nestas regiões, é impossível continuar a reduzir o tamanho da população e manter a mesma proporção $\frac{\mu}{\mu+\lambda} = 1/16$.

MSAC produz melhores resultados nestas regiões. Entretanto, para obter maior acurácia e reduzir os ângulos diedrais, ESSAC é vantajoso. A partir de duas mil avaliações de custo, ESSAC produz erros mais baixos tanto para $g = 0,70$ quanto $g = 0,90$. Para $g = 0,99$, ESSAC e MSAC produzem os mesmos resultados de acurácia a partir de três mil avaliações de custo. Para todos os três valores de g, o ponto onde as curvas de ESSAC e MSAC se cruzam é crítico, definindo áreas onde um ou o outro é vantajoso. Dessa análise, pode-se concluir que ESSAC requer menos esforço computacional quando elevada acurácia é desejada. Este resultado é esperado porque ESSAC trabalha com uma busca sistemática que concentra esforço de busca adaptativamente, enquanto MSAC busca em todo o espaço indistintamente.

5.5.2 Resultados dos dados de escaneamento

Uma nuvem de pontos disponibilizada por Borrmann et al. (2012) foi adotada para avaliar o algoritmo com dados reais. Registrada com um escaneador Riegl VZ-400,

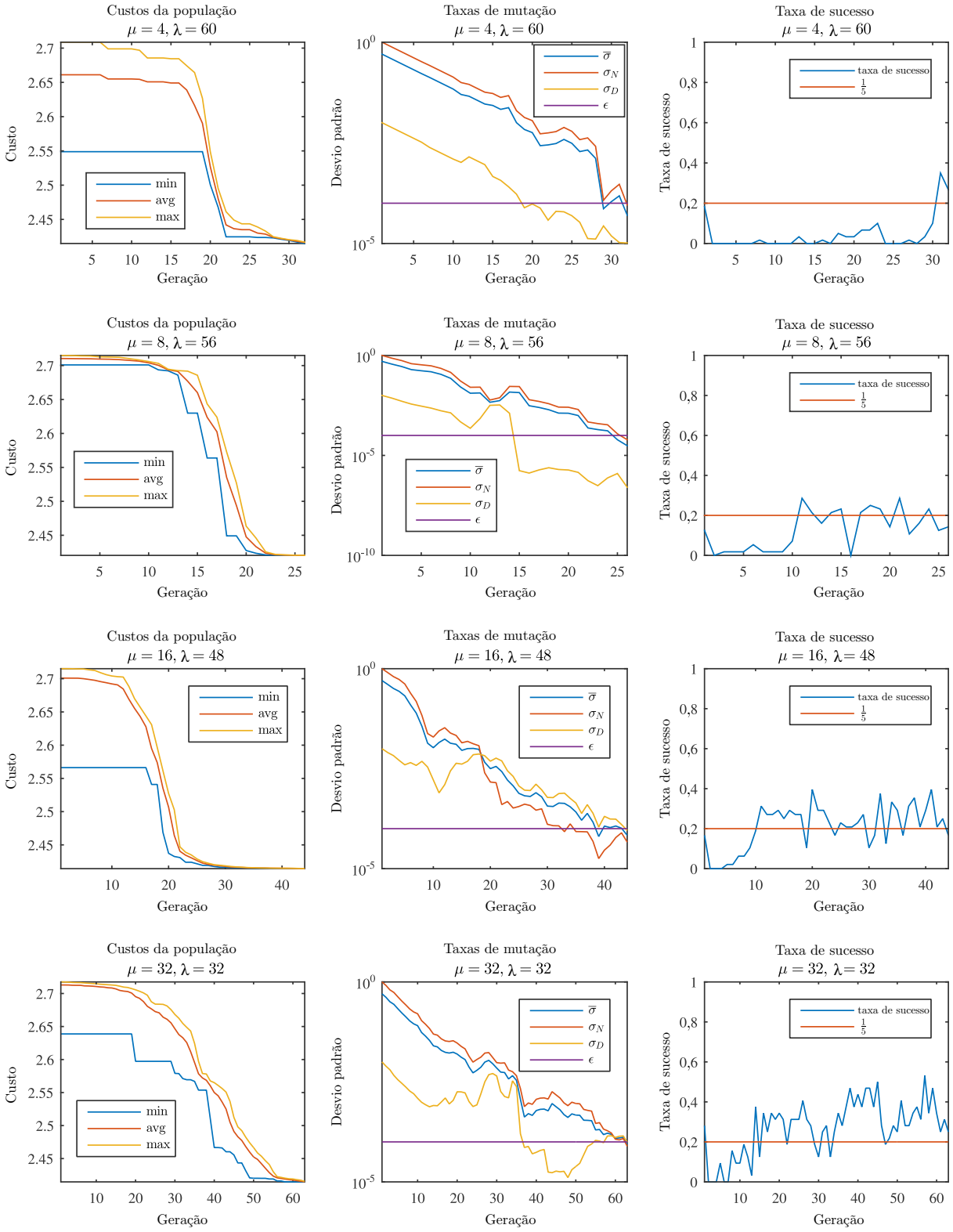

Figura 20 – Processo evolutivo para diferentes valores de μ e λ. $g = 0,70$ e $\varepsilon = 0,0001$. A coluna à esquerda apresenta os custos máximo, mínimo e médio para cada geração. Desvios padrão de mutação são apresentados pela coluna direita em escala logarítmica.

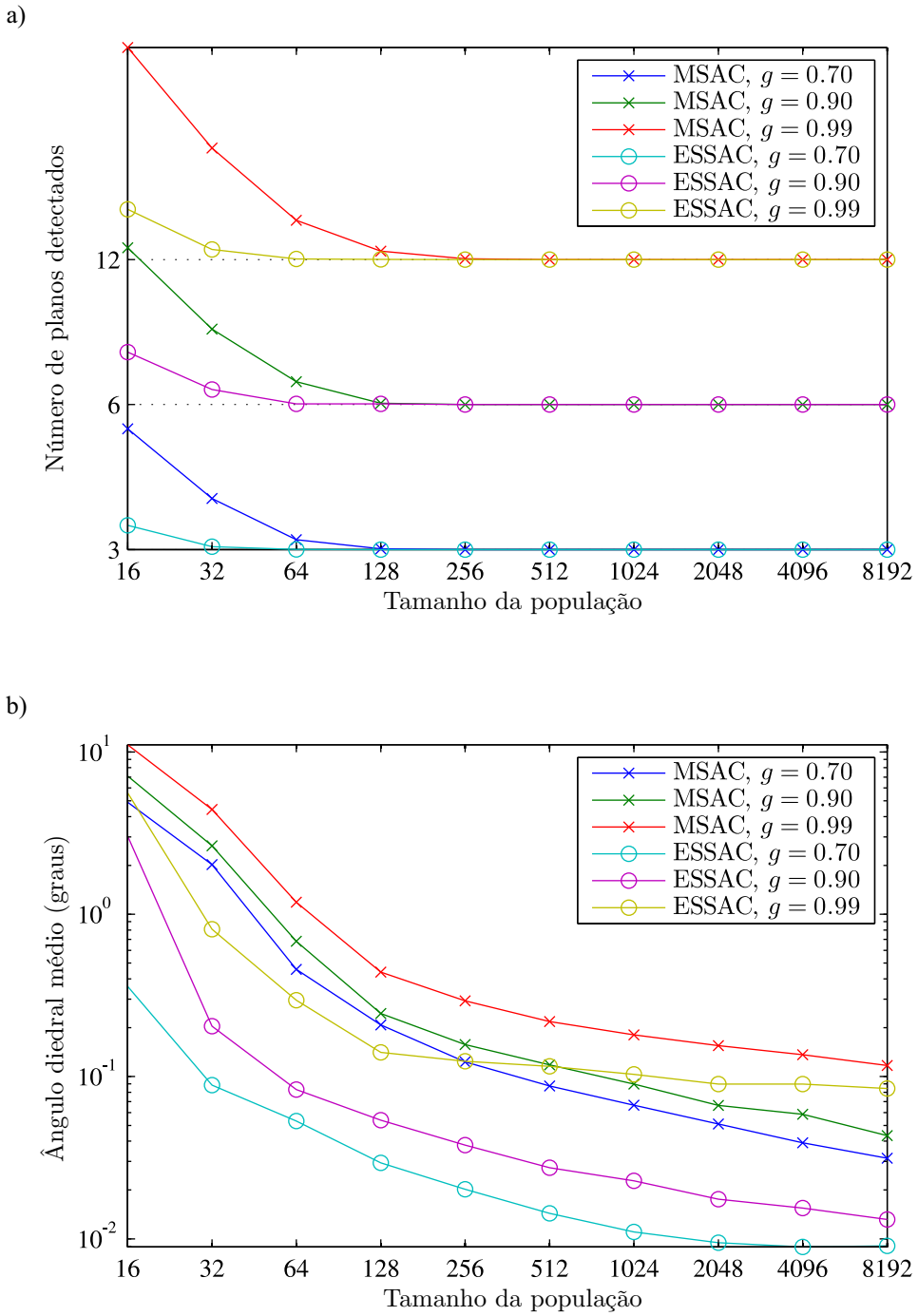

Figura 21 – Resultados dos testes para diferentes valores de critério de parada g e de tamanhos de população para MSAC (TORR; ZISSERMAN, 2000) e ESSAC. $\frac{\mu}{\mu+\lambda} = 1/16$ e $\varepsilon = 0,001$. (a) Número de planos detectados; (b) Ângulo diedral médio entre o plano de referência e o plano detectado. Cada ponto representado resulta de uma média de mil amostras para elevada significância estatística.

consiste de várias capturas tridimensionais tomadas no Laboratório de Automação da Jacobs University Bremen. Trata-se de um bom representante de ambientes internos, repleto de objetos típicos de um escritório. A Figura 24 apresenta uma visão perspectiva

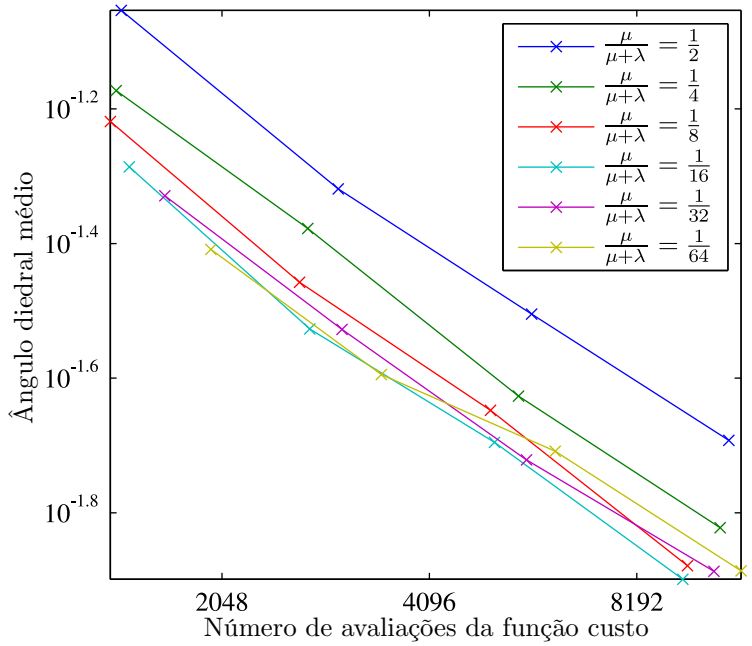

Figura 22 – Ângulo diedral médio para diferentes razões pai/tamanho da população $\frac{\mu}{\mu+\lambda}$. $g = 0,70$ e $\varepsilon = 0,001$. Cada ponto representado resulta de uma média de mil amostras para elevada significância estatística.

desta nuvem de pontos.

A implementação de RANSAC da biblioteca de nuvens de pontos (do inglês, *point cloud library* – PCL) (RUSU; COUSINS, 2011) foi adotada como patamar de comparação. PCL consiste de um projeto de código aberto financiado por instituições acadêmicas e industriais. Esta biblioteca oferece várias implementações SAC para segmentação baseada em modelos e implementa o RANSAC clássico proposto por Fischler e Bolles (1981). A Figura 25 apresenta os resultados de teste para ambos os métodos a diferentes valores de g. A Figura 25 (a) mostra que ambas as abordagens convergem para o número real de planos a encontrar, dadas iterações suficientes. A Figura 25 (b) mostra que ESSAC requer menor tempo de execução para dado ângulo diedral. Os testes foram realizados executando cem vezes para cada conjunto de parâmetros e então realizando a média dos valores obtidos. Os tempos de execução avaliados variam de 10^3 a 10^5 ms.

Como esperado, mais acurácia (menor ângulo diedral médio) é obtida com maior tempo de computação. Entretanto, para dado tempo de execução, o algoritmo proposto obteve, em média, erros uma ordem de magnitude menor. Este resultado pode ser explicado pelo fato de que RANSAC amostra todos os planos possíveis indiscriminadamente, enquanto ESSAC realiza uma busca focada. Isso economiza poder computacional que RANSAC desperdiça avaliando parâmetros muito distantes do conjunto ótimo.

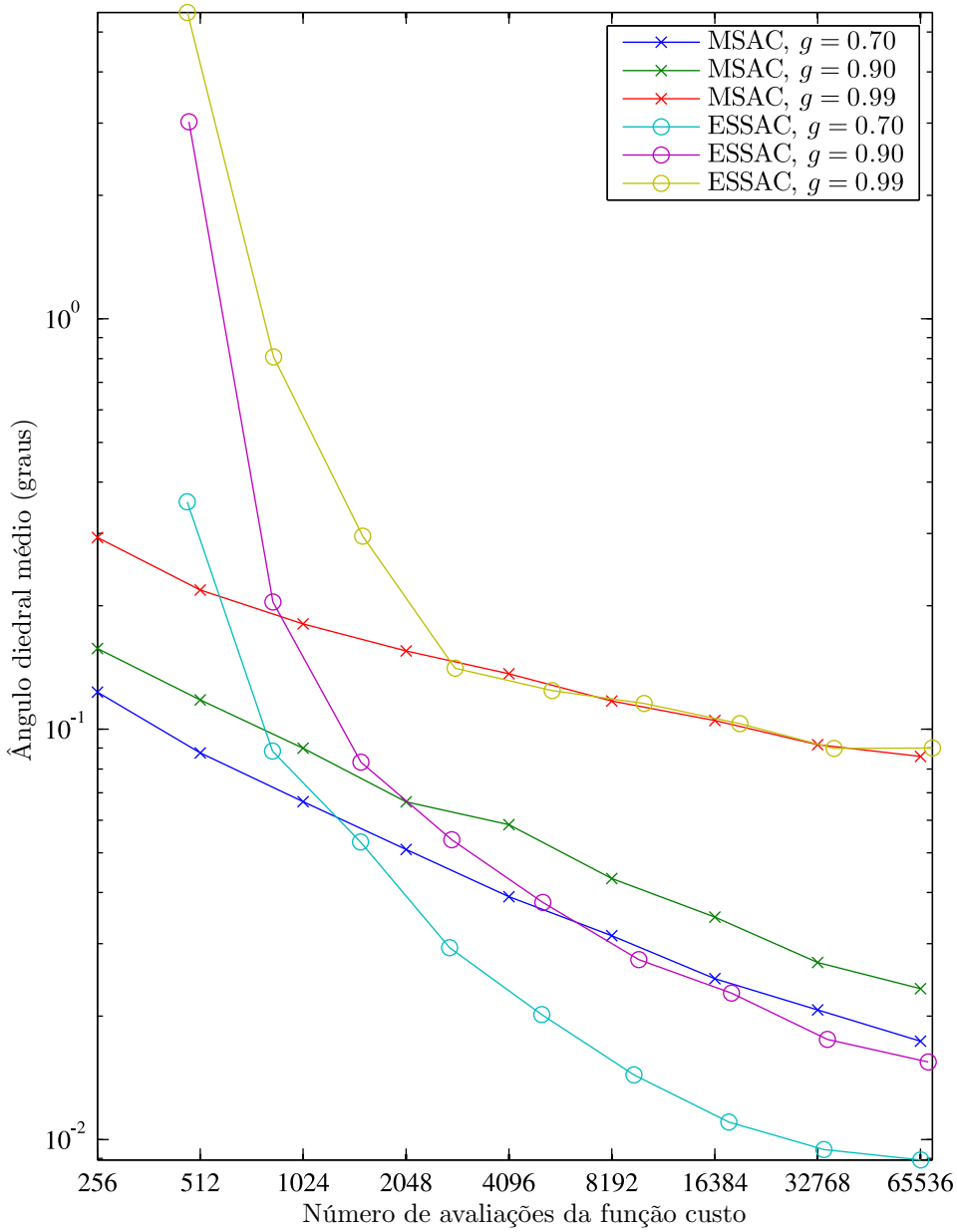

Figura 23 – Ângulo diedral médio para diferentes valores do critério de parada g para MSAC (TORR; ZISSERMAN, 2000) e ESSAC. $\frac{\mu}{\mu+\lambda} = 1/16$ e $\varepsilon = 0,001$. Cada ponto representado resulta de uma média de mil amostras para elevada significância estatística.

5.6 Conclusões

Este capítulo apresenta uma nova abordagem para extração de formas planas de nuvens de pontos não organizadas aplicando ESSAC. Comparada a outras abordagens estado-da-arte, tais como RANSAC e MSAC, ela resulta em um melhor balanço entre custo e acurácia quando os requerimentos são elevados. Este atributo a torna altamente apropriada para representação de estruturas artificiais e peças mecânicas.

Em cenas complexas, tais como a cena apresentada pela Figura 24, ESSAC requer

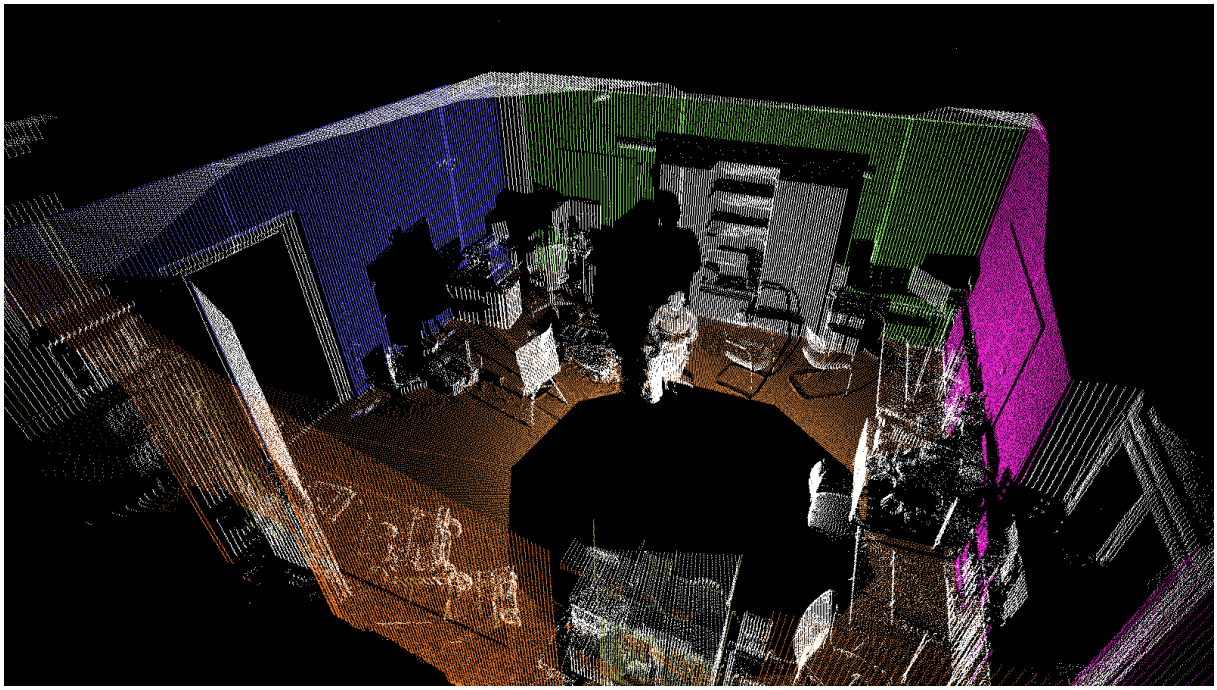

Figura 24 – Visão perspectiva dos resultados de segmentação de planos para a nuvem de pontos do laboratório de automação disponibilizada por Borrmann et al. (2012), usando o método proposto.

um tempo de execução consideravelmente menor que o método de consenso de amostragem totalmente aleatório. Diferentemente de outros métodos, tal como o proposto por Bazargani, Mateus e Loja (2015), ESSAC não requer nenhuma suposição específica sobre os conteúdos da nuvem, tais como o tamanho mínimo dos planos ou o número mínimo de pontos por plano. A taxas de mutação adaptativas fazem com que a busca se auto ajuste a mudanças de parâmetros, como demonstrado pela análise de sensibilidade.

Ao combinar este método com uma subdivisão adaptativa por árvores quaternárias, o autor propõe um método para detecção e extração de piso em ambientes urbanos para navegação robótica (MARCON et al., 2015). O Capítulo 6 apresenta uma análise detalhada de desempenho do algoritmo em um estudo de caso com dados reais coletados por um escaneador lidar em um ambiente urbano. São apontadas as vantagens e desvantagens da aplicação do método para este fim através de uma análise fundamentada em métricas de qualidade.

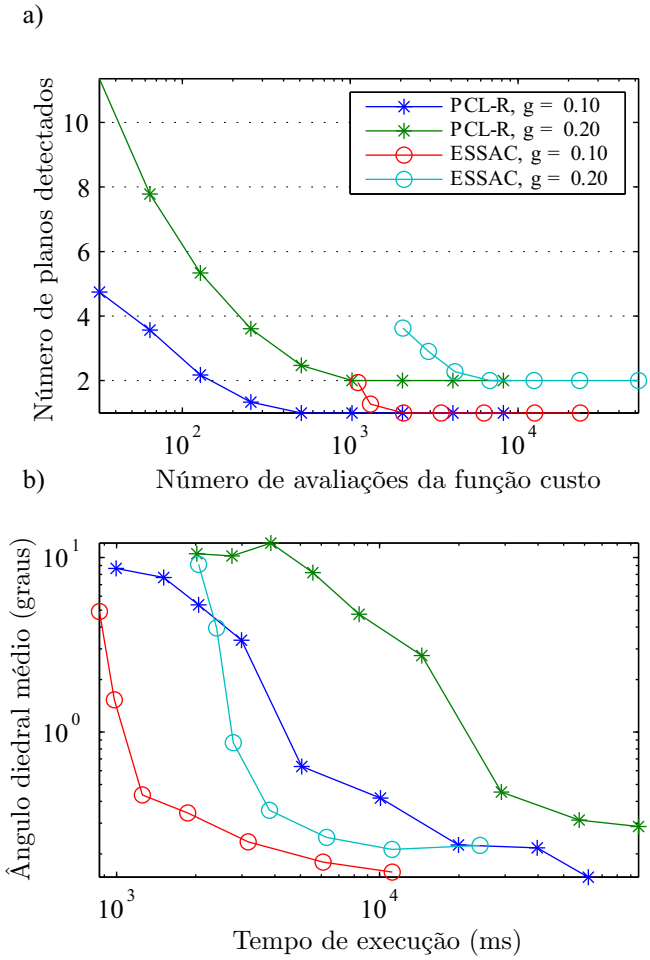

Figura 25 – Resultados de teste para a nuvem de pontos do laboratório de automação (BORRMANN et al., 2012) para o RANSAC PCL (RUSU; COUSINS, 2011) e para o ESSAC proposto. (a) Número de planos detectados versus número de avaliações de função custo; e (b) ângulo diedral médio versus tempo de execução. Cada ponto representado resulta de uma média de mil amostras para elevada significância estatística.

6 Aplicação de ESSAC para extração de piso em nuvens de pontos

O Capítulo 5 propõe a aplicação de consenso de amostragem com estratégias evolucionárias (ESSAC) para a detecção de planos em nuvens de pontos. Em publicação recente (MARCON et al., 2015), o autor propõe um algoritmo rápido para extração de piso em tempo real, que não depende de nenhuma ordenação de sensor ou fusão de dados, aplicando ESSAC. Este capítulo apresenta detalhadamente o método proposto com análises mais extensivas sobre seus resultados de desempenho.

O trabalho de (DOUILLARD et al., 2011) provou experimentalmente que realizar a extração de piso antes da segmentação de objetos reduz dramaticamente o tempo de processamento sem afetar a acurácia de segmentação. Isso facilita a segmentação de objetos por melhorar sua separabilidade. Com base nisso, o objetivo do algoritmo proposto de segmentação é separar pontos de piso dos pontos de obstáculo.

O algoritmo consiste de uma projeção ortogonal sobre o plano horizontal, seguida de uma segmentação *top-down* por árvore quaternária. A redução de dimensionalidade beneficia o tempo de execução sem afetar a acurácia porque se pode supor que o piso é uma superfície unitária que não se sobrepõe. A segmentação se adapta à nuvem, focando esforço de processamento em áreas detalhadas e extrai superfícies de piso que não são necessariamente planas. A segmentação *top-down* por árvore quaternária segue a essência de Hampp e Bormann (2013), Park, Choi e Yu (2014), mas opera sobre uma projeção ortogonal ao invés de uma projeção perspectiva/cilíndrica.

Em contraste com o que é feito por Schnabel, Wahl e Klein (2007), a busca pelo plano é simplificada. Por ser apenas um passo preparatório, o número de planos candidatos pode ser dramaticamente reduzido, portanto não eleva significativamente o tempo de execução. O processo de estimação de plano foi adotado para determinar a direção da projeção ortogonal. A Seção 6.1 descreve o algoritmo e define seus parâmetros. A Seção 6.2 apresenta a abordagem de teste e os resultados para avaliar velocidade e acurácia.

6.1 Algoritmo proposto

O algoritmo consiste de dois processos que devem ser executados sequencialmente sobre os dados. O primeiro processo objetiva encontrar o plano que melhor aproxima o piso. Supondo que nenhum sensor externo ofereça orientação em relação ao piso, a estimativa do plano do piso permite rotacionar a nuvem tornando o piso paralelo ao plano horizontal

xy. Este passo é necessário para que os valores de z representem altura, que é utilizada nos passos seguintes. A Figura 26 apresenta uma nuvem de pontos antes e após a rotação. Note a câmera na origem do sistema de coordenadas. Antes da rotação, ela aponta para o eixo y. O método apresentado no Capítulo 5 é utilizado para este fim.

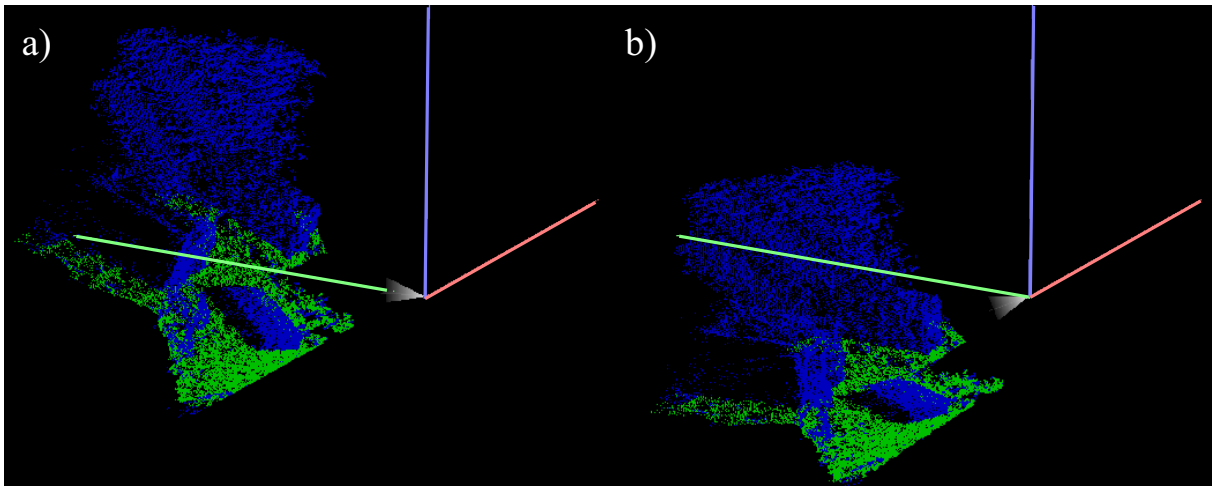

Figura 26 – Rotação da nuvem após a estimação do plano do piso. (a) Antes e (b) depois da rotação. O propósito da rotação é tornar o plano do piso paralelo ao plano xy do sistema de coordenadas. Os eixos x, y e z são representados em vermelho, verde e amarelo, respectivamente. O cone cinza representa a câmera.

O segundo processo consiste em dividir a nuvem de pontos em áreas retangulares sobre o plano usando uma árvore quaternária. O processo adaptativamente divide a nuvem em blocos de alturas similares. Isso permite que grandes áreas planas sejam divididas em grandes blocos, enquanto superfícies complexas são divididas em pequenos blocos. Ao fim deste processo de subdivisão de áreas, todos os blocos são avaliados e aqueles com pequenos espectros verticais são considerados como que contendo pontos de piso. As Subseções 6.1.1 e 6.1.2 detalham cada processo.

6.1.1 Detecção do plano do piso e rotação da nuvem

O primeiro passo do algoritmo localiza o plano que melhor aproxima o piso. A estimação deste plano permite que se alinhe o piso ao plano xy horizontal do eixo de coordenadas. Assim, a coordenada z de um ponto será igual à distância ao piso, o que é importante para próximo passo do algoritmo, que se baseia na altura de cada ponto em relação ao piso. Em outras palavras, a distância até o plano do piso exerce um papel central e por isso este plano deve ser aproximado.

Para a estimação do plano do piso, o método ESSAC é utilizado, como apresentado no Capítulo 5, sofrendo pequenas simplificações. A suposição é de que o plano que contém a maior parte dos pontos há de constituir a melhor estimativa para aproximação do piso.

Em notação ES, E(μ+λ) indica que há μ pais e λ filhos, cada indivíduo representados por $\vec{p} = \begin{bmatrix} a & b & c & d \end{bmatrix}$. As premissas a seguir foram escolhidas: a) E(μ+λ) com $\mu = 1$, ou seja: um pai em uma população de λ, em notação ES, com o foco em simplificar a implementação em dispositivos embarcados; b) Uso de uma função de densidade de probabilidade constante uniforme (do inglês, *probability density function* – PDF) para as operações de mutação e nenhum uso de cruzamento. Estas premissas garantem os benefícios da otimização não-linear ES com implementação simplificada. Para reduzir ao máximo a complexidade desta etapa, não foi utilizada coevolução.

No algoritmo proposto, a população de planos de $\lambda + 1$ indivíduos é

$$P = \{\vec{p}_1, \vec{p}_2, \ldots, \vec{p}_\lambda, \vec{p}_{\lambda+1}\}. \tag{6.1}$$

A operação de mutação também foi simplificada e é parametrizada por σ_n e σ_d de valor constante, pré-definido. Cada parâmetro define o espectro da PDF simétrica uniforme para operações de mutação sobre os componentes $\vec{v}_n$ e d de $\vec{p}$, respectivamente, como apresentado pela Figura 27. Esta operação é aplicada como um passo incremental de tamanho aleatório no intervalo $(-\sigma, \sigma)$. Mais uma vez, a adoção de PDFs uniformes ao invés de PDFs gaussianas visa simplificar ao máximo esta etapa do algoritmo.

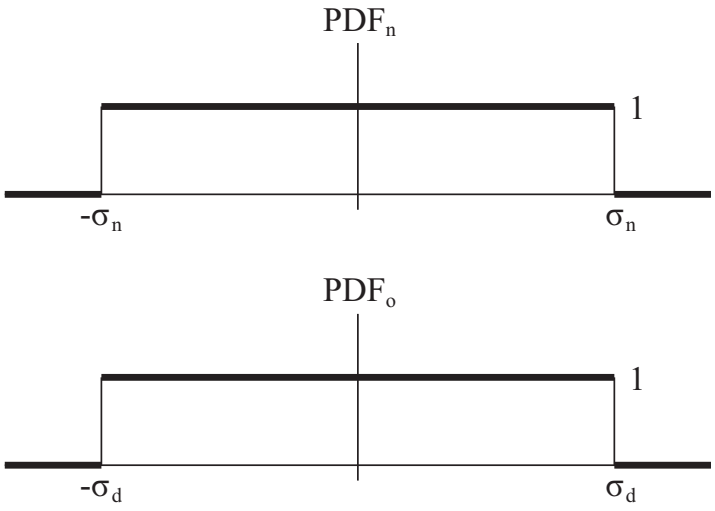

Figura 27 – PDFs uniformes usadas para operações de mutação.

O processo de geração de descendentes consiste em produzir λ cópias mutantes de $\vec{p}_1$, o pai. Tais cópias irão então substituir o restante da população, sobrando apenas $\vec{p}_1$. O processo de encontrar o indivíduo mais adequado de P avalia todos os planos $\lambda + 1$ em P e move o mais adequado para $\vec{p}_1$, fazendo dele o pai da próxima iteração. Tal seleção de adequabilidade é baseada apenas na função de aptidão clássica de RANSAC (Equação 5.5).

A centróide $\vec{c}$ de uma nuvem C é

$$\vec{c} = \begin{bmatrix} \frac{1}{n}\sum_{i=1}^{n} x_i \\ \frac{1}{n}\sum_{i=1}^{n} y_i \\ \frac{1}{n}\sum_{i=1}^{n} z_i \end{bmatrix}. \tag{6.2}$$

Isolando d na Equação 5.1 e considerando o ponto $\vec{v_o} = [\ x_0 \ \ y_0 \ \ z_0\ ]$ do plano

$$d = -ax_0 - by_0 - cz_0. \tag{6.3}$$

em natação vetorial,

$$d = -\vec{v_n} \cdot \vec{v_0}. \tag{6.4}$$

Onde $\vec{v}_n = [\ a \ \ b \ \ c\ ]$ é um vetor normal a $\vec{p}$. Portanto

$$\vec{p} = \begin{bmatrix} a \\ b \\ c \\ -\vec{v_n} \cdot \vec{v_0} \end{bmatrix}, \tag{6.5}$$

$$\vec{p} = \begin{bmatrix} \vec{v}_n \\ -\vec{v_n} \cdot \vec{v_0} \end{bmatrix}. \tag{6.6}$$

Da Equação 6.6, fazendo $\vec{v}_n = [\ 0 \ \ 0 \ \ 1\ ]$ e $\vec{v_0} = \vec{c}$, o plano $\vec{p}_1$ é inicializado como

$$\vec{p}_1 = \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ -\frac{1}{n}\sum_{i=1}^{n} z_i \end{bmatrix}. \tag{6.7}$$

Isso faz com que o plano inicial passe através da centróide $\vec{c}$ e seja ortogonal ao eixo z vertical. Para robótica móvel terrestre, esta é uma boa estimativa inicial porque o piso é geralmente horizontal e comumente passa pelo centróide da nuvem, ou próximo deste. Isto ocorre quando há grande quantidade de pontos de piso na nuvem, uma suposição razoável para a aplicação em questão.

O processo geral é descrito pelo Algoritmo 4. O laço finaliza sua execução quando a aptidão do melhor plano atual é a mesma que a do melhor plano anterior. Esta decisão de condição de parada foi feita empiricamente com base em uma variedade de conjuntos de dados de teste C, observando que resultados de convergência rápida (até dez gerações nos piores casos) foram obtidos com tal critério e geraram estimativas adequadas para o plano do piso. Quando este processo é completado, $\vec{p}_1$ contém um plano que aproxima bem os pontos do piso.

Data: Nuvem de pontos C
Result: Plano $\vec{p}_1$ que aproxima o piso.
1 Inicializar todos os elementos de P como na Equação 6.7;
2 **repeat**
3 | Calcular a aptidão de todos os $\lambda + 1$ indivíduos;
4 | Selecionar o melhor indivíduo como pai;
5 | Gerar λ cópias mutadas do pai;
6 | Substituir os λ velhos com as λ cópias;
7 **until** *AptidãoPaiAtual == AptidãoPaiAnterior*;

Algoritmo 4: ES para aproximação de plano do piso.

Em uma situação dinâmica, um robô deve interpretar uma série de nuvens de pontos como $C_1, C_2, \ldots, C_k, C_{k+1}, \ldots$, que são recebidas em tempo real de um dispositivo sensor. Para um conjunto abrangente de cenários, se pode supor que a posição do piso relativa ao robô, em uma nuvem de pontos C_k, varia apenas levemente da posição do piso da próxima nuvem obtida, C_{k+1}. Assim, ao invés de inicializar $\vec{p}_1$ como na Equação 6.7 para cada C_{k+1}, $\vec{p}_1$ de C_k pode ser usado. Isso permite uma convergência mais rápida para todos os casos onde C_{k+1} é muito semelhante a C_k.

Encontrar $\vec{p}_1$ permite que se rotacione a nuvem de pontos para tornar o piso perpendicular ao eixo vertical z do quadro de referência. Um ponto importante a enfatizar é que o objetivo ao estimar o plano é apenas de facilitar o próximo estágio do processo. Assim, não é necessário determinar minuciosamente este plano – basta ser uma boa estimativa inicial.

Uma vez que $\vec{p}_1$ é encontrado, cada ponto $\vec{q}$ em C será avaliado e aqueles dentro do limiar de distância D_g de $\vec{p}_1$ serão selecionados. Isto produz $C^{'}$, que é uma nuvem constituída pelos pontos próximo ao piso do plano. Estes são os pontos candidatos que serão passados para os próximos passos do algoritmo, dado que estão mais propensos a pertencer ao piso. Assim como t, D_g é definido empiricamente porque depende das qualidades gerais de dados com que se trabalha, tais como ruído e a irregularidade média do piso.

Usa-se então a normal de $\vec{p}_1$, $\vec{v}_n$, para rotacionar $C^{'}$, tornando o piso perpendicular ao eixo z. Esta rotação ocorre em torno de um eixo na direção de $\vec{u}$ por um ângulo θ, onde $\vec{u}$ é o unitário do vetor $\vec{u}^*$

$$\vec{u} = \frac{\vec{u}^*}{|\vec{u}^*|}, \tag{6.8}$$

e $\vec{u}^*$ é o produto vetorial entre $\vec{v}_n$ e a direção unitária $\vec{k}$

$$\vec{u}^* = \vec{v}_n \times \vec{k}, \tag{6.9}$$

θ é o ângulo entre $\vec{k}$ e $\vec{v}_n$. Supondo que $\vec{v}_n$ é unitário, $cos(\theta) = c$ e $sin(\theta) = \sqrt{a^2 + b^2}$. A

Figura 28 apresenta uma perspectiva geométrica do descrito acima. Sob tais suposições, tem-se que

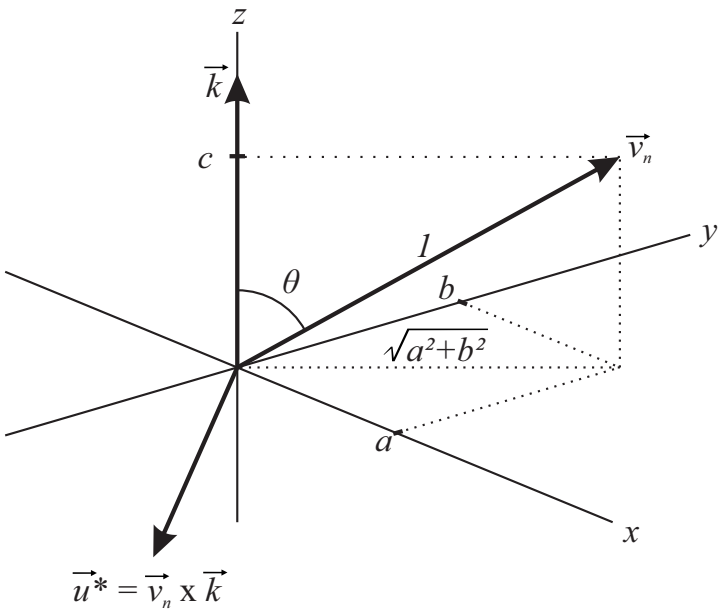

Figura 28 – Espaço de rotação.

$$\vec{u}^* = \begin{bmatrix} b \\ -a \\ 0 \end{bmatrix} \tag{6.10}$$

e

$$\vec{u} = \begin{bmatrix} \frac{b}{\sqrt{a^2+b^2}} \\ \frac{-a}{\sqrt{a^2+b^2}} \\ 0 \end{bmatrix}. \tag{6.11}$$

Da fórmula de rotação de Rodrigues (MURRAY; SASTRY; ZEXIANG, 1994), dado o eixo de rotação definido por $\vec{u}$ e o ângulo de rotação definido por θ, a matriz de rotação é

$$\begin{bmatrix} cos\theta + u_x^2(1-cos\theta) & u_x u_y(1-cos\theta) & u_y sin\theta \\ u_x u_y(1-cos\theta) & cos\theta + u_y^2(1-cos\theta) & -u_x sin\theta \\ -u_y sin\theta & u_x sin\theta & cos\theta \end{bmatrix}, \tag{6.12}$$

$$\begin{bmatrix} \frac{c+b^2(1-c)}{a^2+b^2} & \frac{-ab(1-c)}{a^2+b^2} & -a \\ \frac{-ab(1-c)}{a^2+b^2} & \frac{c+a^2(1-c)}{a^2+b^2} & -b \\ a & b & c \end{bmatrix}. \tag{6.13}$$

Que se pode simplificar para obter a matriz de rotação H

$$H = \begin{bmatrix} \frac{a^2c+b^2}{a^2+b^2} & \frac{ab(z-1)}{a^2+b^2} & -a \\ \frac{ab(z-1)}{a^2+b^2} & \frac{b^2z+a^2}{a^2+b^2} & -b \\ a & b & c \end{bmatrix}, \tag{6.14}$$

$$\vec{q_i}^* = H \cdot \vec{q_i}. \tag{6.15}$$

Aplicando H para todos os pontos $\vec{q_i}$ em $C^{'}$, os rotaciona e produz uma nova nuvem C^* de pontos $\vec{q_i}^*$. O plano do piso de C^* é então aproximadamente horizontal. Este primeiro processo de estimação do plano do piso e rotação da nuvem se conclui.

A finalização do próximo processo do algoritmo produz uma nuvem segmentada C^{S*}, que deve ser rotacionada de volta à sua posição original. Esta rotação pode ser feita ao aplicar a matriz de transformação inversa H^{-1}, que se iguala a H^T

$$H^{-1} = \begin{bmatrix} \frac{a^2c+b^2}{a^2+b^2} & \frac{ab(z-1)}{a^2+b^2} & a \\ \frac{ab(z-1)}{a^2+b^2} & \frac{b^2z+a^2}{a^2+b^2} & b \\ -a & -b & c \end{bmatrix}. \tag{6.16}$$

O número de operações deve escalar linearmente quando o número de pontos, n, é incrementado. Isso ocorre porque a densidade de pontos não afeta diretamente a convergência do plano aproximador e só é necessário um múltiplo de n operações para cada iteração de estimação do plano do piso.

6.1.2 Subdivisão adaptativa de área

A esta altura, todos os pontos de C que estão próximos do plano do piso foram selecionados e rotacionados. O plano do piso se tornou paralelo ao plano xy, de forma que a subdivisão de área pode ser iniciada. O propósito desta etapa final é separar os pontos que constituem uma forma aproximadamente horizontal dos pontos que constituem formas

mais complexas. Pontos são separados em blocos e as diferenças de altura dentro de cada bloco são avaliadas. Isso permite que um bloco que contém apenas pontos de piso seja facilmente diferenciado de um bloco que está obstruído. Exceto pelos efeitos do ruído nos dados e de irregularidades das superfícies, supõe-se que o bloco de piso seja geralmente plano, tendo comparativamente pouca variação vertical.

Para esta tarefa, o uso de uma árvore quaternária é proposto, ao invés de uma grade regular, para realizar uma subdivisão adaptativa de área. Dessa forma, o algoritmo deve focar em regiões complexas, tais como as representadas por áreas entre paredes e outros objetos, subdividindo-as de maneira adaptativa. Grandes áreas planas tais como um gramado ou uma rua vazia, devem ser mantidos intactos em grandes blocos.

Cada nó de árvore contém uma lista L de pontos de C^* e a forma retangular que os compreende. O processo de subdivisão, assim, consiste de uma descida recursiva prefixada da árvore. A Figura 29 apresenta uma árvore de exemplo.

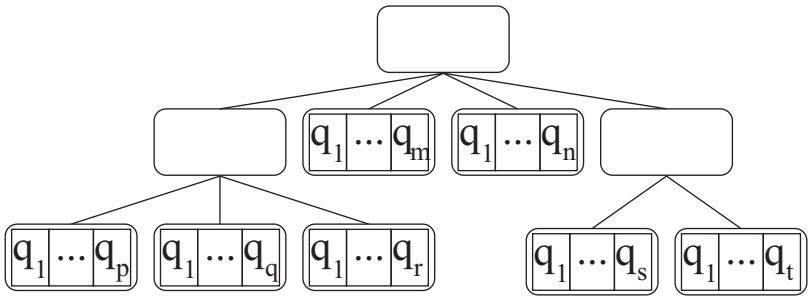

Figura 29 – Representação da árvore quaternária proposta. Cada nó folha contém uma lista de pontos q_i e um retângulo que contempla minimamente todos os pontos da lista.

O nó raiz é inicializado com todos os elementos de C^* listados em L. A cada visita de nó, o alcance vertical dos pontos em L é calculado como

$$v_r = |max(L^z) - min(L^z)| \,. \tag{6.17}$$

Onde L^z indica os valores z dos pontos em L. Devido ao alinhamento do plano do piso, os valores z de um ponto qualquer $\vec{q}$ representam verdadeiramente a distância entre $\vec{q}$ e o plano do piso. Considerando o caso onde o piso faz uma ladeira, sem este alinhamento, a premissa acima não seria válida e a acurácia do processo estaria ameaçada.

O critério para descida de um nó da árvore é que os lados de seu retângulo sejam ambos maiores que o limiar de resolução R_t e que $v_r > V_r$, onde V_r é um limiar de alcance vertical. O retângulo R é então dividido em quatro sub-retângulos, cada um tendo um quarto de sua área. Para cada sub-retângulo contendo mais de um ponto, um novo nó será instanciado e visitado recursivamente.

Uma vez que a condição de descida é alcançada e que um filho é instanciado, L é liberada para que não haja redundância na listagem dos pontos da nuvem entre nós da

árvore. Isso permite que apenas nós folha contenham listas de pontos L. O Algoritmo 5 apresenta a chamada recursiva de descida da árvore.

A métrica para diferenciar nós que contêm pontos de piso é derivada de v_r. Consiste da razão entre v_r e o comprimento diagonal do retângulo correspondente, tal que

$$v_g = \frac{v_r}{\sqrt{largura(R)^2 + altura(R)^2}}. \tag{6.18}$$

Esta métrica se traduz na relação entre alcance vertical e alcance horizontal dos pontos de um nó. Foi adotada empiricamente no lugar de v_r como um indicador de piso. Determinou-se, por resultados experimentais, que usar v_g como critério gera resultados ligeiramente mais acurados ao avaliar pertinência ao piso de blocos próximos a paredes e outros obstáculos.

Ao processar uma nuvem com n pontos, o número de nós na árvore será da ordem de $log(n)$. Este algoritmo envolve avaliar cada ponto de L para cada nó. L será subsequentemente dividida a cada descida. Assim, a ordem de complexidade do processo é $O(log^2(n))$. Adicionalmente, uma cópia da nuvem de pontos é mantida na memória durante todo o processo. Apenas nós folha mantêm referências aos pontos da nuvem e tais listas cobrem apenas os pontos que a si pertencem. Portanto, não haverá redundância ao listar os pontos e o uso de memória também será da ordem de $O(n)$.

Dados: Lista de pontos L e retângulo R.
Resultados: Subdivisão adaptativa da nuvem de pontos.
if $v_r > V_r$ *e* $perímetro(R) > R_t$ **then**
    Dividir R em R_1,R_2,R_3 e R_4;
    Transferir pontos de L para L_1,L_2,L_3 e L_4;
    if $tamanho(L_1) > 1$ **then**
        instanciar e descer sub-nó 1 com L_1 e R_1;
    if $tamanho(L_2) > 1$ **then**
        instanciar e descer sub-nó 2 com L_2 e R_2;
    if $tamanho(L_3) > 1$ **then**
        instanciar e descer sub-nó 3 com L_3 e R_3;
    if $tamanho(L_4) > 1$ **then**
        instanciar e descer sub-nó 4 com L_4 e R_4;
if *Nó é folha* **then**
    if $v_g < V_r$ **then**
        marcar todos os pontos em L como pontos de piso;
    else
        marcar todos os pontos em L como pontos de não piso;
    end

Algoritmo 5: Descida recursiva da árvore quaternária.

Note que não é necessário que a árvore seja completa nem cheia, de forma que os nós podem ter de zero a quatro filhos. Áreas com menos de dois pontos não serão cobertas por qualquer nó folha e o tempo de processamento não será gasto neles. Esta restrição existe porque ao menos dois pontos são necessários para calcular v_r corretamente.

R_t depende do quão esparsos são os dados e dos objetivos de acurácia. Este parâmetro pode ser usado para explorar o balanço entre requerimentos de acurácia e de tempo de execução para diferentes necessidades de projeto. O valor de V_r é responsável por diferenciar os nós de piso e é definido empiricamente.

Uma vez completada a descida de árvore, todos os pontos serão referenciados por nós folha, exceto aqueles poucos que não foram referenciados por serem os únicos pontos de seus nós. Tais pontos não serão considerados no processo, porém os resultados dos experimentos mostram que acurácia satisfatória ainda pode ser obtida sem eles.

Por fim, nós folha contendo $v_g < V_r$ são marcados como nós de piso e os nós folha que têm $v_g \geq V_r$ são marcados como nós de não-piso (obstáculos). O processo se completa e a rotação inversa é realizada sobre C^{S*} usando H^T para levar a nuvem para sua orientação original. O Algoritmo 6 sumariza o processo como um todo. A Figura 30 apresenta um resultado de exemplo de subdivisão de área de nuvem de pontos usando o Algoritmo 5.

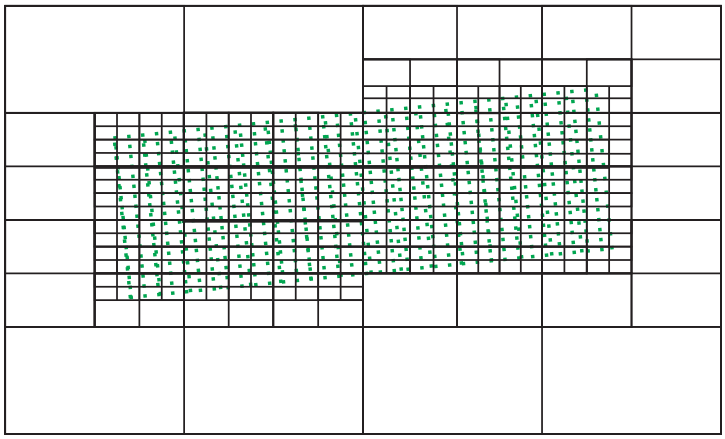

Figura 30 – Exemplo sintético de subdivisão de área usando árvore quaternária. Pontos verdes representam a nuvem de pontos tratada.

1 **Dados:** Nuvem de pontos C a ser segmentada.
2 **Resultados:** Nuvem de pontos segmentada C^s.
3 Encontrar o plano aproximador usando o Algoritmo 4;
4 Assegurar que $\vec{v}_n$ é unitário fazendo $\vec{v}_n = \frac{\vec{v}_n}{||\vec{v}_n||}$;
5 Selecionar os pontos dentro da distância D_g do plano usando a Equação 5.4, produzindo C';
6 Rotacionar C' usando H (Equação 6.14), produzindo C^*;
7 Encontrar o retângulo R que compreende todos os pontos de C^*;
8 Instanciar o nó raiz n_0;
9 Usar o Algoritmo 5 para descer $n_0(C^*, R)$, produzindo pontos segmentados C^{s*};
10 Rotacionar C^{s*} de volta usando H^T, produzindo C^s.

Algoritmo 6: Processo geral de segmentação de piso.

A ordem de complexidade geral do processo pode ser aproximada por $O(log^2(n)+n)$, que é comparável a $O(n)$. Assim, o mesmo escala bem, permitindo que grandes nuvens

Tabela 3 – Valores de parâmetros adotados

Parâmetro	Símbolo	Valor
Tamanho do passo para o vetor normal	σ_n	0,02 m*
Tamanho do passo para o coeficiente d	σ_d	1,00 m
Limiar de tolerância do plano	t	1,00 m
Distância máxima até o plano	D_g	1,00 m
Resolução da subdivisão	R_t	0,10 m
Limiar de classificação	V_r	0,20 m
Tamanho da população de planos	λ	10

*m - metros

sejam processadas eficientemente.

6.2 Procedimento experimental e resultados

O desempenho de tempo e de acurácia do algoritmo foi testado usando o conjunto de dados de nuvens de pontos disponibilizado por Munoz, Vandapel e Hebert (2009). Este conjunto de dados consiste de 17 nuvens de pontos, cada uma com aproximadamente cem mil pontos marcados. Os dados foram capturados usando escaneadores lidar em modo de varredura, acoplados a um carro de rua. Este conjunto de dados foi escolhido porque traz desafios ao algoritmo e representa um ambiente do mundo real rico em detalhes, contendo tanto ruas relativamente planas quanto rampas e ladeiras. Cobre uma área urbana com ruas, jardins, árvores, arbustos, postes, construções, cercas e carros.

6.2.1 Parâmetros do algoritmo

Os parâmetros do algoritmo foram definidos empiricamente, usando intuição e tentativa-e-erro para obter bons resultados. É importante ressaltar que diferentes grupos de valores resultaram em resultados igualmente bons e que os parâmetros aqui apresentados não são necessariamente os melhores para o conjunto de dados presente. A Tabela 3 apresenta os valores dos parâmetros usados para segmentar todas as 17 nuvens de pontos deste procedimento de teste.

O parâmetro σ_d define o tamanho máximo de passo para o componente d do plano candidato (indivíduo) $\vec{p}$ a cada geração. O coeficiente d do plano define uma translação linear no espaço, movendo o plano na direção de sua normal, $\vec{v}_n$. O parâmetro σ_d foi, portanto, escolhido ao observar a distância média que o plano inicial deveria ser deslocado de sua posição inicial, $\vec{c}$, para alcançar boa aproximação. Esta quantidade está relacionada à distância média entre os centróides das nuvens e a posição do piso relativa aos outros objetos nas cenas.

O parâmetro σ_n define o tamanho máximo do passo para os componentes a,b e c

de $\vec{p}$ a cada geração. Estes três componentes constituem $\vec{v}_n$ e representam a orientação do plano (o vetor normal ao plano). Um pequeno valor de σ_n foi adotado porque seriam necessárias apenas pequenas variações em $\vec{v}_n$ para alcançar convergência para o caso médio. Esta orientação está relacionada à variância de inclinação do piso encontrado nas cenas processadas. Cenas como aquelas apresentadas pelos dados do conjunto de teste têm pequenas variações angulares de inclinação do piso, da ordem de poucas dúzias de graus.

O parâmetro t define a tolerância inicial do plano. Para definir o valor deste parâmetro, é necessário entender este processo inicial, que consiste na otimização dos parâmetros do plano a,b,c e d, tal que o plano contenha a quantidade máxima de pontos dentro da tolerância t. A acurácia de interpretação de cenas de piso plano seria beneficiada por valores pequenos de t, enquanto cenas de terreno muito irregular se beneficiariam com valores maiores de t.

Definir D_g e V_r requer compreender o critério de reconhecimento do método. Este critério consiste de uma razão entre a altura e a largura do bloco, como descrito pela Equação 6.18. Assim, um bom valor para D_g e V_r depende dos níveis de ruído dos dados e dos níveis de irregularidades do piso da cena. Os parâmetros t, D_g e V_r foram definidos para contabilizar pela rugosidade média do piso de todas as cenas do conjunto de testes.

O parâmetro R_t define a resolução mínima para a subdivisão de área, limitando a descida da árvore. O valor de R_t foi então definido de acordo com o espaçamento médio entre os pontos da nuvem. Uma nuvem mais densa exige uma subdivisão de área mais densa para se alcançar boa acurácia, e assim um valor menor de R_t. Por outro lado, uma nuvem mais esparsa não se beneficiaria de uma resolução muito minuciosa, e um valor maior para R_t resultaria em um balanço custo/acurácia mais favorável.

λ é um parâmetro de otimização e depende pouco do conteúdo da cena. Este valor foi definido ao observar a evolução da função de aptidão dos planos em diferentes casos e ajustado para reduzir tempos de processamento e ainda obter boa acurácia.

Os valores escolhidos para os parâmetros são apropriados para o ambiente urbano em questão e não renderiam exatamente o mesmo desempenho em outros ambientes, tal como uma estrada rural muito acidentada. Entretanto, muitos robôs são construídos para operar sob restrições específicas. Um robô feito para navegar um campus universitário, como este apresentado pelas nuvens do conjunto de teste, teria um desempenho comparável.

6.2.2 Resultados e otimização de parâmetros

Alguns dos resultados obtidos ao executar o primeiro estágio do algoritmo, que consiste da estimação do plano do piso, são apresentados pela Figura 31. Na maior parte dos casos houve convergência bem-sucedida ao piso, enquanto em outros não, como demonstrado pelas nuvens C_{13} (c) e C_{14} (d). A nuvem C_{13} revela a maior fraqueza do algoritmo: requer

que o usuário explore o balanço entre acurácia e a variedade de irregularidades de piso ao definir os parâmetros σ_d, t, D_g e V_r.

Estes parâmetros podem ser otimizados para executar muito precisamente em uma região plana, restringindo seu desempenho a tais regiões. Alternativamente, podem ser definidos para suportar terrenos altamente desnivelados de forma mais robusta. Por o piso ser relativamente plano para a maior parte das nuvens neste conjunto de dados, os parâmetros foram otimizados para tal, deixando apenas alguns casos excepcionais de fora, tais como as rampas encontradas em C_{13} e C_{14}. Nestes casos, o plano aproximador não cobre todo o piso, reduzindo as taxas de acurácia.

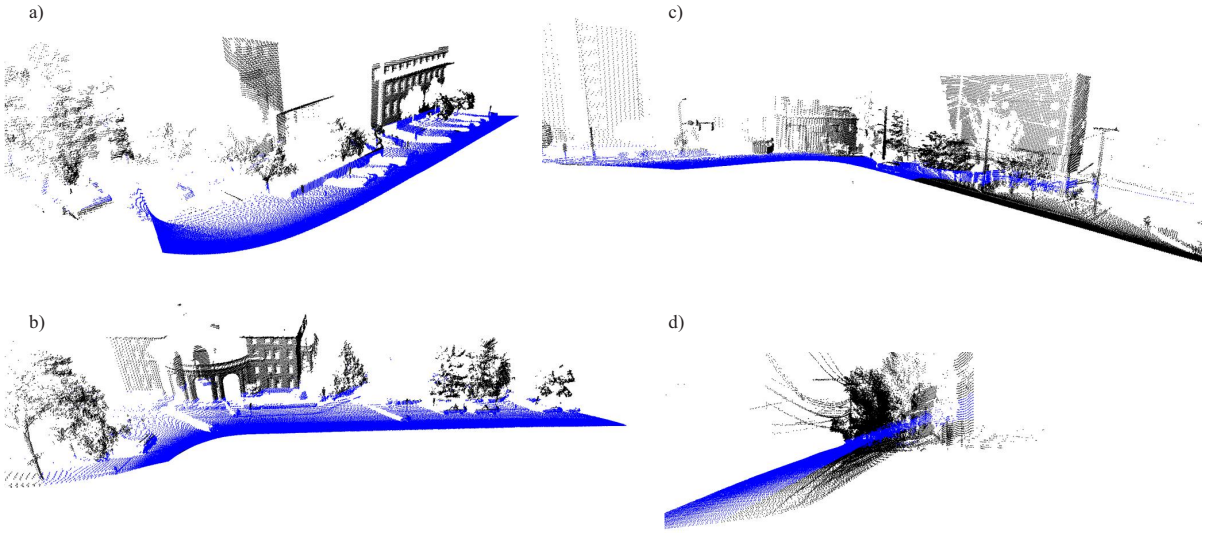

Figura 31 – Resultados de aproximação de plano para nuvens C_1 (a), C_3 (b), C_{13} (c) e C_{14} (d). Pontos em azul indicam pontos contemplados pelo plano aproximante.

6.2.3 Critérios de acurácia

Os marcadores disponíveis no conjunto de dados foram usados como *ground-truth* para avaliar a acurácia do algoritmo. Como o objetivo é o reconhecimento de pontos de piso e não-piso, as seguintes classes foram tratadas como piso: *piso*, *estrada pavimentada*, *trilha*, *meio-fio* e *calçada*.

Considerando que os pontos com tais classes são piso e que todos os demais pontos são não-piso, as falsos positivos f_p, falsos negativos f_n e estimativas corretas foram computadas. Assim, o número de pontos de não-piso marcados como piso (f_p) e o número de pontos de piso marcados como não-piso (f_n) pode ser computado. Se uma nuvem C

tem uma quantidade total de pontos n, a acurácia é calculada como

$$Acc = \frac{n - f_p - f_n}{n} \cdot 100\%. \tag{6.19}$$

A Figura 32 apresenta vistas perspectivas das nuvens de pontos e destaca os resultados da avaliação de acurácia, indicando pontos de piso corretamente marcados, pontos de não-piso corretamente marcados, falsos negativos e falsos positivos em cores diferentes. Estas vistas permitem apreciação qualitativa do desempenho do algoritmo sobre o conjunto de dados selecionado.

6.2.4 Execução e resultados de tempo

O código foi implementado usando a linguagem C++. Uma placa de microprocessador embarcado Texas Instruments Beaglebone Black foi adotada para os testes de avaliação de tempo de execução. Esta placa executa um sistema operacional Linux Debian e traz um microprocessador ARMv7 1000 Hz. Foi escolhida por ser atualmente uma das plataformas de processamento embarcado de mais baixo custo capaz de executar um sistema operacional completo. Comercializada no varejo internacional pelo valor de 55,00 dólares americanos, custando 175,60 reais considerando a cotação atual de 3,19 reais. O baixo custo, baixo consumo de energia e tamanho reduzido a fazem viável para robótica embarcada. É também uma boa opção para desafiar o desempenho do algoritmo proposto neste capítulo.

O tempo de execução foi mensurado para quatro diferentes passos do processo:

t_0**:** C cálculo da centróide ;

t_1**:** Estimação do plano do piso;

t_2**:** Transformação de rotação (ida e volta);

t_3**:** Subdivisão de área adaptativa e classificação de pontos.

O código foi executado 100 vezes para cada nuvem e a média de cada resultado foi computada. A Tabela 4 apresenta os resultados de média de acurácia com falsos positivos e falsos negativos, além da média do número de gerações para o processo de estimação de plano.

A Tabela 5 apresenta os resultados detalhados de média de tempo para cada um dos estágios do processo. As Figuras 33 a 36 mostram uma representação gráfica das nuvens marcadas resultantes, cobrindo todo o conjunto de teste.

A maior parte do tempo de execução foi tomado pelo processo de aproximação de plano, que se relaciona diretamente ao número de gerações ocorridas. Para diversas

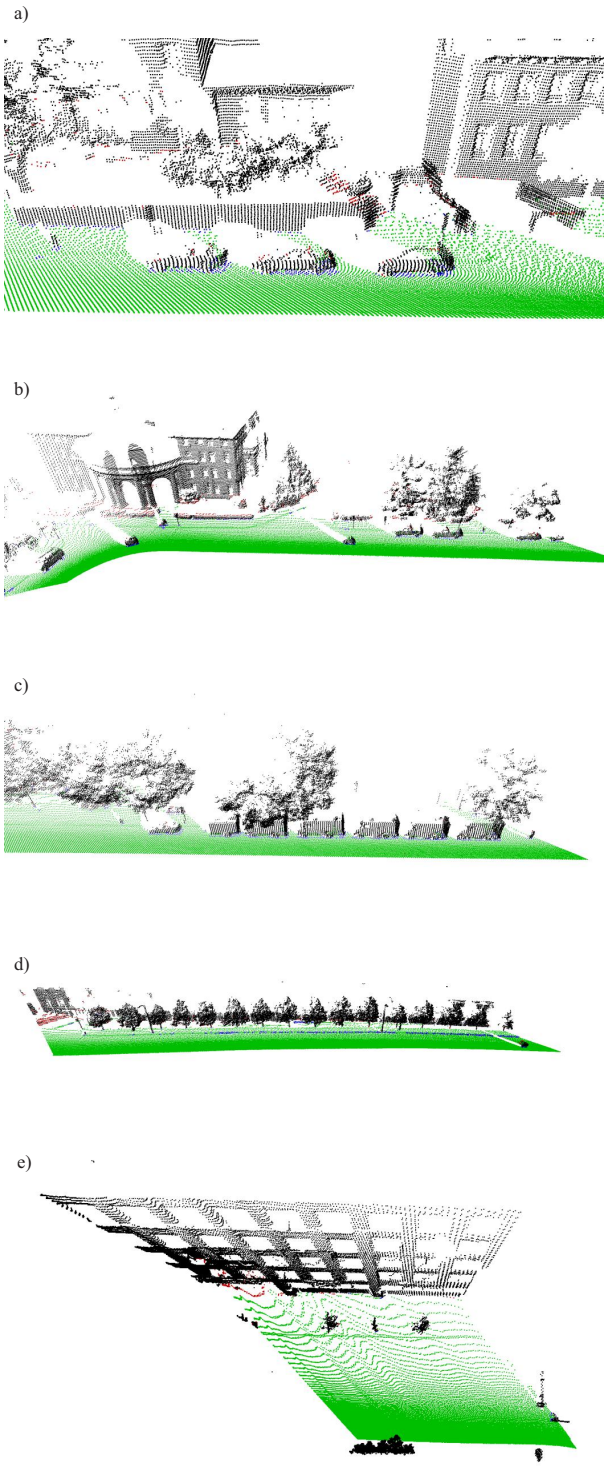

Figura 32 – Vistas perspectivas dos resultados de segmentação de piso para as nuvens C_1, C_3, C_4, C_8 e C_{16} – (a)-(e), respectivamente. *Verde:* pontos de piso corretamente marcados; *preto:* pontos de não-piso corretamente marcados; *vermelho:* falsos-positivos; e *azul:* falsos-negativos.

aplicações de visão computacional para robótica terrestre, pode-se supor que a orientação do robô em relação ao piso não será alterada dramaticamente de um quadro para outro. Como discutido na subseção 6.1.1, ao fazer tal suposição, $\vec{p}_1$ de C_k pode ser mantido como

Tabela 4 – Médias de acurácia e de número de gerações.

C	$f_p(\%)$	$f_n(\%)$	$Acc(\%)$	Ger.
1	0.69%	1.22%	98.09%	5.250
2	0.24%	1.99%	97.77%	3.370
3	0.79%	0.82%	98.39%	3.230
4	1.93%	0.70%	97.37%	2.650
5	2.32%	1.69%	96.00%	2.750
6	2.02%	1.01%	96.97%	2.430
7	1.43%	0.42%	98.16%	2.540
8	0.85%	1.00%	98.15%	2.120
9	1.06%	0.62%	98.32%	2.830
10	0.90%	1.38%	97.72%	2.220
11	0.69%	0.99%	98.32%	2.810
12	1.36%	2.60%	96.03%	2.820
13	1.83%	27.21%	70.96%	1.860
14	1.42%	12.50%	86.08%	3.250
15	1.62%	2.81%	95.57%	4.950
16	1.96%	0.10%	97.94%	2.710
17	0.96%	4.50%	94.54%	5.460
Média	1.30%	3.62%	95.08%	3.132

Tabela 5 – Resultados detalhados de tempo de execução. Valores em milissegundos

C	t_0	t_1	t_2	t_3	Total
1	33.6	1363.3	34.7	213.1	1645.2
2	33.9	879.6	32.5	158.2	1104.8
3	32.6	822.6	33.2	179.0	1068.1
4	34.3	673.0	32.2	168.2	908.3
5	33.6	706.8	32.3	168.0	941.3
6	33.0	602.5	30.1	154.8	821.0
7	33.1	630.0	30.6	139.9	834.3
8	34.0	518.6	21.3	174.9	749.6
9	33.5	704.3	30.3	148.4	916.9
10	33.3	543.4	26.0	176.3	779.6
11	33.4	714.6	33.6	173.8	956.0
12	30.4	653.5	28.9	167.6	880.8
13	33.7	454.8	12.7	137.2	638.7
14	33.0	822.8	27.2	101.4	984.9
15	32.7	1265.2	30.3	140.1	1468.7
16	28.2	573.4	27.3	130.1	759.6
17	14.5	584.3	12.2	51.5	662.8
Média	31.8	736.0	28.0	151.9	948.3

C_{k+1}, economizando iterações de aproximação de plano. Nestes experimentos, tal estratégia foi aplicada sobre 100 operações sequenciais de segmentação de piso para cada nuvem e um tempo médio geral de execução de 547.8 milissegundos foi obtido, uma redução de mais de 40% sobre a média de tempo apresentada pela Tabela 5. Isso mostra que em casos onde se

pode supor que ocorrerão apenas pequenas mudanças relativas à orientação do piso de um quadro para outro, o tempo de execução pode ser reduzido de forma significante.

6.3 Conclusões

Exceto por alguns casos especiais, tais como C_{13} e C_{14}, os resultados de acurácia se provaram satisfatórios para o conjunto de dados atual: a maior parte das nuvens foi segmentada com acurácia superior a 94%, algumas com mais de 97% e o pior caso apresentou $70,96\%$ de acurácia. Acurácia depende de configuração de parâmetros, características de sensor e ambiente representado pelas nuvens. O algoritmo pode ser configurado para se comportar acuradamente em uma ampla variedade de cenários. Entretanto, o conjunto de parâmetros que rende bons resultados para um ambiente não renderá necessariamente bons resultados para um outro ambiente. Desempenho para execução em tempo real em dispositivos embarcados de baixo custo é demonstrado através de resultados de testes quantitativos e qualitativos.

É importante ressaltar que o algoritmo foi desenvolvido e testado para nuvens de pontos obtidas a partir de escaneadores lidar, que apresentam grande acurácia, porém com custos elevados, equipamentos grandes, pesados e de elevado consumo de energia. Tais equipamentos são inviáveis para aplicações em robótica móvel de pequeno porte e baixo custo. Para tal, escaneadores mais baratos e menores são indicados, tais como aqueles que usam pares de câmeras estéreo. Nuvens de ponto estéreo são geralmente muito mais ruidosas e imprecisas. Para interpretar tais nuvens, são necessários métodos mais robustos. O Capítulo 7 apresenta uma adaptação deste algoritmo, adotando redes MLP, para melhorar a robustez da segmentação e reduzir o número de parâmetros necessários. A adoção de redes MLP permite o processamento bem-sucedido de nuvens de pontos estéreo obtidas por reconstrução em tempo real.

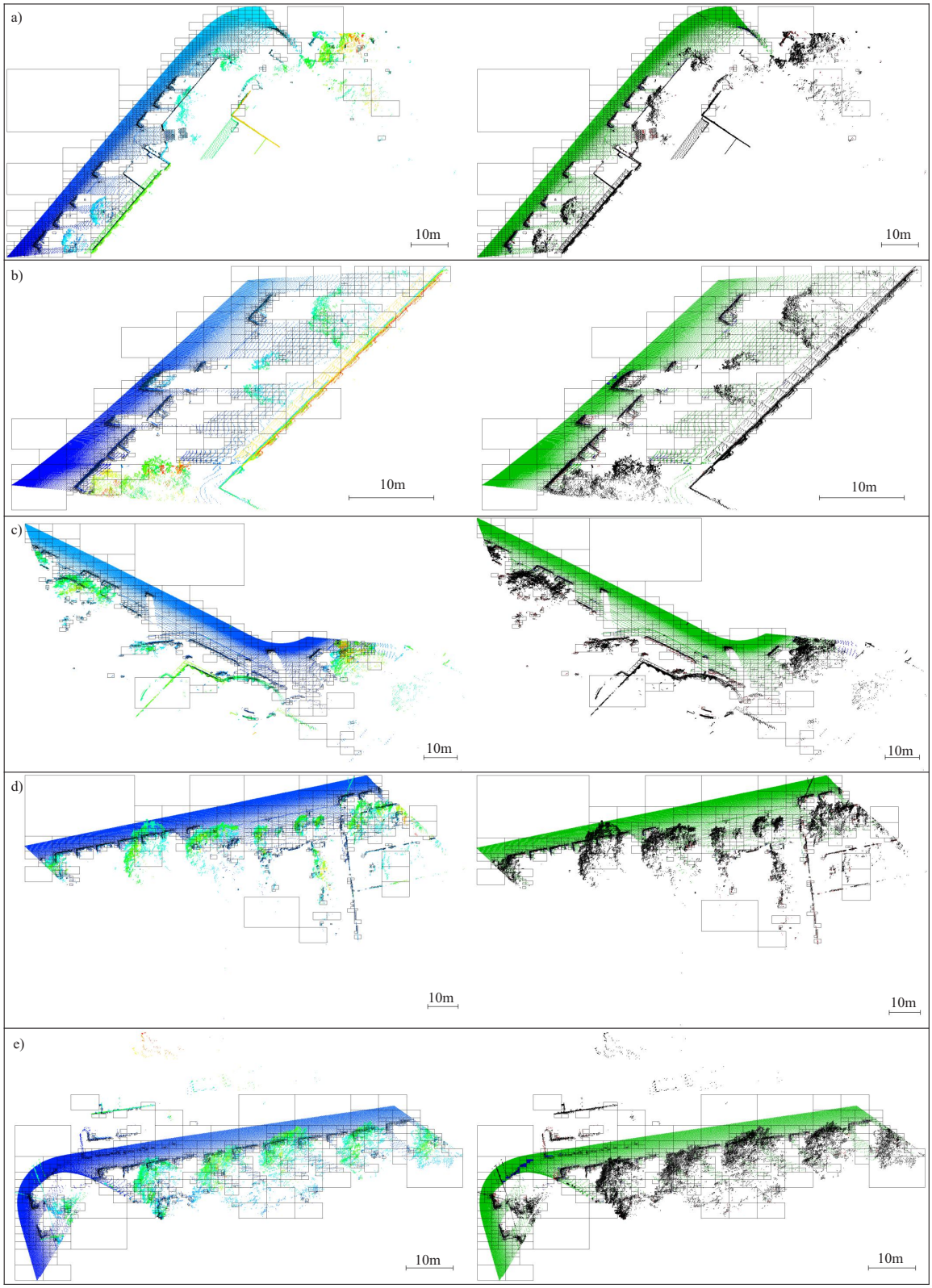

Figura 33 – Resultados de segmentação de piso para as nuvens C_1, C_2, C_3, C_4 e C_5 – (a)-(e), respectivamente. Projeção ortogonal *top-down* com sobreposição dos retângulos das folhas da árvore quaternária. Lado esquerdo: piscina de cores representando altura geométrica. Lado direito: *Verde:* pontos de piso corretamente marcados; *preto:* pontos de não-piso corretamente marcados; *vermelho:* falsos-positivos; e *azul:* falsos-negativos.

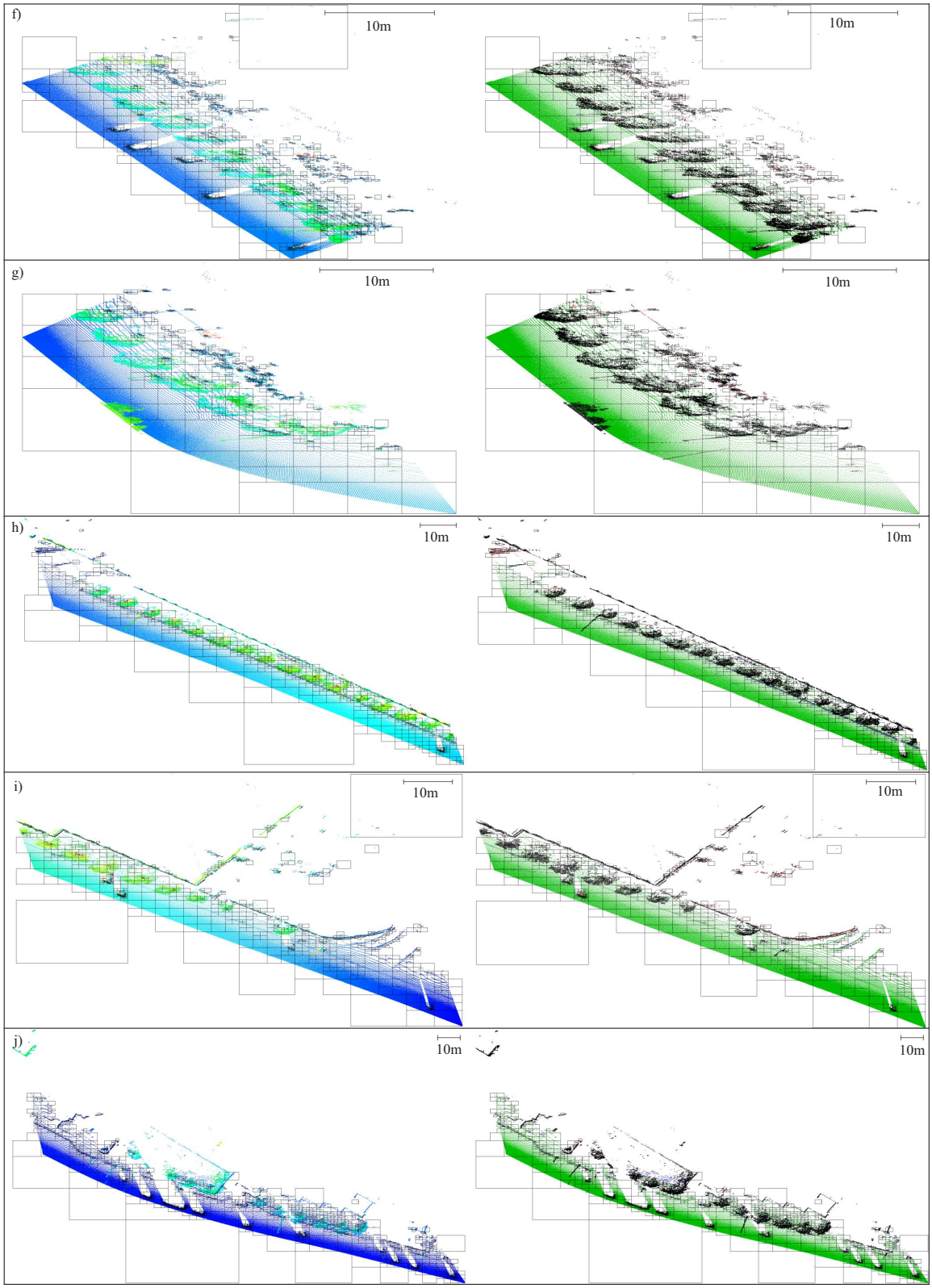

Figura 34 – Resultados de segmentação de piso para as nuvens C_6, C_7, C_8, C_9 e C_{10} – (f)-(j), respectivamente. Projeção ortogonal *top-down* com sobreposição dos retângulos das folhas da árvore quaternária. Lado esquerdo: piscina de cores representando altura geométrica. Lado direito: *Verde:* pontos de piso corretamente marcados; *preto:* pontos de não-piso corretamente marcados; *vermelho:* falsos-positivos; e *azul:* falsos-negativos.

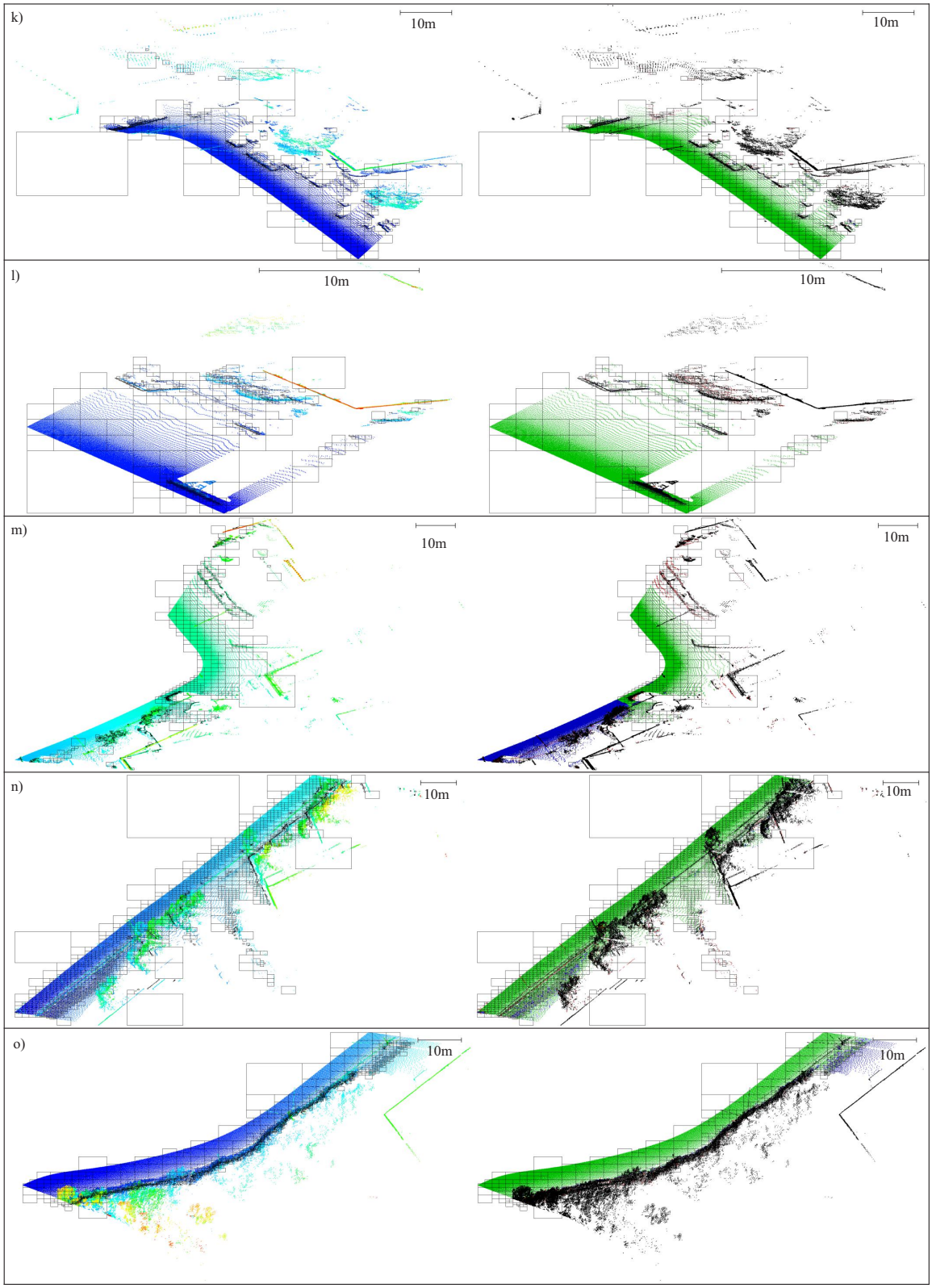

Figura 35 – Resultados de segmentação de piso para as nuvens C_{11}, C_{12}, C_{13}, C_{14} e C_{15} – (k)-(o), respectivamente. Projeção ortogonal *top-down* com sobreposição dos retângulos das folhas da árvore quaternária. Lado esquerdo: piscina de cores representando altura geométrica. Lado direito: *Verde:* pontos de piso corretamente marcados; *preto:* pontos de não-piso corretamente marcados; *vermelho:* falsos-positivos; e *azul:* falsos-negativos.

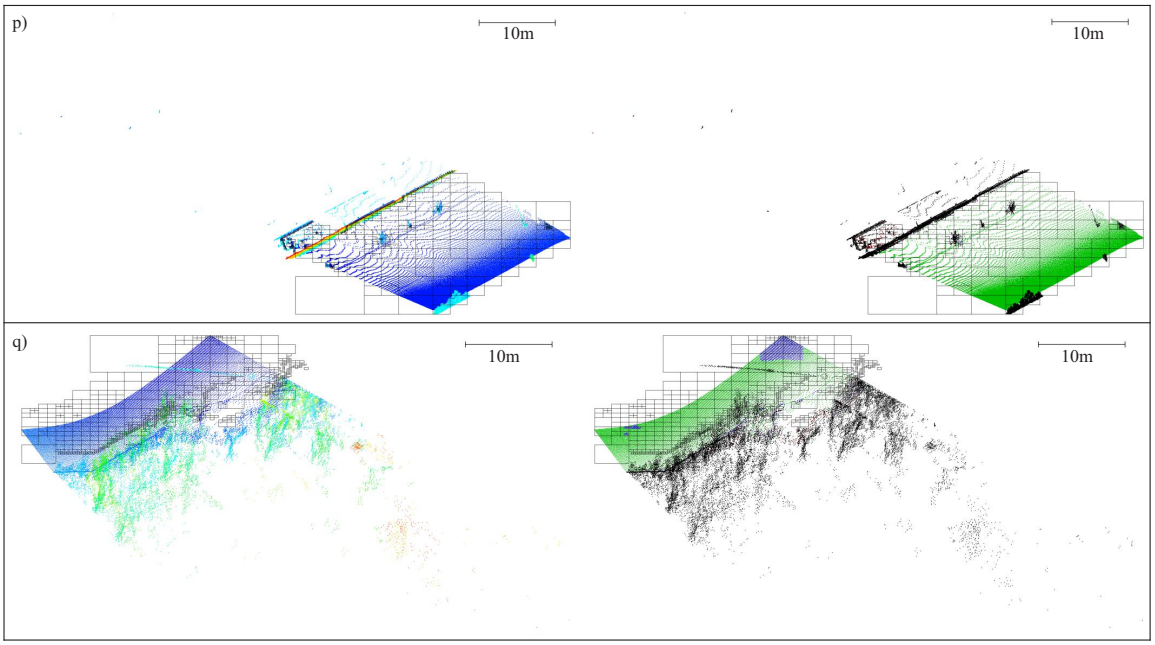

Figura 36 – Resultados de segmentação de piso para as nuvens C_{16} e C_{17} – (p)-(q), respectivamente. Projeção ortogonal *top-down* com sobreposição dos retângulos das folhas da árvore quaternária. Lado esquerdo: piscina de cores representando altura geométrica. Lado direito: *Verde:* pontos de piso corretamente marcados; *preto:* pontos de não-piso corretamente marcados; *vermelho:* falsos-positivos; e *azul:* falsos-negativos.

7 Aplicação de uma rede MLP para extração de piso em nuvens de pontos

Este capítulo apresenta a proposta de um algoritmo para extração rápida de piso que é capaz de operar sobre nuvens de pontos obtidas de câmeras estéreo de baixo custo para navegação e mapeamento em robótica móvel. Como uma melhoria sobre o método proposto pelo autor (MARCON et al., 2016) e apresentado no Capítulo 6, o processo também se beneficia de uma segmentação *top-down* por árvore quaternária. Em adição, um processo de classificação por MLP é utilizado, ao invés de limiarização, para determinar se o nó de uma árvore pertence ao piso ou a um obstáculo. O uso do MLP torna o processo mais robusto e mais apto a trabalhar sobre nuvens de pontos estéreo ruidosas. Sua natureza *top-down*, associada à classificação MLP, o torna um método robusto a dados omissos e ruído. Também não depende de informações de cor, fusão de dados, ordenação da estrutura de dados e nenhuma suposição sobre a ordenação da nuvem.

A contribuição desta proposta é um algoritmo rápido e robusto para extração de piso de nuvens de pontos estéreo não-organizadas obtidas por reconstrução em tempo real. O método também não depende de nenhum sensor externo ou suposição sobre pose. Consiste de uma segmentação adaptativa *top-down* sobre o plano de projeção ortogonal xy, seguida de uma classificação MLP. Supondo que o piso é uma superfície singular que não se sobrepõe, esta redução de dimensionalidade beneficia o desempenho em tempo de execução sem comprometer a acurácia. A subdivisão de área se adapta à nuvem, focando esforço de processamento em áreas detalhadas e melhorando a separabilidade do espaço de atributos da MLP.

7.1 Algoritmo proposto

A Figura 37 apresenta a sequência de passos do algoritmo proposto, atuando sobre a nuvem da Figura 1. O desvio padrão estimado para as medidas do sensor, σ_e, é usado como um limiar para a etapa de subdivisão. Para nuvens de pontos estéreo, o valor de σ_e pode ser definido como δZ (Equação 1).

O algoritmo consiste de quatro etapas: (1) estimação do piso do plano; (2) rotação da nuvem de pontos C para alinhar o plano do piso com o plano horizontal xy do sistema de coordenadas; (3) divisão adaptativa da nuvem usando uma árvore quaternária; e (4) classificação com MLP sobre cada nó da árvore, entre piso e não piso. Diferentemente do método proposto no Capítulo 6, o foco deste método é o classificador MLP. A etapa de estimação do piso do plano é realizada com o método MSAC semelhante ao utilizado por

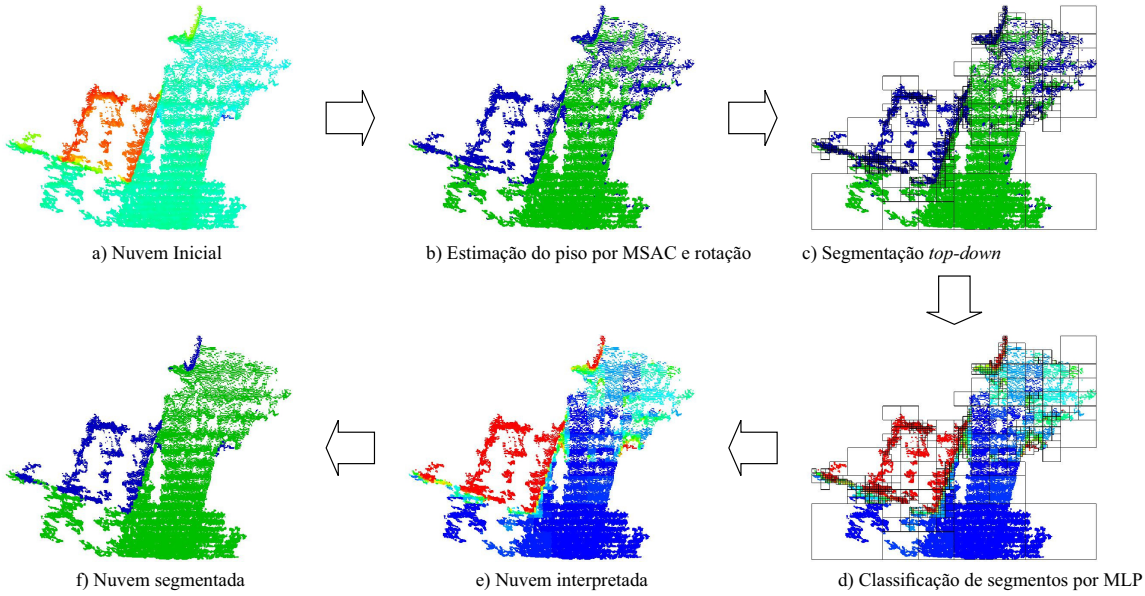

Figura 37 – Visão geral do algoritmo proposto. As nuvens representam caixas espalhadas pelo piso, com vista superior da projeção ortogonal. (a) Nuvem de pontos original, onde cor representa altura. (b) Resultados de MSAC para aproximar o plano do piso, seguido da rotação da nuvem para torna-la horizontal – verde representa pontos de piso. (c) Após a segmentação adaptativa por árvore quaternária. (d) Classificação dos nós da árvore pela MLP. (e) Resultados da classificação, onde cor representa certeza: vermelho 100% obstáculo e azul 100% piso. (f) Segmentação de piso usando limiarização sobre as saídas de certeza da MLP. Verde representa piso e azul representa obstáculos.

Konolige et al. (2009) por sua simplicidade, de forma a reduzir a análise ao uso da rede MLP. Cada passo é descrito em detalhes nas Subseções a seguir.

7.1.1 Estimação do plano do piso e rotação da nuvem

Como discutido extensivamente no Capítulo 5, RANSAC foi introduzido por Fischler e Bolles (1981) como um método de regressão robusta, mas, junto com outras variantes de SAC, foi aplicado com sucesso para a extração de planos de nuvens de pontos (SCHNABEL; WAHL; KLEIN, 2007; KONOLIGE et al., 2009; YANG; FÖRSTNER, 2010). Utilizado especificamente para segmentação de planos de piso de nuvens estéreo por Konolige et al. (2009), RANSAC encontra o plano do piso ao considerar pontos de obstáculo como *outliers* no conjunto de dados. Isso resulta da suposição de que os pontos do piso vão constituir de maneira aproximada o maior plano da cena, o que é uma suposição razoável para robótica móvel.

O RANSAC clássico tem uma condição de parada específica que supõe que a razão de *outliers* seja conhecida. Entretanto, ao segmentar nuvens de pontos, todos os pontos que não se enquadram ao modelo do plano são considerados *outliers*. Isso inclui todos os

outros objetos da cena cujo tamanho e quantidade são desconhecidos antes da segmentação. Assim, o teste clássico da taxa de *outliers* torna-se dificilmente válido e frequentemente leva a um número desnecessário de iterações. Nossa implementação segue do fluxo de trabalho do RANSAC clássico, mas ao invés de usar a clássica condição de parada, um número fixo foi gerado e avaliado, c, de candidatos, assim como proposto por Konolige et al. (2009). Adicionalmente, os candidatos são avaliados usando o estimador robusto MSAC de Torr e Zisserman (2000) porque este oferece um bom balanço entre custo e acurácia (Equação 5.9).

As nuvens são previamente decimadas para obter menor tempo de execução. A taxa de decimação, d, e o número de planos candidatos, c, são parâmetros definidos empiricamente. A Figura 38 apresenta uma análise de sensibilidade experimental para estes parâmetros, obtida com nuvens de pontos coletadas no laboratório. Note que d tem pouco efeito sobre os resultados da busca, mas tem grande impacto no tempo de execução. Valores crescentes de c geram resultados gradativamente melhores para $c \leq 1024$, mas saturam a partir desta faixa. O tempo de execução aumenta exponencialmente para valores grandes de c. Os parâmetros $d = 100$ e $c = 1024$ foram escolhidos por oferecer bom balanço entre tempo de execução e acurácia, segundo a análise de sensibilidade. Após a estimação do plano do piso, sua normal é utilizada para rotacionar a nuvem de pontos, como mostrado pela Figura 26.

7.1.2 Segmentação por árvore quaternária

Esta etapa é análoga à etapa de segmentação apresentada na Seção 6.1.2. Após a rotação, as coordenadas z de cada ponto representam sua *altura*. Subtraindo a altura do plano do piso obtém-se a distância aproximada até o piso, d_f. Cada nó da árvore contém um conjunto de pontos L da nuvem C e um retângulo que circunscreve os pontos de L, como na Figura 29; o nó raiz inicialmente lista todos os elementos da nuvem C. O desvio padrão da *altura*, σ_z, indica o quão plano (verticalmente homogêneo) é um conjunto de pontos:

$$\sigma_z = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(L_i^z - \overline{L^z})^2}. \tag{7.1}$$

onde L^z indica os valores z dos pontos de L. Diferente do Algoritmo 5, o critério de descida aqui envolve o desvio padrão σ_z e não apenas o alcance vertical (Equação 6.17). O uso do desvio padrão como critério de descida torna o método mais robusto a ruídos quando a quantidade de elementos por nó é elevada.

Executando uma segmentação *top-down* por árvore quaternária, isolam-se as áreas que contêm regiões amplas e planas das áreas que contêm estruturas verticais. Iniciando com um retângulo que engloba todos os pontos da nuvem C, o mesmo é divido recursivamente de acordo com σ_z. Nós com $\sigma_z > \sigma_e$ serão subdivididos em quatro filhos de dimensões iguais,

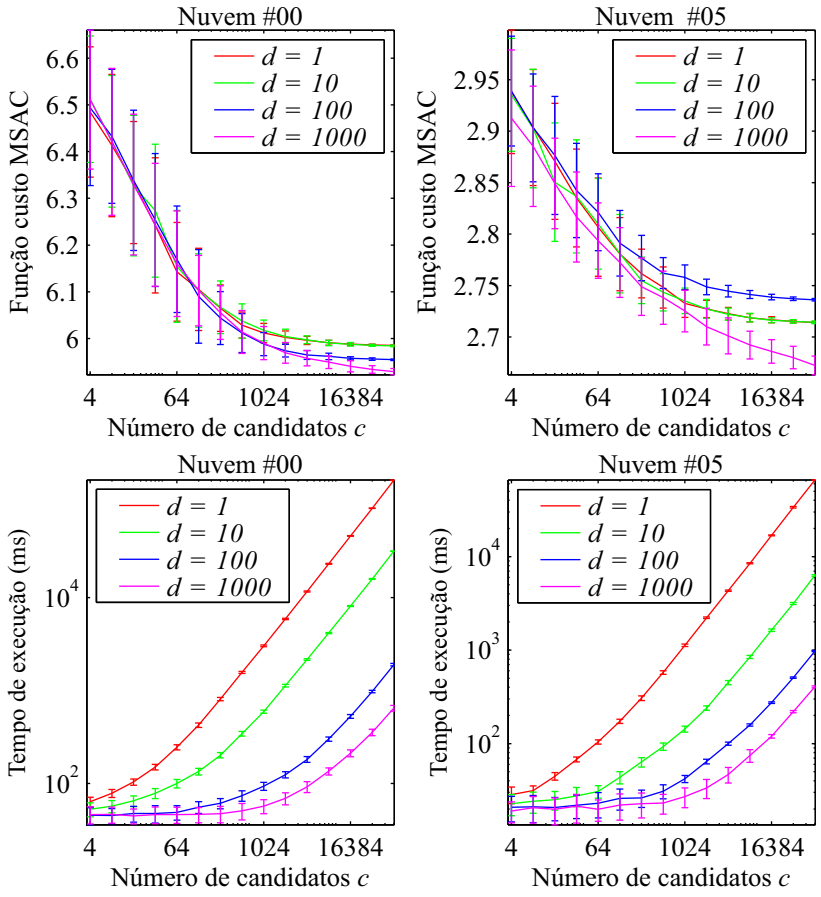

Figura 38 – Desempenho da estimação do plano do piso para nuvens selecionadas. Cada grupo de parâmetros foi executado cem vezes para obter significância estatística. Note que diferentes nuvens apresentam diferentes margens de de custo.

cada um cobrindo um quarto de sua área. Nós com σ_z reduzido não serão subdivididos porque têm grande potencial de conter majoritariamente pontos de piso. Nós com valores σ_z elevados continuam a subdividir para separar nós de piso e não-piso. Nós com menos de dois pontos ou perímetro menor que $4\sigma_e$ terão sua subdivisão encerrada; esta condição de parada evita segmentação desnecessária. O algoritmo, assim, foca em regiões complexas, tais como aquelas que contêm obstáculos, as subdividindo de maneira refinada. A Figura 39 apresenta um exemplo de subdivisão de árvore quaternária sobre uma nuvem de pontos obtida por reconstrução estéreo. Após a instanciação dos filhos, L é apagada para evitar redundância; assim, apenas nós folha vão conter listas de ponto L. O Algoritmo 7 mostra a chamada recursiva de descida da árvore. Ao invés de classificar os nós com um simples limiar, como no Algoritmo 5, a classificação envolve o uso de uma rede MLP e constitui a próxima etapa do método.

Dezesseis cenas complexas de objetos espalhados pelo piso foram usadas para oferecer casos típicos de navegação com obstáculos em ambientes internos. A Figura 40 (b) apresenta o número médio de nós de árvore obtidos da segmentação após mil execuções.

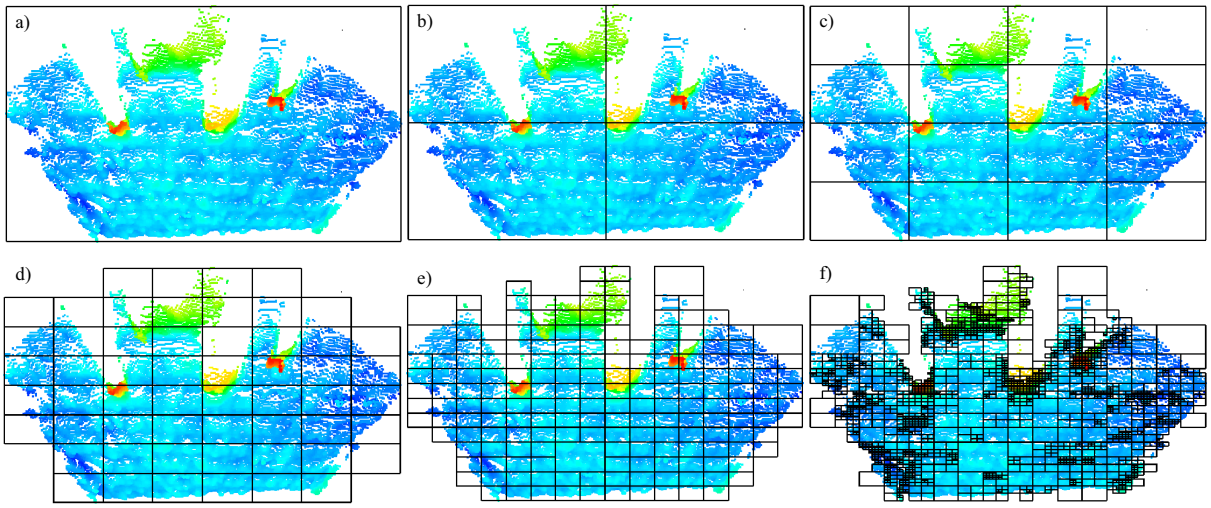

Figura 39 – Subdivisão da árvore quaternária passo-a-passo (a-f). A cor representa a altura: azul é mais baixo e vermelho é mais alto.

Dados: Lista de pontos L e retângulo R.
Resultados: Subdivisão adaptativa da nuvem de pontos.
if $\sigma_z > \sigma_e$ *e perímetro*$(R) > 4\sigma_e$ **then**
    Dividir R em R_1,R_2,R_3 e R_4;
    Transferir pontos de L para L_1,L_2,L_3 e L_4;
    if *tamanho*$(L_1) > 1$ **then**
        instanciar e descer sub-nó 1 com L_1 e R_1;
    if *tamanho*$(L_2) > 1$ **then**
        instanciar e descer sub-nó 2 com L_2 e R_2;
    if *tamanho*$(L_3) > 1$ **then**
        instanciar e descer sub-nó 3 com L_3 e R_3;
    if *tamanho*$(L_4) > 1$ **then**
        instanciar e descer sub-nó 4 com L_4 e R_4;

Algoritmo 7: Segmentação recursiva por árvore quaternária.

Note que a elevada estabilidade e repetibilidade do processo de subdivisão, demonstrado pelo baixo desvio no número de nós.

7.1.3 Classificação de segmentos

Após a subdivisão adaptativa de área, cada nó da árvore contém conjuntos de pontos que compartilham atributos similares. Uma rede neural artificial MLP é usada para classificar estes nós em entre piso e obstáculo. Três atributos são fornecidos para a MLP para este fim: desvio padrão da altura, σ_z; perímetro do retângulo, p; e distância aproximada até o piso, d_f.

O espaço de atributos e as amostras de treinamento são representadas na Figura 41. Este conjunto de treinamento foi obtido das nuvens #0 a #7 marcando manualmente cada

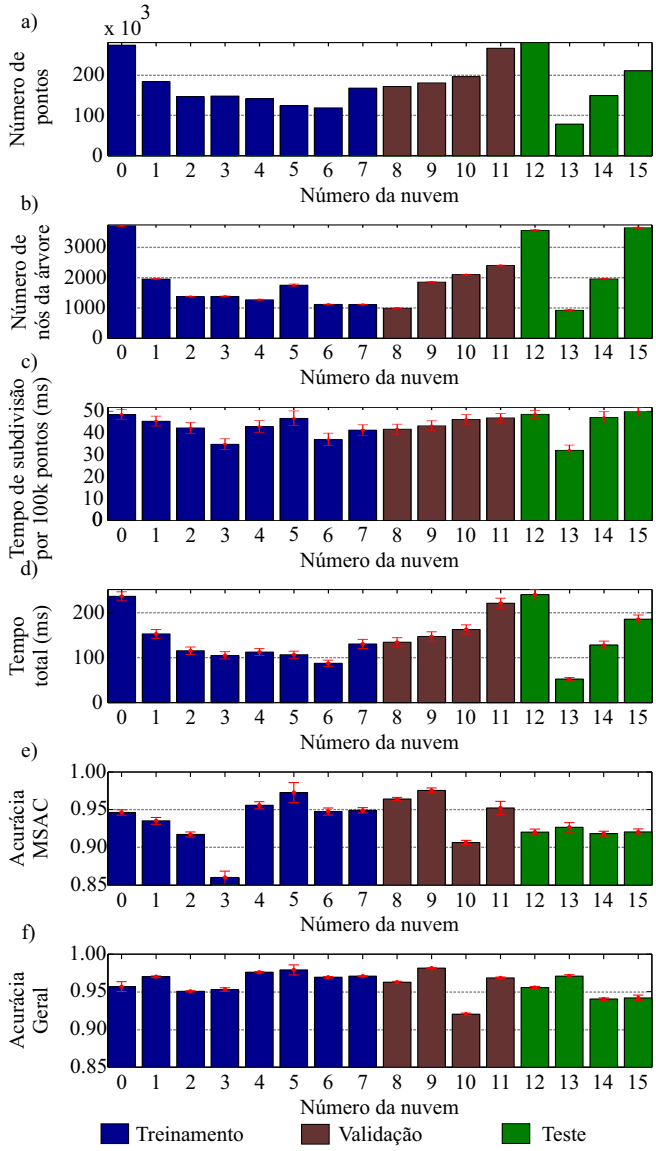

Figura 40 – Conjuntos de treinamento, validação e teste com resultados de tempo e acurácia.

ponto como piso e não-piso, como na Figura 1 (d). Note que o processo de subdivisão que precede a classificação por MLP elimina nós que sejam ambos grandes e verticalmente heterogêneos. Assim, as amostras de treinamento estão restritas às áreas onde ou p ou σ_z são próximas de zero.

Após a subdivisão quaternária, cada nuvem fornece centenas de amostras de treinamento. A razão piso/não-piso de cada nó é registrada como certeza de obstáculo para o grupo de treinamento, como mostrado pela Figura 42 (a). O algoritmo clássico de retro-propagação de erro é usado para o treinamento da MLP. Após o treinamento, a MLP gera resultados tais como os apresentados pela Figura 42 (b). Estas saídas de certeza podem ser limiarizadas ou usadas em processos de redução de incerteza e mapeamento. Ao estabelecer um limiar, pode-se obter diretamente uma segmentação final como na

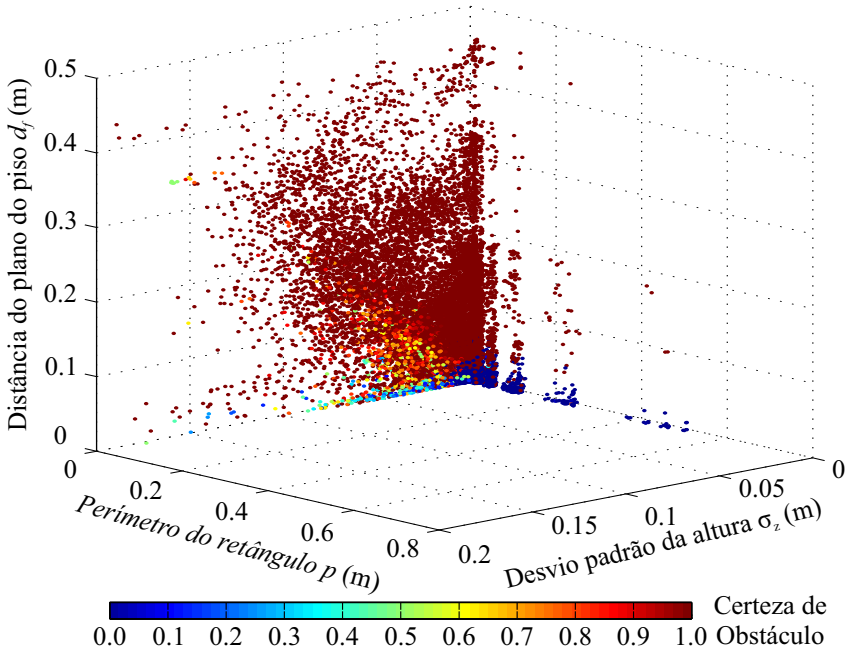

Figura 41 – Amostras de treinamento no espaço de atributos. Vermelho representa máxima certeza de obstáculo e azul representa máxima certeza de piso. Estes atributos permitem diferenciação acurada entre piso e obstáculo.

Figura 37 (f).

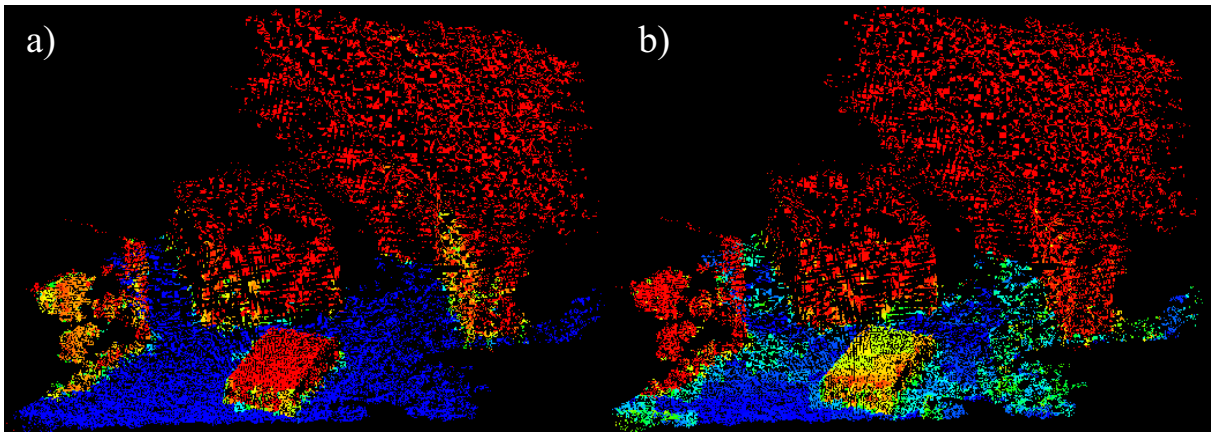

Figura 42 – Conjunto de treinamento MLP (a) e saía da MLP (b). Vermelho representa máxima certeza de obstáculo e azul representa máxima certeza de piso.

7.2 Plataforma para teste

O Code Laboratories DUO M é um par integrado de câmeras estéreo de amplo campo de visão, alta velocidade, compacto e configurável. Foi utilizado na coleta de

todas as nuvens de pontos utilizadas neste capítulo. A interface programação de aplicações integrada de *firmware* e *drivers* oferece uma interface encapsulada para captura de imagens, reconstrução estéreo e obtenção de nuvens de pontos. O dispositivo incorpora algoritmos de reconstrução estéreo em tempo real de alto desempenho, entre eles SBM e SGBM. Para obter melhor balanço entre qualidade e desempenho, adotou-se SGBM a uma resolução de 320x240. A Figura 1 mostra um exemplo de nuvem obtida usando esta configuração.

Para este par de câmeras, $b = 30mm$ e $f = 3mm$. A 320x240, o erro de disparidade esperado é de $\mu = 12um$. Limitando a profundidade a um metro – um alcance razoável para robótica de pequeno porte para ambientes internos – o erro de profundidade esperado é de, no máximo, 13 cm, calculado a partir da Equação 1.

Para avaliação de desempenho de execução em tempo real, uma plataforma de processamento embarcada de baixo custo foi adotada. A Nvidia Jetson TK1 é um computador embarcado CPU-GPU baseado na arquitetura ARM. Capaz de executar um sistema operacional completo, e ainda com consumo reduzido de energia, pode ser facilmente integrado a pequenos robôs móveis.

7.3 Validação e teste

As nuvens foram manualmente marcadas para oferecer amostras para treinamento e para avaliação de acurácia. O número de pontos e o número de nós obtidos pela segmentação por árvore quaternária de cada nuvem são apresentados pelas Figuras 40 (a) e 40 (b). Cada nuvem oferece em torno de dois mil nós; assim, o conjunto de treinamento consiste de aproximadamente dezesseis mil amostras (Figura 41).

O conjunto de validação é gerado a partir das nuvens de #8 a #11. O mesmo foi utilizado para análise de sensibilidade sobre o número de neurônios na camada interna, variando de 1-15. A Figura 43 apresenta os resultados de validação. Elevar o número de neurônios na camada interna impacta o tempo de classificação, mas não beneficia a acurácia. Isso se deve à separabilidade e à baixa dimensionalidade do espaço de atributos (Figura 41) resultantes do processo de subdivisão de área precedente. Ter vários neurônios na camada interna, para a aplicação em questão, não melhora claramente a acurácia da classificação. Assim, um neurônio na camada oculta foi adotado para a fase de testes porque oferece baixo tempo de computação e resultados de acurácia apreciáveis.

Por nenhum outro algoritmo na literatura operar sob as mesmas restrições, não foi possível oferecer uma comparação de desempenho justa de um-para-um, exceto para a abordagem SAC proposta por Konolige et al. (2009). A Figura 40 apresenta o tempo de subdivisão de área (c), o tempo total de processamento (d) e a acurácia final (f) ao executar na plataforma embarcada TK1. A acurácia foi calculada através da limiarização das saídas da MLP, comparando as saídas limiarizadas aos dados marcados manualmente.

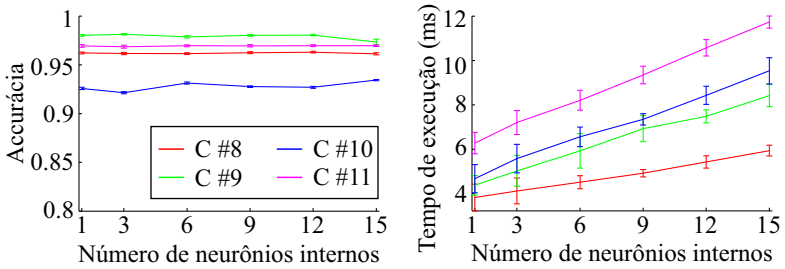

Figura 43 – Acurácia e tempo de classificação para vários números de neurônios na camada interna.

O limiar foi definido com valor $0,5$ – saídas menores que $0,5$ são consideradas como piso. Todos os testes registraram acurácia superior a 90%, um bom resultado para as nuvens ruidosas utilizadas. Os curtos tempos de execução são compatíveis com aplicações em tempo real. Adicionalmente, os núcleos GPU disponíveis nesta placa não foram utilizados em nossa implementação. Tanto a busca SAC quando a segmentação são adequados para a paralelização; assim, o desempenho pode ser melhorado usando tais núcleos. O tempo de execução está diretamente relacionado ao tamanho das nuvens de pontos, mas a acurácia depende de fatores específicos à cena.

A Figura 44 apresenta graficamente os resultados de classificação para o conjunto de testes (nuvens #12 a #15) através de vistas perspectivas. Note que a maior parte dos erros de classificação ocorrem em áreas de transição de piso para obstáculo. Devido ao ruído, estas áreas 'cinzas' são difíceis de classificar, mesmo para classificadores humanos dotados de informações prévias sobre a cena.

O método SAC para extração de piso proposto por Konolige et al. (2009) constitui um passo do algoritmo proposto (subseção 7.1.1); entretanto, usamos uma função de custo RANSAC melhorada. Para comparação, a acurácia foi avaliada para o passo de MSAC isoladamente (Figura 40 (e)), seguindo o mesmo procedimento usado para avaliação geral de acurácia. O método completo (com subdivisão quaternária e MLP) apresenta uma clara vantagem em acurácia em todos os testes. As Figuras 37 (b) e (f) demonstram geometricamente as diferenças entre os resultados dos dois métodos de segmentação. MSAC não se adapta a detalhes locais, deixando várias lacunas sobre o piso detectado, constituídas por pontos ligeiramente acima ou abaixo do plano aproximador. A subdivisão quaternária com classificação MLP é um refinamento necessário para a segmentação de piso por compensar irregularidades no piso e compensar ruídos e distorções do processo de reconstrução estéreo de tempo real.

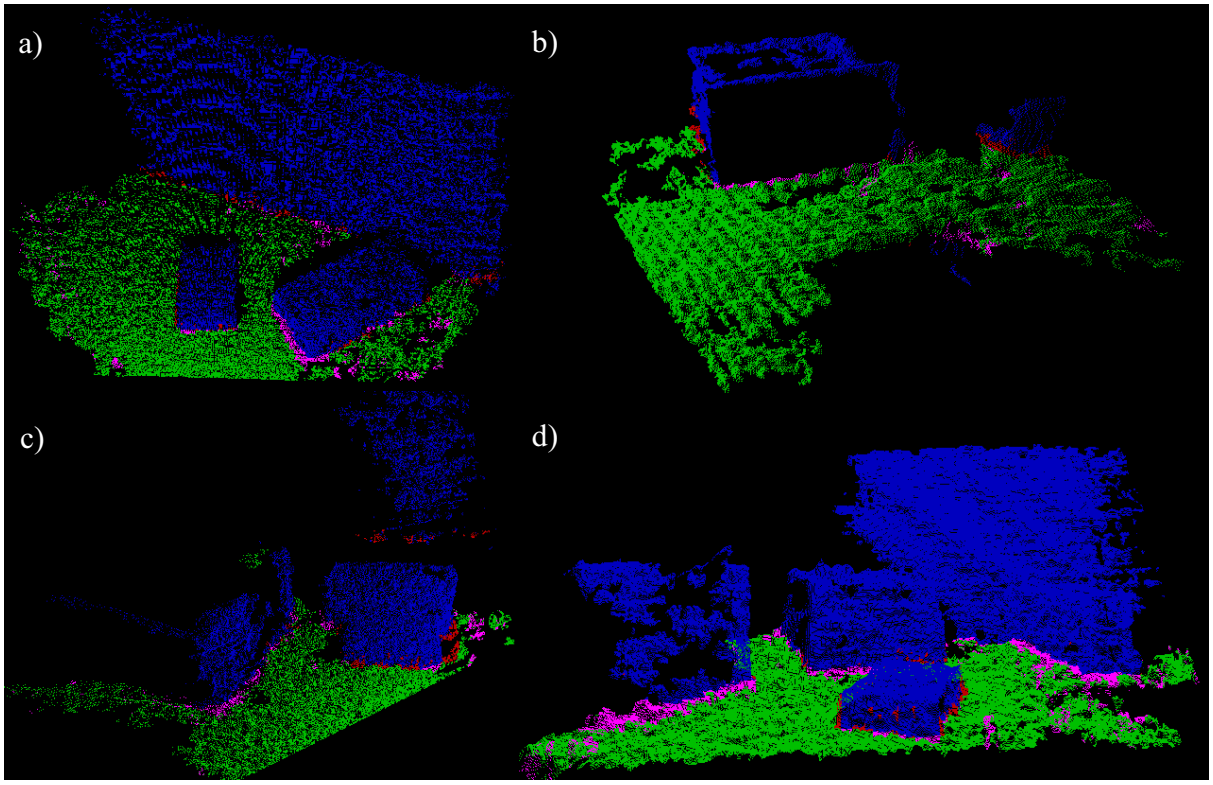

Figura 44 – Resultados finais para o conjunto de teste: nuvens #12 a #15 (a)-(d). Verde e azul representam piso e não-piso corretamente classificados, respectivamente. Rosa e vermelho representam piso e não piso classificados de maneira incorreta, respectivamente.

7.4 Conclusões

Este capítulo apresenta um algoritmo rápido para segmentação de nuvens de pontos estéreo que classifica pontos entre aqueles que pertencem ao piso e aqueles que pertencem a obstáculos, oferecendo valores de certeza. A acurácia é avaliada quantitativamente e os casos de erro são apresentados graficamente. A maior parte dos erros ocorre em áreas onde mesmo o classificador humano cauteloso há de apresentar baixo grau de certeza.

O desempenho se mostra apropriado para execução em tempo real em sistemas embarcados de baixo custo. O método proposto oferece uma boa solução para projetos de robótica de pequeno porte e de orçamento reduzido que não podem acomodar escaneadores ativos grandes e caros.

8 Conclusão

Este trabalho detalha três contribuições ao campo de processamento de nuvens de pontos. Os três métodos propostos são frutos de investigação de sensoriamento tridimensional de baixo custo para robótica autônoma. Os métodos são fortemente relacionados e seu desenvolvimento ocorreu na forma de sucessivas iterações a partir do estudo da literatura e de observações empíricas e experimentos. Ainda que tendo em vista a detecção de obstáculos para navegação autônoma, sua aplicabilidade tem escopo mais amplo, especialmente para o método apresentado pelo Capítulo 5. Diferentemente da maioria das abordagens de segmentação de nuvens de pontos, os algoritmos propostos não dependem de estruturas de imagens de profundidade 2,5D, nem de nenhuma informação de sensores auxiliares. Todos os processos envolvidos consideram apenas a geometria e não dependem de nenhuma informação de cor ou de refletividade.

O Capítulo 5 apresenta um novo método para extração de formas planas de nuvens de pontos. O método envolve ESSAC – RANSAC com estratégias evolucionárias – para reduzir o número de parâmetros e suposições e melhorar a razão custo/acurácia oferecida por outros métodos. Implementando coevolução adaptativa, o método proposto não requer nenhuma suposição sobre as cenas a processar. Uma análise de sensibilidade completa é apresentada para os parâmetros exigidos, mostrando o quão robusta é a abordagem a variações de parâmetros e a diferentes níveis de ruído. Nossa implementação *naive* supera em tempo de execução métodos estado-da-arte disponíveis em bibliotecas padrão industrial, tal como a PCL. Os resultados mostram que o método proposto produz melhor razão custo/acurácia que os métodos-patamar quando considerado o tempo de processamento. Este atributo o faz um detector de planos efetivo com grande potencial para representação de estruturas e partes mecânicas, não dependendo de ordenação de pontos nem de nenhuma organização prévia dos mesmos.

O Capítulo 6 propõe um algoritmo rápido para segmentação de piso em nuvens de pontos. O algoritmo diferencia entre pontos de piso e pontos de obstáculo em nuvens de pontos não estruturadas. Tal segmentação consiste em executar uma projeção ortogonal sobre o plano horizontal xy e então uma subdivisão de área *top-down*. Este processo se adapta à nuvem de pontos, focando esforço de processamento em áreas complexas.

Os tempos de execução mostram a viabilidade do algoritmo para implementações em tempo real em dispositivos embarcados, permitindo uma taxa de visão de 1 Hz. Esta taxa já é o suficiente para uma variedade de aplicações que usam sensoriamento via visão estéreo, e.g., robôs lentos, cortadores de grama, coletores, etc. Pode-se esperar a obtenção de desempenho mais elevado com outros sistemas embarcados de prateleira, tais os que

integram unidades de processamento gráfico (do inglês, *graphics processing units* – GPU) ou arranjos de portas programáveis por campo (do inglês, *field-programmable gate arrays* – FPGA). Tais tecnologias poderiam dramaticamente acelerar a execução do algoritmo ao explorar sua natureza paralela. O desempenho de outros paradigmas de algoritmos evolucionários para esta tarefa possibilita áreas para investigação futura.

O Capítulo 7 apresenta uma variação do método do Capítulo 6 otimizada para maior robustez no reconhecimento de piso em nuvens de pontos não estruturadas obtidas por reconstrução estéreo. O uso da rede MLP eleva sua tolerância a ruído e a dados faltosos, além de tornar o algoritmo menos sensível e dependente de parâmetros definidos pelo usuário. Propõe-se aplicar uma segmentação *top-down* quaternária seguida de uma classificação por rede MLP sobre cada semento. Esta abordagem adaptativa permite a diferenciação acurada entre pontos de piso e pontos de obstáculo em nuvens de pontos ruidosas de cenas complexas. Desempenho de tempo real é obtido em uma plataforma embarcada de baixo custo.

O algoritmo proposto permite extração rápida de pontos de piso a partir de nuvens de pontos não estruturadas sem depender de quaisquer especificidades de sensor nem qualquer ordenamento de dados. Isto permite sensoriamento por visão estéreo para contornar obstáculos e planejar rotas em projetos de robótica de baixo custo. Projetos de *swarm* de robôs também podem ser beneficiados porque o preço unitário é um grande fator limitante para o tamanho do *swarm*. Futuros trabalhos podem focar em explorar a natureza paralela do algoritmo, tal como a busca SAC ou a mesmo a descida da árvore quaternária, para minimizar seu tempo de execução, além de integrar o algoritmo a um *framework* de localização e mapeamento.

Uma vasta gama de estudos pode se dar como forma de continuação deste estudo. A análise experimental de outros métodos de computação flexível sobre os *frameworks* aqui propostos tem o potencial de gerar métodos ainda mais eficientes e robustos para segmentação de nuvens de pontos e detecção de obstáculos. O estudo de outros métodos de otimização como substituto de ES para detecção de formas planas ainda carece de investigação e experimentação. O uso de outros classificadores no lugar de MLP, tais como máquinas de suporte vetorial também pode render resultados mais interessantes para detecção de piso e de obstáculos em nuvens estéreo. Por fim, a combinação de outros métodos computacionais, tais como a agregação de abordagens *bottom-up* à segmentação por árvores quaternárias pode gerar algoritmos híbridos com um balanço mais atrativo entre custo e acurácia.

Referências

ABAYOWA, B. O.; YILMAZ, A.; HARDIE, R. C. Automatic registration of optical aerial imagery to a lidar point cloud for generation of city models. *{ISPRS} Journal of Photogrammetry and Remote Sensing*, v. 106, p. 68 – 81, 2015. ISSN 0924-2716. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0924271615001434>. Citado na página 13.

BAUER, J. et al. Segmentation of building models from dense 3d point-clouds. In: *In 27th Workshop of the Austrian Association for Pattern Recognition*. [S.l.: s.n.], 2003. p. 253–259. Citado 2 vezes nas páginas 14 e 51.

BAZARGANI, M.; MATEUS, L.; LOJA, M. Planar surfaces recognition in 3d point cloud using a real-coded multistage genetic algorithm. In: MORA, A. M.; SQUILLERO, G. (Ed.). *Applications of Evolutionary Computation*. Springer International Publishing, 2015, (Lecture Notes in Computer Science, v. 9028). p. 529–540. ISBN 978-3-319-16548-6. Disponível em: <http://dx.doi.org/10.1007/978-3-319-16549-3_43>. Citado 8 vezes nas páginas 5, 6, 51, 52, 53, 55, 60 e 70.

BEHLEY, J.; STEINHAGE, V.; CREMERS, A. Efficient radius neighbor search in three-dimensional point clouds. In: *Robotics and Automation (ICRA), 2015 IEEE International Conference on*. [S.l.: s.n.], 2015. p. 3625–3630. Citado 2 vezes nas páginas 16 e 49.

BEWLEY, A.; UPCROFT, B. Advantages of exploiting projection structure for segmenting dense 3d point clouds. In: KATUPITIYA, J.; GUIVANT, J.; EATON, R. (Ed.). *Australasian Conference on Robotics and Automation (ACRA2013)*. University of New South Wales, Sydney, NSW: Australian Robotics & Automation Association, 2013. p. 1–8. Disponível em: <http://eprints.qut.edu.au/66623/>. Citado 2 vezes nas páginas 15 e 50.

BORRMANN, D. et al. The 3d hough transform for plane detection in point clouds: A review and a new accumulator design. *3D Research*, Springer, v. 2, n. 2, p. 1–13, 2011. Citado na página 53.

BORRMANN, D. et al. The project thermalmapper – thermal 3d mapping of indoor environments for saving energy. *Proceedings of the 10th International IFAC Symposium on Robot Control (SYROCO)*, v. 10, p. 1, 2012. Citado 3 vezes nas páginas 65, 70 e 71.

CHANG, C.; CHATTERJEE, S. Quantization error analysis in stereo vision. In: IEEE. *Signals, Systems and Computers, 1992. 1992 Conference Record of The Twenty-Sixth Asilomar Conference on*. [S.l.], 1992. p. 1037–1041. Citado na página 14.

CHEN, J.; CHEN, B. Architectural modeling from sparsely scanned range data. *International Journal of Computer Vision*, Springer, v. 78, n. 2-3, p. 223–236, 2008. Citado 2 vezes nas páginas 14 e 51.

CHEN, T. et al. Gaussian-process-based real-time ground segmentation for autonomous land vehicles. *Journal of Intelligent & Robotic Systems*, Springer, v. 76, n. 3-4, p. 563–582, 2014. Citado na página 15.

CHESTER, D. L. Why two hidden layers are better than one. In: *Proceedings of the international joint conference on neural networks*. [S.l.: s.n.], 1990. v. 1, p. 265–268. Citado na página 27.

CHOI, S.; KIM, T.; YU, W. Performance evaluation of ransac family. *Journal of Computer Vision*, v. 24, n. 3, p. 271–300, 2009. Citado 3 vezes nas páginas 56, 58 e 59.

CHUM, O.; MATAS, J. Matching with prosac-progressive sample consensus. In: IEEE. *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*. [S.l.], 2005. v. 1, p. 220–226. Citado 2 vezes nas páginas 56 e 58.

CIRESAN, D. C. et al. Flexible, high performance convolutional neural networks for image classification. In: *IJCAI Proceedings-International Joint Conference on Artificial Intelligence*. [S.l.: s.n.], 2011. v. 22, n. 1, p. 1237. Citado na página 32.

CYBENKO, G. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, Springer, v. 2, n. 4, p. 303–314, 1989. Citado na página 27.

DOUILLARD, B. et al. On the segmentation of 3d lidar point clouds. In: *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. [S.l.: s.n.], 2011. p. 2798–2805. ISSN 1050-4729. Citado 2 vezes nas páginas 47 e 72.

DUBE, D.; ZELL, A. Real-time plane extraction from depth images with the randomized hough transform. In: IEEE. *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*. [S.l.], 2011. p. 1084–1091. Citado na página 53.

FISCHLER, M. A.; BOLLES, R. C. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, ACM, v. 24, n. 6, p. 381–395, 1981. Citado 4 vezes nas páginas 52, 56, 68 e 94.

FOGEL, D. B. Phenotypes, genotypes, and operators in evolutionary computation. In: IEEE. *Evolutionary Computation, 1995., IEEE International Conference on*. [S.l.], 1995. v. 1, p. 193. Citado na página 59.

FOGEL, L. J.; OWENS, A. J.; WALSH, M. J. *Artificial Intelligence through Simulated Evolution*. New York, USA: John Wiley, 1966. Citado na página 38.

FOIX, S.; ALENYA, G.; TORRAS, C. Lock-in time-of-flight (tof) cameras: a survey. *Sensors Journal, IEEE*, IEEE, v. 11, n. 9, p. 1917–1926, 2011. Citado na página 13.

FUNAHASHI, K.-I. On the approximate realization of continuous mappings by neural networks. *Neural networks*, Elsevier, v. 2, n. 3, p. 183–192, 1989. Citado na página 27.

FUSIELLO, A.; TRUCCO, E.; VERRI, A. A compact algorithm for rectification of stereo pairs. *Machine Vision and Applications*, Springer, v. 12, n. 1, p. 16–22, 2000. Citado na página 43.

GOMPERTS, A.; UKIL, A.; ZURFLUH, F. Implementation of neural network on parameterized fpga. In: *AAAI Spring Symposium: Embedded Reasoning*. [S.l.: s.n.], 2010. Citado na página 32.

GOTARDO, P. et al. Range image segmentation into planar and quadric surfaces using an improved robust estimator and genetic algorithm. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, v. 34, n. 6, p. 2303–2316, 2004. ISSN 1083-4419. Citado 7 vezes nas páginas 52, 53, 54, 55, 57, 59 e 60.

GUO, H.; ZHU, D.; MORDOHAI, P. Correspondence estimation for non-rigid point clouds with automatic part discovery. *The Visual Computer*, Springer Berlin Heidelberg, p. 1–14, 2015. ISSN 0178-2789. Disponível em: <http://dx.doi.org/10.1007/s00371-015-1136-5>. Citado na página 13.

GUPTA, M.; YIN, Q.; NAYAR, S. K. Structured light in sunlight. In: IEEE. *Computer Vision (ICCV), 2013 IEEE International Conference on*. [S.l.], 2013. p. 545–552. Citado na página 13.

GUTMANN, J.-S.; FUKUCHI, M.; FUJITA, M. 3d perception and environment map generation for humanoid robot navigation. *The International Journal of Robotics Research*, SAGE Publications, v. 27, n. 10, p. 1117–1134, 2008. Citado na página 51.

HAMPP, J.; BORMANN, R. Quadtree-based polynomial polygon fitting. In: *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*. [S.l.: s.n.], 2013. p. 4207–4213. ISSN 2153-0858. Citado 3 vezes nas páginas 15, 50 e 72.

HAST, A.; NYSJÖ, J.; MARCHETTI, A. Optimal ransac-towards a repeatable algorithm for finding the optimal set. Václav Skala-Union Agency, 2013. Citado na página 58.

HERNANDEZ, J.; MARCOTEGUI, B. Point cloud segmentation towards urban ground modeling. In: *Urban Remote Sensing Event, 2009 Joint*. [S.l.: s.n.], 2009. p. 1–5. Citado 2 vezes nas páginas 15 e 50.

HIMMELSBACH, M.; HUNDELSHAUSEN, F. V.; WÜNSCHE, H.-J. Fast segmentation of 3d point clouds for ground vehicles. In: IEEE. *Intelligent Vehicles Symposium (IV), 2010 IEEE*. [S.l.], 2010. p. 560–565. Citado na página 15.

HIMMELSBACH, M.; LUETTEL, T.; WUENSCHE, H. Real-time object classification in 3d point clouds using point feature histograms. In: *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*. [S.l.: s.n.], 2009. p. 994–1000. Citado 2 vezes nas páginas 15 e 50.

HIRSCHMÜLLER, H. Accurate and efficient stereo processing by semi-global matching and mutual information. In: IEEE. *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*. [S.l.], 2005. v. 2, p. 807–814. Citado na página 46.

HIRSCHMÜLLER, H. Stereo processing by semiglobal matching and mutual information. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, IEEE, v. 30, n. 2, p. 328–341, 2008. Citado na página 46.

HOLLAND, J. H. *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence.* [S.l.]: U Michigan Press, 1975. Citado na página 36.

HOLZ, D. et al. Real-time plane segmentation using rgb-d cameras. In: *RoboCup 2011: robot soccer world cup XV*. [S.l.]: Springer, 2011. p. 306–317. Citado na página 49.

HORNIK, K.; STINCHCOMBE, M.; WHITE, H. Multilayer feedforward networks are universal approximators. *Neural networks*, Elsevier, v. 2, n. 5, p. 359–366, 1989. Citado na página 27.

HORNUNG, A. et al. Monte carlo localization for humanoid robot navigation in complex indoor environments. *International Journal of Humanoid Robotics*, World Scientific, v. 11, n. 02, p. 1441002, 2014. Citado na página 51.

HUBER, P. J. *Robust statistics*. [S.l.]: Springer, 2011. Citado na página 56.

HULIK, R. et al. Continuous plane detection in point-cloud data based on 3d hough transform. *Journal of Visual Communication and Image Representation*, v. 25, n. 1, p. 86 – 97, 2014. ISSN 1047-3203. Visual Understanding and Applications with RGB-D Cameras. Disponível em: <http://www.sciencedirect.com/science/article/pii/S104732031300062X>. Citado na página 15.

HUMENBERGER, M. et al. A fast stereo matching algorithm suitable for embedded real-time systems. *Computer Vision and Image Understanding*, Elsevier, v. 114, n. 11, p. 1180–1202, 2010. Citado na página 14.

HUSH, D. R.; HORNE, B. G. Progress in supervised neural networks. *Signal Processing Magazine, IEEE*, IEEE, v. 10, n. 1, p. 8–39, 1993. Citado na página 27.

JONG, K. D. *Evolutionary Computation: A Unified Approach*. MIT Press, 2006. (A Bradford book). ISBN 9780262041942. Disponível em: <http://books.google.com.br/books?id=OIRQAAAAMAAJ>. Citado 8 vezes nas páginas 33, 34, 35, 36, 37, 38, 40 e 59.

JORDAN, K.; MORDOHAI, P. A quantitative evaluation of surface normal estimation in point clouds. In: *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*. [S.l.: s.n.], 2014. p. 4220–4226. Citado na página 15.

KECMAN, V. *Learning and Soft Computing: Support Vector Machines, Neural Networks, and Fuzzy Logic Models*. Bradford Book, 2001. (A Bradford book). ISBN 9780262112550. Disponível em: <https://books.google.com.br/books?id=W5SAhUqBVYoC>. Citado 5 vezes nas páginas 20, 22, 24, 26 e 28.

KIDA, Y. et al. Human finding and body property estimation by using floor segmentation and 3d labelling. In: IEEE. *Systems, Man and Cybernetics, 2004 IEEE International Conference on*. [S.l.], 2004. v. 3, p. 2924–2929. Citado na página 51.

KLASING, K.; WOLLHERR, D.; BUSS, M. A clustering method for efficient segmentation of 3d laser data. In: *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*. [S.l.: s.n.], 2008. p. 4043–4048. ISSN 1050-4729. Citado 2 vezes nas páginas 15 e 50.

KONOLIGE, K. et al. Mapping, navigation, and learning for off-road traversal. *Journal of Field Robotics*, Wiley Online Library, v. 26, n. 1, p. 88–113, 2009. Citado 5 vezes nas páginas 15, 94, 95, 100 e 101.

KOZA, J. R. *Genetic programming: on the programming of computers by means of natural selection*. [S.l.]: MIT press, 1992. v. 1. Citado 2 vezes nas páginas 36 e 39.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. Imagenet classification with deep convolutional neural networks. In: *Advances in neural information processing systems*. [S.l.: s.n.], 2012. p. 1097–1105. Citado na página 32.

KUURKOVA, V. Kolmogorov's theorem and multilayer neural networks. *Neural networks*, Elsevier, v. 5, n. 3, p. 501–506, 1992. Citado na página 27.

LALONDE, J.-F. et al. Natural terrain classification using three-dimensional ladar data for ground robot mobility. *Journal of field robotics*, Hoboken, NJ: Wiley, c2006-, v. 23, n. 10, p. 839–862, 2006. Citado na página 15.

LEVINSON, J. et al. Towards fully autonomous driving: Systems and algorithms. In: IEEE. *Intelligent Vehicles Symposium (IV), 2011 IEEE*. [S.l.], 2011. p. 163–168. Citado na página 13.

LIMBERGER, F. A.; OLIVEIRA, M. M. Real-time detection of planar regions in unorganized point clouds. *Pattern Recognition*, v. 48, n. 6, p. 2043 – 2053, 2015. ISSN 0031-3203. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0031320315000072>. Citado 2 vezes nas páginas 15 e 53.

MARCON, G. A.; FERRAO, V. T. *Proposta de Plataforma de Pesquisa em Robótica Móvel Autônoma*. Goiânia, Brasil: Trabalho de Conclusão de Curso, Universidade Federal de Goiás, 2016. Citado 2 vezes nas páginas 17 e 52.

MARCON, G. A. et al. An adaptive algorithm for embedded real-time point cloud ground segmentation. In: *Soft Computing and Pattern Recognition (SoCPaR), 2015 7th International Conference on*. Fukuoka, Japan: Machine Intelligence Research Labs, 2015. p. 76–83. ISBN 978-1-4673-9360-7/15. Citado 3 vezes nas páginas 50, 70 e 72.

MARCON, G. A. et al. Performance analysis for a novel adaptive algorithm for real-time point cloud ground segmentation. *The International Journal of Hybrid Intelligent Systems (IJHIS)*, IOS Press, v. 12:3, 2016. ISSN 1448-5869. Citado 4 vezes nas páginas 17, 50, 51 e 93.

MARK, W. V. D.; GAVRILA, D. M. Real-time dense stereo for intelligent vehicles. *Intelligent Transportation Systems, IEEE Transactions on*, IEEE, v. 7, n. 1, p. 38–50, 2006. Citado na página 14.

MASUDA, H. et al. Reconstruction of polygonal faces from large-scale point-clouds of engineering plants. *Computer-Aided Design and Applications*, v. 12, n. 5, p. 555–563, 2015. Disponível em: <http://dx.doi.org/10.1080/16864360.2015.1014733>. Citado na página 13.

MATAS, J.; CHUM, O. Randomized ransac with sequential probability ratio test. In: IEEE. *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*. [S.l.], 2005. v. 2, p. 1727–1732. Citado na página 58.

MERRY, B.; MARAIS, P.; GAIN, J. Compression of dense and regular point clouds. In: WILEY ONLINE LIBRARY. *Computer Graphics Forum*. [S.l.], 2006. v. 25, n. 4, p. 709–716. Citado na página 14.

MOOSMANN, F.; PINK, O.; STILLER, C. Segmentation of 3d lidar data in non-flat urban environments using a local convexity criterion. In: *Intelligent Vehicles Symposium, 2009 IEEE*. [S.l.: s.n.], 2009. p. 215–220. ISSN 1931-0587. Citado 2 vezes nas páginas 15 e 50.

MUNOZ, D.; VANDAPEL, N.; HEBERT, M. Onboard contextual classification of 3-d point clouds with learned high-order markov random fields. In: *Robotics and Automation, 2009. ICRA '09. IEEE International Conference on*. [S.l.: s.n.], 2009. p. 2009–2016. ISSN 1050-4729. Citado na página 82.

MUNRO, P.; GERDELAN, A. P. Stereo vision computer depth perception. *Country United States City University Park Country Code US Post code*, Citeseer, v. 16802, 2009. Citado na página 42.

MURRAY, R. M.; SASTRY, S. S.; ZEXIANG, L. *A Mathematical Introduction to Robotic Manipulation*. 1st. ed. Boca Raton, FL, USA: CRC Press, Inc., 1994. ISBN 0849379814. Citado na página 77.

OCHMANN, S. et al. Automatic reconstruction of parametric building models from indoor point clouds. *Computers and Graphics*, v. 54, p. 94 – 103, 2016. ISSN 0097-8493. Special Issue on CAD/Graphics 2015. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0097849315001119>. Citado na página 13.

OMONDI, A. R.; RAJAPAKSE, J. C. *FPGA implementations of neural networks*. [S.l.]: Springer, 2006. v. 365. Citado 2 vezes nas páginas 31 e 32.

ONIGA, F.; NEDEVSCHI, S. Processing dense stereo data using elevation maps: Road surface, traffic isle, and obstacle detection. *Vehicular Technology, IEEE Transactions on*, IEEE, v. 59, n. 3, p. 1172–1182, 2010. Citado na página 50.

PARK, J.; CHOI, S.; YU, W. Quadtree sampling-based superpixels for 3d range data. In: *Robotics and Automation (ICRA), 2014 IEEE International Conference on*. [S.l.: s.n.], 2014. p. 5495–5501. Citado 3 vezes nas páginas 15, 50 e 72.

PASSINO, K. M. *Biomimicry for optimization, control, and automation*. [S.l.]: Springer Science & Business Media, 2005. Citado 2 vezes nas páginas 19 e 34.

RODEHORST, V.; HELLWICH, O. Genetic algorithm sample consensus (gasac)-a parallel strategy for robust parameter estimation. In: IEEE. *Computer Vision and Pattern Recognition Workshop, 2006. CVPRW'06. Conference on*. [S.l.], 2006. p. 103–103. Citado na página 59.

ROSENBLATT, F. Principles of neurodynamics. Spartan Book, 1962. Citado na página 22.

ROUSSEEUW, P. J. Least median of squares regression. *Journal of the American statistical association*, Taylor & Francis, v. 79, n. 388, p. 871–880, 1984. Citado 2 vezes nas páginas 56 e 57.

RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. *Learning internal representations by error propagation*. [S.l.], 1985. Citado na página 23.

RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. Learning representations by back-propagating errors. *Cognitive modeling*, v. 5, 1988. Citado na página 24.

RUSU, R. B.; COUSINS, S. 3D is here: Point Cloud Library (PCL). In: *IEEE International Conference on Robotics and Automation (ICRA)*. Shanghai, China: [s.n.], 2011. Citado 2 vezes nas páginas 68 e 71.

RUSU, R. B. et al. Model-based and learned semantic object labeling in 3d point cloud maps of kitchen environments. In: IEEE. *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*. [S.l.], 2009. p. 3601–3608. Citado na página 51.

SABOV, A.; KRÜGER, J. Segmentation of 3d points from range camera data using scanlines. In: IEEE. *Systems, Signals and Image Processing, 2008. IWSSIP 2008. 15th International Conference on*. [S.l.], 2008. p. 429–432. Citado na página 53.

SCHNABEL, R.; WAHL, R.; KLEIN, R. Efficient ransac for point-cloud shape detection. *Computer Graphics Forum*, Blackwell Publishing Ltd, v. 26, n. 2, p. 214–226, 2007. ISSN 1467-8659. Disponível em: <http://dx.doi.org/10.1111/j.1467-8659.2007.01016.x>. Citado 10 vezes nas páginas 14, 15, 49, 52, 53, 55, 56, 59, 72 e 94.

SCHOENBERG, J.; NATHAN, A.; CAMPBELL, M. Segmentation of dense range information in complex urban scenes. In: *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*. [S.l.: s.n.], 2010. p. 2033–2038. ISSN 2153-0858. Citado 2 vezes nas páginas 15 e 50.

SCHWEFEL, H. P. Evolutionsstrategie und numerische optimierung. *Dr.-Ing. Diss. Technical University of Berlin, Department of Process Engineering, Berlin*, 1975. Citado na página 37.

SPEARS, W. M. et al. An overview of evolutionary computation. In: SPRINGER. *Machine Learning: ECML-93*. [S.l.], 1993. p. 442–459. Citado na página 59.

STEELE, G. L. *Common LISP: the language*. [S.l.]: Digital press, 1990. Citado na página 38.

STROM, J.; RICHARDSON, A.; OLSON, E. Graph-based segmentation for colored 3d laser point clouds. In: *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*. [S.l.: s.n.], 2010. p. 2131–2136. ISSN 2153-0858. Citado 2 vezes nas páginas 15 e 50.

SUGANTHAN, S.; COLEMAN, S.; SCOTNEY, B. Single-step planar surface extraction from range images. In: CITESEER. *Proc. 4th IEEE International Symposium on 3D Data Processing, Visualization and Transmission*. [S.l.], 2008. p. 363–372. Citado na página 53.

TODA, Y.; KUBOTA, N. Behavior analysis of evolution strategy sample consensus. In: *Mecatronics (MECATRONICS), 2014 10th France-Japan/ 8th Europe-Asia Congress on*. [S.l.: s.n.], 2014. p. 250–255. Citado 4 vezes nas páginas 52, 53, 59 e 60.

TORR, P. H.; ZISSERMAN, A. Mlesac: A new robust estimator with application to estimating image geometry. *Computer Vision and Image Understanding*, Elsevier, v. 78, n. 1, p. 138–156, 2000. Citado 7 vezes nas páginas 54, 56, 57, 58, 67, 69 e 95.

TREVOR, A. J. et al. Planar surface slam with 3d and 2d sensors. In: IEEE. *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. [S.l.], 2012. p. 3041–3048. Citado na página 51.

TUKEY, J. W. A survey of sampling from contaminated distributions. *Contributions to probability and statistics*, v. 2, p. 448–485, 1960. Citado 2 vezes nas páginas 56 e 57.

WAHL, R.; GUTHE, M.; KLEIN, R. Identifying planes in point-clouds for efficient hybrid rendering. In: *The 13th Pacific Conference on Computer Graphics and Applications*. [S.l.: s.n.], 2005. Citado 5 vezes nas páginas 14, 52, 53, 55 e 59.

WEBER, T.; HäNSCH, R.; HELLWICH, O. Automatic registration of unordered point clouds acquired by kinect sensors using an overlap heuristic. *{ISPRS} Journal of Photogrammetry and Remote Sensing*, v. 102, p. 96 – 109, 2015. ISSN 0924-2716. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0924271614002895>. Citado na página 13.

WIDROW, B.; HOFF, M. E. et al. Adaptive switching circuits. Defense Technical Information Center, 1960. Citado na página 20.

XIAO, J. et al. Planar segment based three-dimensional point cloud registration in outdoor environments. *Journal of Field Robotics*, Wiley Online Library, v. 30, n. 4, p. 552–582, 2013. Citado 3 vezes nas páginas 15, 51 e 53.

YANG, M. Y.; FÖRSTNER, W. Plane detection in point cloud data. In: *Proceedings of the 2nd int conf on machine control guidance, Bonn*. [S.l.: s.n.], 2010. v. 1, p. 95–104. Citado 5 vezes nas páginas 52, 53, 55, 59 e 94.

ZHANG, C.; WANG, L.; YANG, R. Semantic segmentation of urban scenes using dense depth maps. In: *Computer Vision–ECCV 2010*. [S.l.]: Springer, 2010. p. 708–721. Citado na página 49.

ZHANG, J.; SINGH, S. Visual-lidar odometry and mapping: low-drift, robust, and fast. In: *IEEE International Conference on Robotics and Automation, ICRA 2015, Seattle, WA, USA, 26-30 May, 2015*. [s.n.], 2015. p. 2174–2181. Disponível em: <http://dx.doi.org/10.1109/ICRA.2015.7139486>. Citado na página 13.

ZHANG, Z. Determining the epipolar geometry and its uncertainty: A review. *International journal of computer vision*, Springer, v. 27, n. 2, p. 161–195, 1998. Citado na página 56.

ZHU, J.; SUTTON, P. Fpga implementations of neural networks–a survey of a decade of progress. In: *Field Programmable Logic and Application*. [S.l.]: Springer, 2003. p. 1062–1066. Citado na página 31.