

INSTITUTO FEDERAL DO ESPÍRITO SANTO
BACHARELADO EM SISTEMAS DE INFORMAÇÃO

LEONARDO CAMILO RIBEIRO

**DETECÇÃO E RASTREAMENTO DE UM ALVO NO
CAMPO VISUAL DE UM ROBÔ AUTÔNOMO USANDO
UM SENSOR RGB-D**

SERRA
2017

LEONARDO CAMILO RIBEIRO

**DETECÇÃO E RASTREAMENTO DE UM ALVO NO
CAMPO VISUAL DE UM ROBÔ AUTÔNOMO USANDO
UM SENSOR RGB-D**

Trabalho de Conclusão de Curso apresentado
à Coordenadoria do Curso de Bacharelado
em Sistemas de Informação do Instituto
Federal do Espírito Santo, como requisito
parcial para obtenção do título de Bacharel
em Sistemas de Informação.

Orientador:
Prof. Me. Eduardo Max A. Amaral

SERRA
2017

P149e Leonardo Camilo Ribeiro

DETECÇÃO E RASTREAMENTO DE UM ALVO NO CAMPO VISUAL DE UM ROBÔ AUTÔNOMO USANDO UM SENSOR RGB-D/ Leonardo Camilo Ribeiro. – Serra, 2017-

79 p. : il. (algumas color.) ; 30 cm.

Orientador: Prof. Me. Eduardo Max A. Amaral

Monografia (Graduação) – Instituto Federal do Espírito Santo ,
Coordenadoria de Informática, Curso Bacharelado em Sistemas de Informação, 2017.

1. Visão Computacional. 2. Kinect. 3. Nuvem de Pontos. I. Amaral, Eduardo. II. Instituto Federal do Espírito Santo. III. Título.

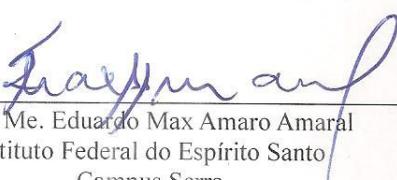
LEONARDO CAMILO RIBEIRO

**DETECÇÃO E RASTREAMENTO DE UM ALVO NO CAMPO VISUAL DE UM ROBÔ
AUTÔNOMO USANDO UM SENSOR RGB-D**

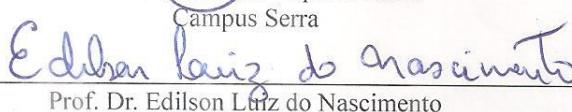
Trabalho de Conclusão de Curso apresentado
como parte das atividades para obtenção do
título de Bacharel em Sistemas de Informação,
do curso de Bacharelado em Sistemas de
Informação do Instituto Federal do Espírito
Santo.

Aprovado em 13 de julho de 2017.

COMISSÃO EXAMINADORA

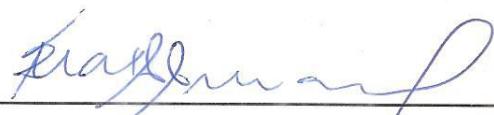

Prof. Me. Eduardo Max Amaro Amaral
Instituto Federal do Espírito Santo
Campus Serra


Prof. Dr. Maxwell Eduardo Monteiro
Instituto Federal do Espírito Santo
Campus Serra


Prof. Dr. Edilson Luiz do Nascimento
Instituto Federal do Espírito Santo
Campus Serra

Aceite do Professor Orientador de TCC Bacharelado em Sistemas de Informação

Eu, **Eduardo Max Amaro Amaral**, professor orientador do aluno **LEONARDO CAMILO RIBEIRO**, do curso Superior de Bacharelado em Sistemas de Informação, declaro que o Trabalho de Conclusão de Curso do aluno supracitado, intitulado "**DETECÇÃO E RASTREAMENTO DE UM ALVO NO CAMPO VISUAL DE UM ROBÔ AUTÔNOMO USANDO UM SENSOR RGB-D**" cumpre as exigências e modificações solicitadas pela banca examinadora na ata de defesa.



Eduardo Max Amaro Amaral

Aos meus pais

Agradecimentos

Agradeço a Deus por ter me guiado até aqui, a minha Família que sempre me ajudou, a meus Professores que me proporcionaram a oportunidade e o conhecimento para chegar até aqui, e a meus amigos que sempre me apoiaram nos momentos difíceis.

Não andem ansiosos por coisa alguma, mas em tudo, pela oração e súplicas, e com ação de graças, apresentem seus pedidos a Deus. E a paz de Deus, que excede todo o entendimento, guardará os seus corações e as suas mentes em Cristo Jesus.

Filipenses 4:6,7

Resumo

Com a ampla divulgação da tecnologia em nosso cotidiano, estudos na área da robótica se tornaram comuns no meio científico, de modo que diversas inovações vem sendo feitas nessa área. Devido a tais inovações, inúmeras atividades na sociedade, antes exercidas por pessoas, hoje são realizadas por robôs autônomos. Uma das áreas que cada vez mais surgem novidades tecnológicas é a de segurança e vigilância devido a alta insalubridade e propensão à falha humana. Neste trabalho, foi desenvolvido um sistema de detecção e rastreamento de um objeto alvo no campo visual de um robô autônomo usando um sensor RGB-D (Kinect) para aplicações em sistemas de patrulhamento. Para a realização da tarefa, o sistema proposto opera em quatro etapas. A cada leitura do sensor é realizado o pré-processamento, a segmentação, a associação e a classificação da nuvem de pontos 3D. Na etapa de pré-processamento ocorre a remoção dos pontos 3D do chão e a redução do conjunto de amostragem. Na etapa de segmentação, a nuvem de pontos 3D é segmentada em agrupamentos de pontos usando a distância Euclidiana, sendo que cada agrupamento representa um objeto no ambiente. Na etapa de associação, os objetos observados na varredura atual do sensor são associados aos mesmos objetos observados em varreduras anteriores usando o algoritmo do vizinho mais próximo (*nearest neighbor*). Em seguida, os objetos são classificados por meio do método de descrição *VFH* e um classificador *KNN* através da distância entre os histogramas, com base em um conjunto de treino, indicando se pertencem ou não ao conjunto de modelos do objeto alvo. Por fim, uma caixa delimitadora é inserida sobre a nuvem de pontos 3D usando como referência o centro de massa do objeto classificado como objeto alvo. O desempenho do sistema proposto foi avaliado usando dados de um sensor RGB-D (Kinect) gerados por uma Plataforma Robótica experimental (PR) em um ambiente *indoor*. Os resultados experimentais mostraram que o sistema proposto foi capaz de detectar, classificar e rastrear com bom desempenho um objeto alvo no ambiente à frente da plataforma robótica.

Palavras chaves: Visão Computacional, Kinect, Nuvem de Pontos 3D.

Abstract

With the widespread dissemination of technology in our daily lives, studies in the field of robotics have become extremely common in the scientific environment, so that several innovations have been made in this area. Due to such innovations, several activities in society formerly carried out by people are now carried out by autonomous robots. One of the areas that is increasingly implementing studies in the area of robotics is that of security and surveillance due to the high salubrity and propensity to human failure. In this work, was developed a system to detect and track objects in the visual field of a Kinect sensor for security applications in a surveillance robot. For the accomplishment of the task, the system proposed operates in four stages, each sensor's reading is carried out a preprocessing, segmentation, association, and 3D point cloud classification of the cloud. In the pre-processing step is realized removing the floor data through the RANSAC method and the sampling space is reduced through the VoxelGrid method, in segmentation the 3D point cloud is segmented in clusters of points using the Euclidean distance approach, each cluster represents an object in the environment. In the association step, the objects observed in the current reading of the sensor are associated with the same objects observed in past scans, using the nearest neighbor algorithm. Then the objects are classified by the VFH descriptor and KNN classifier, through the distance between the histograms, based on a training set, indicating if they belong to the model set of the target object. Finally, a bounding box is added over the cloud of 3D points using as reference the center of mass of the classified object. The performance of the proposed system was evaluated using data from an RGB-D sensor (Kinect) generated by an experimental Robotic Platform in an indoor environment. The experimental results showed that the system was able to detect, classify and track a target object with good performance in the environment at the front of the robotic platform.

Key words: Computer Vision, Kinect, Point Cloud.

Listas de ilustrações

Figura 1.1 – Exemplo de robôs de patrulha.	15
Figura 2.1 – Fluxo de processamento.	19
Figura 2.2 – Exemplo de Segmentação de uma nuvem de pontos.	20
Figura 2.3 – Exemplo de Agrupamento feito pelo DBSCAN.	21
Figura 2.4 – Exemplo prático de funcionamento do DBSCAN.	22
Figura 2.5 – Exemplo de aplicação do JDC	23
Figura 2.6 – Fluxo de funcionamento da classificação em nuvem de pontos	24
Figura 2.7 – Modelo de execução do PFH	25
Figura 2.8 – Visualização dos componentes $\langle\alpha, \phi, \theta, d\rangle$	26
Figura 2.9 – Raio de influência de um ponto no FPFH	27
Figura 2.10 – Visualização da classificação SVM	28
Figura 2.11 – Exemplo de classificação KNN com $k = 3$	29
Figura 2.12 – Formação da árvore hipóteses	31
Figura 2.13 – Aplicação do MHT em detecção de pedestres	32
Figura 2.14 – Distribuição de partículas	32
Figura 2.15 – Convergência do filtro de partículas no rastreamento	33
Figura 3.1 – Fluxo de funcionamento do sistema proposto.	35
Figura 3.2 – Sensor Kinect da empresa Microsoft.	35
Figura 3.3 – Processo de extração de dados do Kinect	36
Figura 3.4 – Funcionamento da disparidade da câmera.	37
Figura 3.5 – Exemplo de aplicação do RANSAC.	38
Figura 3.6 – Aplicação do RANSAC para remoção do chão.	39
Figura 3.7 – Resolução do problema de falsos positivos na aplicação do RANSAC.	40
Figura 3.8 – Representação de uma grade de voxel	41
Figura 3.9 – Aplicação do VoxelGrid na nuvem de pontos	42
Figura 3.10 – Aplicação da segmentação na nuvem de pontos	45
Figura 3.11 – Aplicação da segmentação com limitador na nuvem de pontos	46
Figura 3.12 – Representação temporal de um ponto	47
Figura 3.13 – Exemplo de associação no tempo de uma sequencia de nuvens	48
Figura 3.14 – Estimação de normal por vizinhos próximos.	50
Figura 3.15 – Vetor normal estimado para cada ponto da nuvem	51
Figura 3.16 – Extração de características entre pontos.	51
Figura 3.17 – Histograma VFH (<i>Viewpoint Feature Histogram</i>).	52
Figura 3.18 – Exemplos de histogramas VFH de modelos aleatórios	53
Figura 4.1 – Infraestrutura utilizada no sistema.	55
Figura 4.2 – Fluxo de funcionamento do ROS.	56

Figura 4.3 – Fluxo de funcionamento dos nós do ROS implementados neste trabalho.	56
Figura 4.4 – Nuvem após RANSAC com distância limiar de $2cm$	58
Figura 4.5 – Nuvem após RANSAC com distância limiar de $4cm$	58
Figura 4.6 – Nuvem após RANSAC com distância limiar de $6cm$	59
Figura 4.7 – Diferença entre nuvens com tamanhos de <i>voxel</i> diferentes.	61
Figura 4.8 – Captura de índices de semelhança de um objeto	63
Figura 4.9 – Captura de índices de semelhança de um objeto	64
Figura 4.10–Instantes de tempo da nuvem no período de teste em diferentes ângulos.	65
Figura 4.11–Diferentes instantes de tempo da nuvem no período de teste	67
Figura 4.12–Diferentes instantes de tempo da nuvem no período de teste	68
Figura 4.13–Variação do deslocamento de objetos classificados.	69

Lista de tabelas

Tabela 4.1 – Desempenho de cada par de parâmetros do RANSAC	57
Tabela 4.2 – Desempenho de redução do <i>VoxelGrid</i>	59
Tabela 4.3 – Impacto do VoxelGrid na segmentação.	59
Tabela 4.4 – Dados obtidos dos testes.	70
Tabela 4.5 – Índices avaliativos da classificação.	70

Sumário

1	Introdução	14
1.1	Objetivos	17
1.1.1	Objetivo Geral	17
1.1.2	Objetivos Específicos	17
1.2	Organização do trabalho	17
2	Referencial Teórico	19
2.1	Segmentação	20
2.1.1	DBSCAN	20
2.1.2	JDC	22
2.2	Descrição e Classificação	23
2.2.1	PFH	24
2.2.2	FPFH	26
2.2.3	SVM	27
2.2.4	KNN	28
2.3	Associação e Rastreamento	29
2.3.1	JPDA	30
2.3.2	MHT	30
2.3.3	Filtro de Partículas	31
3	Desenvolvimento	34
3.1	Visão Geral	34
3.2	Pré-processamento	37
3.2.1	Remoção do Plano do chão	37
3.2.2	VoxelGrid	40
3.3	Segmentação	42
3.3.1	Segmentação de objetos	43
3.4	Associação	46
3.4.1	Associação por Vizinhos Próximos	47
3.5	Classificação	48
3.5.1	VFH	49
3.5.2	Aplicação do VFH	52
4	Experimentos, Resultados e Discussão	54
4.1	Materiais e Métodos	54
4.1.1	Infraestrutura	54
4.1.2	ROS e PCL	54

4.2	Experimentos Associados ao Pré-Processamento	56
4.2.1	Remoção do Plano	56
4.2.2	Redução de Amostragem	57
4.3	Experimentos Associados à Classificação	62
4.3.1	Escolha do Limiar	62
4.3.2	Avaliação Qualitativa	64
4.3.3	Avaliação Quantitativa	66
4.4	Considerações	70
5	Considerações Finais	72
5.1	Trabalhos Futuros	72
	Referências	74
	Apêndice A – Código Fonte	79

1 Introdução

Com a ampla divulgação da tecnologia no cotidiano, sensores e peças eletromecânicas vem se tornando algo cada vez mais de baixo custo e de fácil acesso. Por conta disso, houve uma crescente dedicação à pesquisa de soluções tecnológicas para melhorar a qualidade de vida das pessoas. A tecnologia pode realmente fornecer uma ampla gama de soluções, em diferentes níveis de complexidade e custo. O recente progresso da pesquisa em robótica avançada permite a aplicação de soluções robóticas para ajudar as pessoas, principalmente no que se refere a área de robótica autônoma.

Robótica autônoma se refere a junção de dois conceitos, robótica e automação. Segundo ([THRUN WOLFRAM BURGARD, 2005](#)), robótica é a ciência de perceber e manipular o mundo físico através de dispositivos controlados por computadores, e tais dispositivos, os quais são o principal objeto de estudo da robótica são denominados robôs, que por sua vez podem ser definidos como, um ou mais dispositivos, agrupados, eletricamente ou bio mecanicamente, capazes de realizar trabalhos de maneira autônoma ou pré-programada. Por outro lado, automação, *Automatus* em latim, significa mover-se por si. Segundo ([PRIBERAM, 2016](#)), automação pode ser definida como a execução automática de tarefas industriais ou científicas sem intervenção humana intermediária. Ou seja, trata-se da realização de tarefas de maneira autônoma, sem a assistência ou supervisão de alguém. Portanto, robótica autônoma pode ser definida como a ciência de perceber e manipular o mundo físico sem a intervenção humana intermediária.

O uso de robôs autônomos, sejam eles por software, hardware, ou ambos, se faz cada vez mais necessário em áreas críticas da sociedade, visto que mesmo após anos de evolução, ainda existem tarefas extremamente insalubres, e uma dessas áreas é a da segurança. Há um dito popular que diz, “*o seu direito acaba onde começa o dos outros*”, que diz respeito aos limites em que cada um está sujeito debaixo da lei. Porém, é de conhecimento geral que existem pessoas que não respeitam tais limites, infringindo tanto a lei como o direito alheio, e por conta disso que o ramo da segurança se faz tão necessário em nossa sociedade. Para se ter ideia da importância da mesma, segundo ([MOSANYA, 2017](#)), no dia 04 de fevereiro de 2017 a polícia militar do estado do Espírito Santo deu início a uma greve geral, que teve como consequência uma crise de segurança pública, onde após uma semana de greve foram registradas mais de 120 mortes.

Porém, a área de segurança é demasiadamente insalubre, e, por conta disso, se faz cada vez mais necessário a utilização de robôs e sistemas automatizados nessa área, salvando assim a integridade de quem está disposto a defender a segurança dos outros.

Diversas atividades já são exercidas por robôs nessa área, como por exemplo

desarmadores de bombas, *drones* (Veículo aéreo não tripulado) de reconhecimento, etc. e uma possibilidade que se torna cada vez mais relevante é a de perseguição autônoma, a qual robôs automatizados possam realizar perseguições de alvos sem que seja necessário alguém estar em risco para isso.

Figura 1.1 – Exemplo de robôs de patrulha.



a) robô de patrulha na china



b) robô de patrulha da NNSA



c) robô de patrulha k5



d) Carro de patrulha na faixa de Gaza

Fonte: Imagens retiradas da internet.

Alguns exemplos dessa aplicação são robôs de ronda, como demonstrado na Figura 1.1. Em ([ROGERS, 2016](#)) é mostrado um carro semi-autônomo de patrulha da faixa de Gaza, área considerada extremamente perigosa. Outro exemplo é mostrado em ([SCHILLER, 2015](#)) e ([LI, 2016](#)), onde robôs do modelo *K5*, da empresa *knightscope*, são utilizados para realizar a ronda de inúmeros locais. O robô utiliza inúmeros sensores como lasers, sensores visuais e de áudio, para detecção de sons como buzina de carro, quebra de vidro, pessoas gritando, etc. Além disso, é capaz de executar a construção de mapas tridimensionais do local para criação de rotas de patrulha.

No campo da robótica autônoma, sensores visuais são um dos mais utilizados, pelo fato de serem capazes de entregar uma enorme gama de informações sobre o universo observável em que atuam os robôs. Neste contexto, as câmeras de profundidade estão alterando a percepção do robô e a visão de máquina em todo o mundo, substituindo câmeras mono-visão, estéreo e outros tipos de buscadores de alcance, como sensores laser e ultra-sônicos.

O sensor Microsoft Kinect é uma revolucionária câmera de profundidade que é usada na indústria de jogos para capturar movimentos de pessoas e jogadores de forma eficiente,

usando a tecnologia de uma câmera RGB e um sensor infravermelho para diferenciar a profundidade. No Microsoft X-Box, o Kinect é usado para gerar uma percepção 3D dos movimentos humanos. Na robótica, usando o Kinect, um robô pode ter uma melhor compreensão dos objetos à sua frente através dessa percepção 3D. Com a capacidade do Kinect de produzir imagens de qualidade e informações detalhadas sobre o ambiente e usando certos algoritmos disponíveis na visão computacional, é possível programar algoritmos de navegação e detecção de objetos para robôs autônomos.

Além disso, o uso do sensor Kinect pode representar uma redução significativa dos custos de um robô, já que este sensor pode recriar um ambiente 3D com custo bem inferior ao de um sensor laser. Por outro lado, o uso do Kinect se restringe ao ambiente *indoor* ou ao uso noturno, já que esse sensor sofre interferência da radiação solar, devido ao uso do infravermelho como sensor de profundidade.

Com o Kinect é possível gerar uma nuvem de pontos 3D. Tanto a câmera RGB quanto o sensor de profundidade possuem resolução 640x480, e a cada *frame* é gerada uma nuvem de pontos que possui 307200 pontos. Para cada ponto mapeado, são fornecidas informações de profundidade e de cor (RGB).

Existem diversos trabalhos que utilizam o Kinect como sensor, para gerar uma percepção 3D do ambiente (nuvem de pontos 3D) na detecção e no reconhecimento de objetos, como por exemplo ([PANIAGUA, 2011](#)) que utiliza o *Normal Aligned Radial Feature (NARF)*. O descriptor NARF, proposto por ([STEDER et al., 2011](#)), é um classificador onde os pontos-chave são calculados diretamente a partir da nuvem de pontos 3D. O mesmo pode ser encontrado na *Point Cloud Library (PCL)* ([RUSU; COUSINS, 2011](#)), além de inúmeros outros classificadores como ele, como por exemplo o *Scale Invariant Feature Transform (SIFT)* ([LOWE, 1999](#)), o *Speeded Up Robust Features (SURF)* ([BAY et al., 2008](#)), o *Point Feature Histogram (PFH)* ([RUSU et al., 2008](#)), o *Fast Point Feature Histogram (FPFH)* ([RUSU; BLODOW; BEETZ, 2009](#)), e o *Viewpoint Feature Histogram (VFH)* ([RUSU et al., 2010](#)). Além desses classificadores, existem outras abordagens diferentes, específicas para execução de tarefas pré-definidas, como é demonstrado em ([GRITTI et al., 2014](#)), no qual é utilizada uma abordagem completamente diferente pra identificação de pessoas baseadas em uma visualização próximo ao chão.

Além de reconhecer objetos no campo visual do sensor, também é possível rastreá-los conforme o passar do tempo, de modo que o observador saiba que o objeto que foi visto no tempo anterior é o mesmo que é visto no momento atual. Assim como reconhecimento de objetos, inúmeros trabalhos também realizam a funcionalidade de rastreamento como ([AMARAL et al., 2015](#)), ([SCHULZ et al., 2001](#)), ([ALMEIDA; ALMEIDA; Rui Araujo, 2005](#)), ([Chieh-Chih Wang; THORPE; THRUN, 2003](#)).

O problema de detecção e rastreamento de objetos na robótica autônoma é conhecido como DATMO (*detection and tracking of moving objects*). DATMO envolve a detecção de

objeto em movimento no ambiente à frente do robô autônomo e o seu rastreamento, i.e., a estimativa do seu estado (e.g., posição, orientação e velocidade) ao longo do tempo.

Neste contexto, talvez seja possível a criação de mecanismos automatizados de perseguição de alvo a serem implementados em robôs de patrulha, de modo que não seja necessária a interferência humana. Desta forma, o objetivo desse trabalho é a implementação de um sistema de reconhecimento e perseguição de um alvo em um sistema robótico autônomo a ser aplicado em sistemas de patrulhamento.

1.1 Objetivos

1.1.1 Objetivo Geral

1. O objetivo deste trabalho é desenvolver um sistema, baseado nos conceitos de DATMO, capaz de reconhecer e perseguir um alvo em um sistema robótico autônomo a ser aplicado em sistemas de patrulhamento. Para isso, o sistema irá utilizar o sensoreamento visual de profundidade do Kinect e conceitos de visão computacional para execução da tarefa.

1.1.2 Objetivos Específicos

1. Criar um ambiente de comunicação e troca de dados para desenvolvimento do projeto com o ROS (*Robot Operational System*)
2. Desenvolver um modelo de pré-processamento de dados para remoção de ruído dos dados
3. Estudar e implementar um modelo de segmentação de nuvem de pontos
4. Estudar e implementar um modelo de associação temporal de conjuntos de nuvens de pontos
5. Estudar e implementar um modelo de classificação de uma nuvem de pontos
6. Avaliar o desempenho do modelo desenvolvido em um ambiente real

1.2 Organização do trabalho

O trabalho está dividido da seguinte maneira: No Capítulo 2 é realizada uma revisão dos conceitos e trabalhos correlatos importantes para o entendimento do sistema. No Capítulo 3 é apresentado o sistema proposto neste trabalho para detecção e rastreamento de um alvo no campo visual de um robô autônomo. No capítulo 4 são apresentados a

metodologia experimental, os materiais usados e os resultados dos testes e avaliações. Por fim, no Capítulo 5 são apresentadas as conclusões e possíveis direções para trabalhos futuros.

2 Referencial Teórico

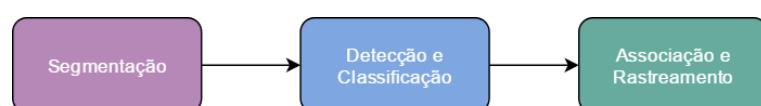
Para realizar a tarefa de perseguição nesse trabalho, utilizamos o conceito de DATMO (*Detection And Tracking of Moving Objects*), que por sua vez se trata de métodos de detecção e rastreamento de objetos móveis utilizando dados de sensores. A necessidade de se automatizar tarefas deste tipo não é uma ideia nova. Segundo (PETROVSKAYA et al., 2012), a necessidade de mecanismos de DATMO nasceu por volta de 1980 juntamente com o interesse em veículos inteligentes e autônomos. Um grande número de projetos exploratórios foram desenvolvidos na Europa em 1986 sobre o nome de PROMETHEUS (*PROgramme for a European Traffic of Highest Efficiency and Unprecedented Safety*), seguido por iniciativas no Japão e nos Estados Unidos.

Acarretados por inovações e melhorias na tecnologia empregada em sensores, pesquisas recentes na área de DATMO tem focado cada vez mais em melhorias de detecção utilizando técnicas de reconhecimento de padrões, aumentando a robustez dos métodos com a utilização de inúmeros sensores diferentes, e melhorando cada vez mais a acurácia dos processos.

Existem inúmeras aplicações diferentes para o problema de DATMO, desde a automatização de um robô de chão para ambientes internos e movimentados (GRITTI et al., 2014), como a automatização de um carro (THRUN et al., 2006), onde uma equipe da universidade de Stanford desenvolveu um carro autônomo. O veículo é denominado Stanley, e venceu uma competição promovida pela *Defense Advanced Research Projects Agency* (DARPA), a competição era denominada *DARPA Grand Challenge*, que se trata de uma corrida para carros autônomos que visa o desenvolvimento da tecnologia.

Porém, independente do contexto em que é aplicado, no que se refere a trabalhos da área de DATMO, grande parte da literatura (AMARAL et al., 2015) utiliza o mesmo fluxo de processamento para realização da tarefa, visto na Figura 2.1, métodos de segmentação, detecção ou classificação, e por fim, associação e rastreamento. Neste trabalho iremos utilizar alguns desse métodos para realizar a perseguição de um alvo, no alcance visual do Kinect, por um robô autônomo.

Figura 2.1 – Fluxo de processamento.



Fonte: Imagem de autoria do autor.

Existem inúmeros tipos de sensores utilizados em trabalhos da área, mas os mais

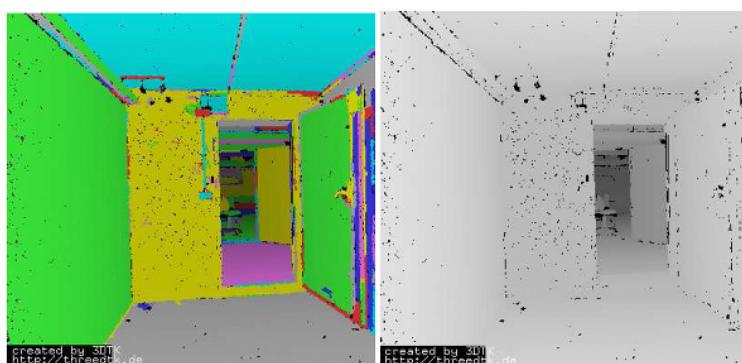
comuns e com maior diversidade de informação são os sensores visuais, como por exemplo sensores LIDAR (*Light Detection And Ranging*) bidimensionais, tridimensionais, câmeras estéreo, etc. Neste trabalho foi utilizado o Kinect, da Microsoft, como sensor da aplicação, devido a seu baixo custo e sua qualidade de sensoreamento.

As próximas sessões abordarão alguns conceitos e métodos utilizados em outros trabalhos para realização de cada uma das etapas de DATMO.

2.1 Segmentação

A etapa de segmentação visa agrupar sub-conjuntos de dados pertencentes a um conjunto maior em agrupamentos distintos, de modo que todos os pontos referentes ao um mesmo objeto sejam segmentados juntos. É comumente utilizado o método de distância limiar para realizar a segmentação em um conjunto de dados de pontos tridimensionais.

Figura 2.2 – Exemplo de Segmentação de uma nuvem de pontos.



Fonte: <http://kos.informatik.uni-osnabrueck.de/icar2013/segmentation.png>

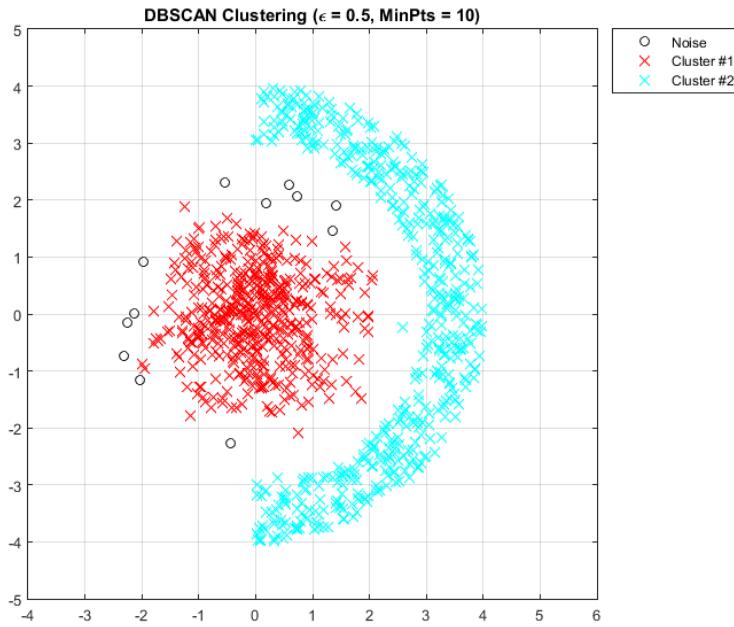
Em uma aplicação onde o conjunto a ser segmentado é uma nuvem de pontos (Figura 2.2), a etapa de segmentação é responsável por interpretar a imagem, de modo a retirar informação útil de um conjunto de pontos no espaço, como por exemplo, quais pontos da nuvem são referentes ao chão, parede, etc., quais pontos compõe cada objeto, quais são considerados ruído e podem ser descartados, entre outras coisas. É uma parte essencial para o processo, possibilitando assim, que seja possível realizar as etapas seguintes, e neste capítulo iremos abordar alguns métodos utilizados para realização de tal tarefa.

2.1.1 DBSCAN

O DBSCAN (*Density-Based Spatial Clustering of Applications with Noise*) definido em (ESTER et al., 1996) se baseia em um método de detecção de agrupamentos por densidade. Ou seja, um conjunto de pontos é denominado como pertencentes ao mesmo agrupamento, se sua disposição no espaço respeitar uma função de densidade,

pré-parametrizada pelo implementador do método. No DBSCAN agrupamentos ou *clusters* são áreas de alta densidade separadas de outras áreas também densas por regiões de baixa densidade. Densidade é definida pelo método como um conjunto de dados próximos. Possui resultados satisfatórios pois gera agrupamentos indiferentes ao formato geométrico, como é visto na Figura 2.3

Figura 2.3 – Exemplo de Agrupamento feito pelo DBSCAN.



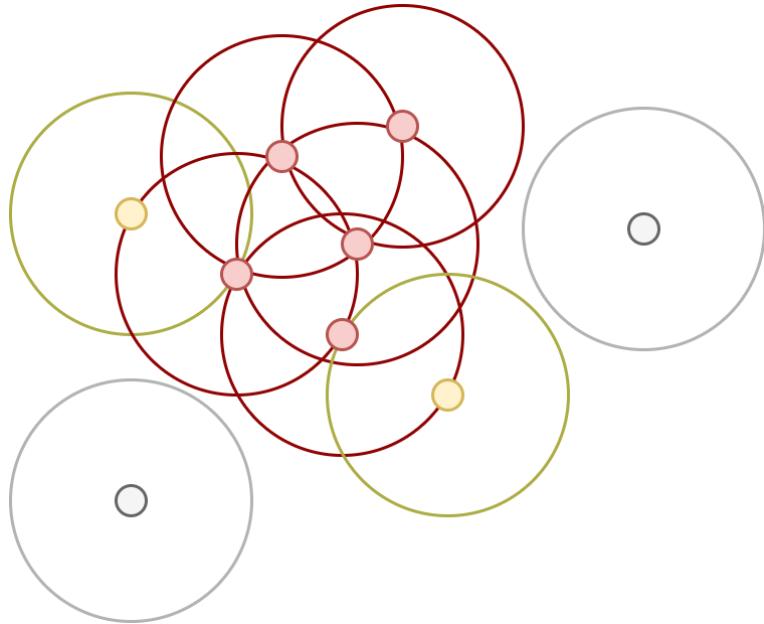
Fonte: <http://yarpiz.com/wp-content/uploads/2015/09/dbSCAN-result.png>

Basicamente, para cada elemento e do conjunto, é verificado seus k vizinhos em um raio ε predefinido no algoritmo, caso e possua $k > v_{min}$ onde v_{min} é a quantidade mínima de vizinhos também predefinida no algoritmo, e é considerado um elemento de centro e dá início a um novo agrupamento. Após isso, para cada vizinho de e é feito o mesmo processo, porém, caso satisfaçam a condição, são adicionados ao agrupamento de e . Elementos não associados com nenhum grupo ao final do processo são considerados *outliers* ou ruído.

A Figura 2.4 demonstra como ficaria um conjunto de dados após a segmentação realizada pelo DBSCAN, utilizando um $v_{min} = 2$, onde pontos vermelhos são elementos de centro do grupo, pontos amarelos considerados elementos de borda, e pontos cinza são *outliers* ou ruído.

O DBSCAN é altamente dependente dos parâmetros ε e v_{min} para realizar uma boa segmentação dos grupos. Por isso, é necessário estimar uma medida ótima para ambos os parâmetros se o conjunto não possuir uma escala bem conhecida, ou uma distribuição espacial muito variada, e definir ambos os parâmetros pode ser algo complexo de se fazer. Outro agravante é sua complexidade computacional de $O(n^2)$, pois, dependendo do tamanho do conjunto, seu processamento pode se tornar algo inviável. Porém, pode

Figura 2.4 – Exemplo prático de funcionamento do DBSCAN.



Fonte: Imagem de autoria do autor.

possuir ótimo funcionamento em conjuntos de amostragem reduzidos e discretizados em um espaço conhecido, como por exemplo é feito no método de *VoxelGrid*, melhor explicado na seção 3.3.2.

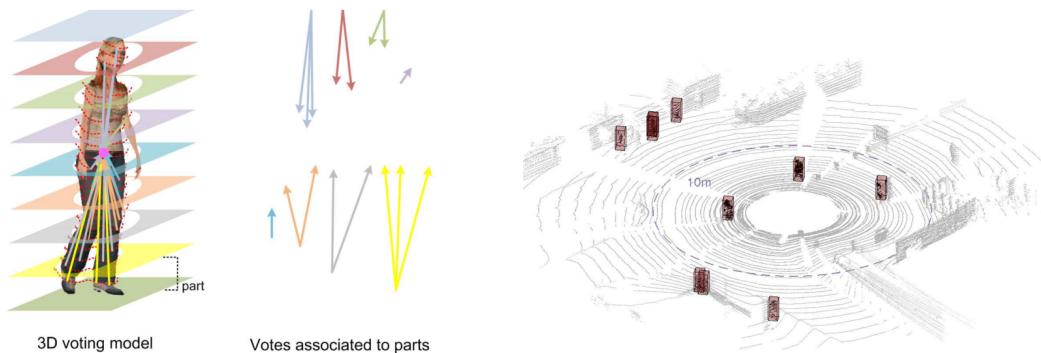
2.1.2 JDC

Jump Distance Clustering (JDC) ([SPINELLO et al., 2010](#)), Utiliza um conceito de segmentação de lasers de duas dimensões em um ambiente tridimensional. Supõe-se que nuvens de pontos tridimensionais podem ser pensadas como camadas de pontos bidimensionais dispostos em diferentes níveis relativos a um terceiro eixo, assim sendo possível estender técnicas de detecção bidimensionais a um ambiente tridimensional.

A ideia do JDC separar a nuvem de pontos em segmentos ou grupos de pontos, subdivididos por nível e distância, é criado um novo segmento denominado S_j cada vez que a distância entre dois pontos consecutivos excede um limiar de distância θ_d onde $j = \{1, \dots, N_i\}$. N_i é o número de segmentos em uma linha i . Após isso, é classificado cada segmento individualmente, e então, por final, é agrupado os segmentos de diferentes níveis em um só objeto. No trabalho de ([SPINELLO et al., 2010](#)) essa abordagem é utilizada para identificação de pessoas, como pode ser visto na Figura 2.5.

Na Figura 2.5 é visto a esquerda o *3D voting model* (modelo tridimensional de votação), o qual é desejado localizar na nuvem de pontos, e os vetores associados (Votes associated) a cada segmento em relação ao centro do *3D voting model*, e a direita, uma nuvem de pontos onde as pessoas localizadas por meio do método são demarcadas com

Figura 2.5 – Exemplo de aplicação do JDC



Fonte: Imagens retiradas de ([SPINELLO et al., 2010](#)).

caixas, três pessoas são segmentadas em um raio de 10 metros, e mais cinco no exterior desse raio. Uma vantagem do JDC é que o mesmo possui que alta robustez em relação a oclusão do objeto, visto que não é necessário todas as partes para realizar a segmentação.

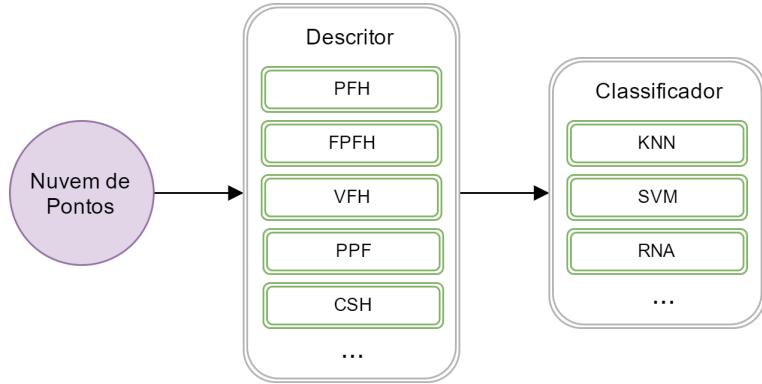
2.2 Descrição e Classificação

A etapa de descrição e classificação visa identificar um grupo de características pertencentes a um conjunto de dados, que sejam capazes de diferenciar esse conjunto de outro, ou dizer que dois conjuntos são semelhantes. Em uma nuvem de pontos, esse processo se dá através de uma análise de um conjunto de pontos no espaço, primeiro é realizada a etapa de identificação das características, realizada por meio de métodos descritores. Métodos descritores são técnicas que reconhecem um conjunto de características de um modelo observável, de modo que seja possível diferenciá-los de outros modelos através de suas características, comparamos as características de um objeto com o conjunto de características que temos de outros modelos, utilizados para treinar o classificador. Assim, é possível classificar o objeto observado, de acordo com o nível de semelhança de características entre o objeto e os modelos de treino. Uma melhor ideia do processo pode ser observada na Figura 2.6, onde uma nuvem de pontos, já segmentada, é processada por algum método descritor PFH, FPFH, etc. Após isso, classificada por um método classificador a escolha do implementador do algoritmo, KNN, SVM, etc. Múltiplos métodos de descrição e classificação podem ser implementados simultaneamente, dependendo da aplicação.

Existem inúmeros métodos de descrição, tanto para imagens bidimensionais, como por exemplo o *Scale Invariant Feature Transform (SIFT)* ([LOWE, 1999](#)) e o *Speeded Up Robust Features (SURF)* ([BAY et al., 2008](#)), como também para imagens tridimensionais,

como *Complex Shape Histogram* (CSH) (BĘDKOWSKI et al., 2016), *Point Feature histogram* (PFH) (RUSU et al., 2008), *Point Pair Feature* (PPF) (DROST et al., 2010). Porém, a escolha de qual utilizar varia de uma aplicação para outra, visto que cada um possui suas próprias peculiaridades. O método para descrição utilizado neste trabalho é baseado em outros dois métodos abordados nas seguintes subseções: (2.2.1), (2.2.2).

Figura 2.6 – Fluxo de funcionamento da classificação em nuvem de pontos



Fonte: Imagem de autoria do autor.

Assim como métodos de descrição, também é necessário a escolha de um bom método de classificação dos resultados, de modo que seja possível retirar a informação desejada do descriptor. Os dois métodos comumente utilizados em trabalhos de classificação de nuvem de pontos e abordados neste capítulo são *Support Vector Machines* (SVM) (CORTES; VAPNIK, 1995) e *K-Nearest Neighbors* (KNN) (ALTMAN, 1992), que serão descritos mais a frente.

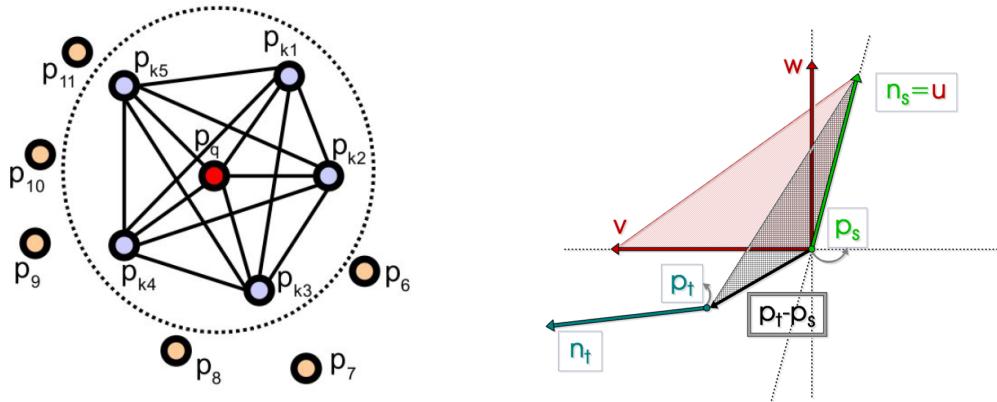
2.2.1 PFH

O descriptor *Point Feature histogram* (PFH) (RUSU et al., 2008), tem como objetivo a identificação de um conjunto de características onde pontos tridimensionais que representem objetos geométricos primitivos possam ser facilmente identificados. O PFH é invariante à rotação, posição e escala do objeto. Para representar uma assinatura para a descrição dessa nuvem, utiliza-se um histograma multidimensional de valores.

A identificação de características do PFH se baseia na relação entre um ponto p e seus k vizinhos (Figura 2.7a), e a estimativa do vetor normal de superfície de cada ponto. Assim, o descriptor realiza a captura das variações no conjunto de pontos da nuvem.

Para cada ponto p_q da nuvem, é calculado um círculo de raio r . Para cada par de pontos $\langle p_{ki}, p_{kj} \rangle$, onde $j = i = \{1, \dots, n\}$ e n é o numero de vizinhos do ponto dentro do círculo de raio r , é calculado um conjunto de relações entre os pares de pontos, para a construção do histograma final.

Figura 2.7 – Modelo de execução do PFH



a) Raio de influência de um ponto no PFH b) Associação de normais do PFH

Fonte: Imagens retiradas de ([RUSU et al., 2008](#)).

Para calcular o conjunto de relações entre dois pontos, o descritor PFH utiliza um espaço de coordenadas de *Darboux*. Para melhor entendimento e sua aplicação para pontos em um espaço tridimensional veja ([WAHL; HILLENBRAND; HIRZINGER, 2003](#)). Conforme mostrado na Figura 2.7b, dado dois pontos p_s e p_t , e suas normais \vec{n}_s e \vec{n}_t definimos os vetores u , v e w como:

$$u = \vec{n}_s$$

$$v = u \times \frac{(p_t - p_s)}{\| p_t - p_s \|}$$

$$w = u \times v$$

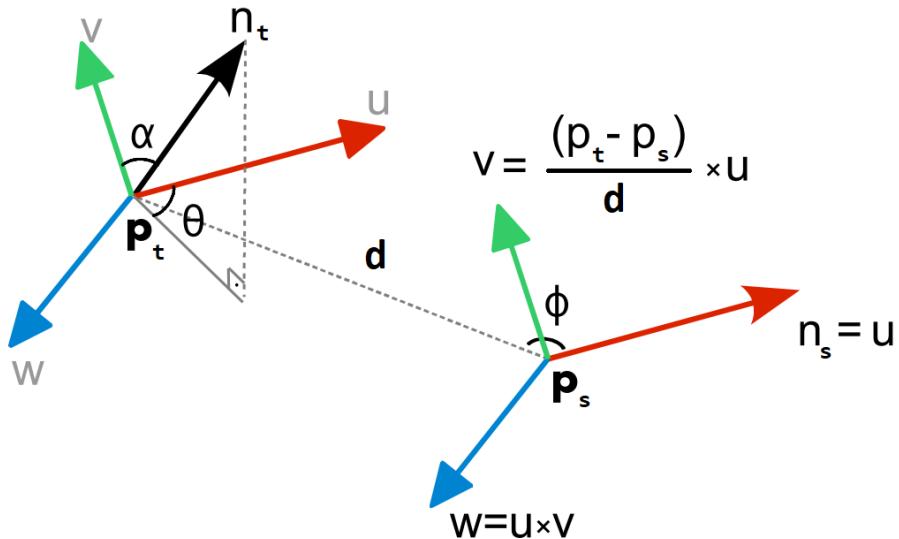
Após isso são calculados quatro parâmetros $\langle d, \alpha, \phi, \theta \rangle$ para cada par (p_s, \vec{n}_s) e (p_t, \vec{n}_t) dentro do raio r , a quádrupla referente ao par é discretizada em um histograma segundo o método citado em ([WAHL; HILLENBRAND; HIRZINGER, 2003](#)), $\langle d, \alpha, \phi, \theta \rangle$ (Figura 2.8) são respectivamente definidos como:

$$d = \| p_t - p_s \|$$

$$\alpha = v \cdot \vec{n}_t$$

$$\phi = \frac{(p_t - p_s)}{d}$$

$$\theta = \arctan(w \times \vec{n}_t, u \times \vec{n}_t)$$

Figura 2.8 – Visualização dos componentes $\langle \alpha, \phi, \theta, d \rangle$ 

Fonte: Imagens retiradas de ([RUSU et al., 2010](#))

2.2.2 FPFH

O *Fast Point Feature Histogram* ([RUSU; BLODOW; BEETZ, 2009](#)) foi desenvolvido com o intuito de melhorar o desempenho do PFH. Devido ao fato do descritor PFH realizar o cálculo entre todos os pares de pontos em um raio r do ponto de análise, a sua complexidade computacional é de $O(k^n)$, onde k é o número de vizinhos e n o número de pontos da nuvem. O PFH pode apresentar um grande gargalo em aplicações de tempo real ou onde o conjunto observável for muito grande, e, assim, para melhorar o desempenho do mesmo, foi desenvolvido o FPFH.

O descritor FPFH possui complexidade computacional $O(nk)$, e possui grande parte da capacidade discriminativa do PFH. Seu funcionamento é muito semelhante, com algumas pequenas nuances, que são:

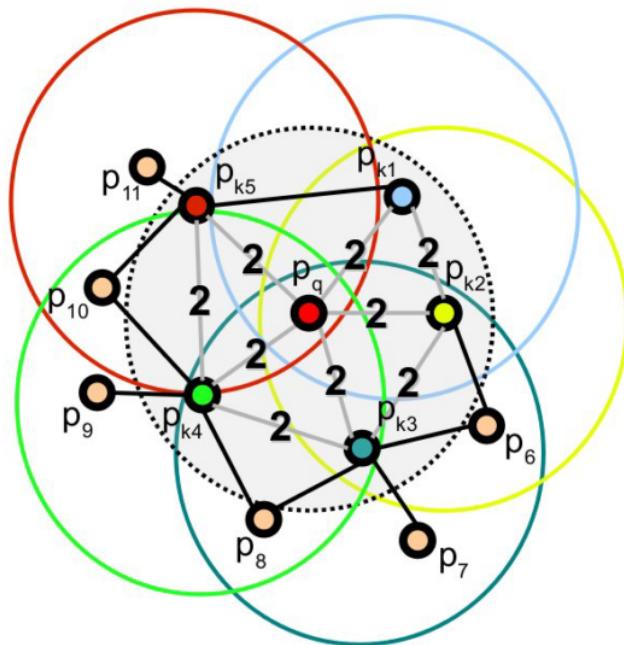
1. Na primeira etapa, para cada ponto p_q , é computado somente as relações entre $\langle p_q, p_j^k \rangle$, onde p_j^k são os pontos vizinhos a p_q . Esse conjunto de dados é chamado de *Simplified Point Feature Histogram* (SPFH).
2. Então, para cada ponto, é novamente determinado seus k vizinhos e utilizado os valores SPFH como peso para o histograma final de p , chamado de FPFH, definido por:

$$FPFH(p) = SPF(p) + \frac{1}{k} \sum_{i=1}^k \frac{1}{\omega_k} \cdot SPF(p_k)$$

Onde o peso ω_k representa a distância entre o ponto p e ponto vizinho p_k no espaço. E $SPF(p)$ a função que infere o valor de p baseado no SPFH.

A diferença básica do FPFH é que, ao invés de calcular todas as possibilidades de pares de pontos em um círculo de raio r (Figura 2.7a), só é calculado entre o ponto central p_q e seus vizinhos. Na Figura 2.9 é visto os pontos P_k , onde os círculos representam o alcance de cada ponto, o valor de peso da conexão dos pontos até o ponto central, e a conexão que é feita entre os pontos. Após a etapa de cálculo do histograma do ponto central, o peso do SPFH é adaptado com base no SPFH dos vizinhos. Desta forma, ele não utiliza todas as informações que definem a geometria em torno de um ponto, porém tem uma estimativa aproximada. Além disso, seu histograma é simplificado, e apesar de prejudicar na eficácia do método, seu uso se justifica por seu ganho em performance, podendo ser usado em aplicações de tempo real.

Figura 2.9 – Raio de influência de um ponto no FPFH



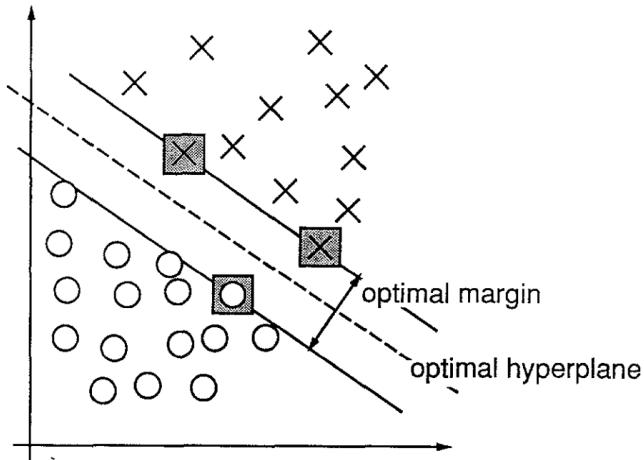
Fonte: Imagem retirada de ([RUSU; BLODOW; BEETZ, 2009](#))

2.2.3 SVM

Support Vector Machines (SVM) ([CORTES; VAPNIK, 1995](#)), é um método de aprendizagem de máquina supervisionado para classificação e análise de dados. A SVM

possui uma classificação binária, definindo-se duas classes de objetos. Dado um conjunto de treino, cada elemento é definido como pertencente a uma das duas classes do método, como visto na Figura 2.10.

Figura 2.10 – Visualização da classificação SVM



Fonte: Imagem retirada de ([CORTES; VAPNIK, 1995](#))

Basicamente, a SVM constrói um hiperplano em um espaço n-dimensional, o qual fica localizado na intermediação dos dois conjuntos. A SVM foi originalmente desenvolvida para ser um classificador linear, porém é possível realizar uma classificação não linear utilizando o método *Kernel*.

Dado um conjunto de n valores $C = \{(\vec{x}_1, y_1), \dots, (\vec{x}_n, y_n)\}$, onde cada \vec{x}_i é um vetor de k dimensões e y_i assume o valor de 1 ou -1, o qual indica a qual classe \vec{x}_i pertence. O objetivo da SVM é encontrar o hiperplano que divide o grupo de valores em suas respectivas classes de modo que a distância do hiperplano para o ponto mais próximo de cada classe seja maximizada.

Um hiperplano pode ser definido desde que o conjunto de pontos \vec{x}_i satisfaça a equação $\vec{w} \cdot \vec{x} - b = 0$ onde \vec{w} é o vetor normal ao hiperplano.

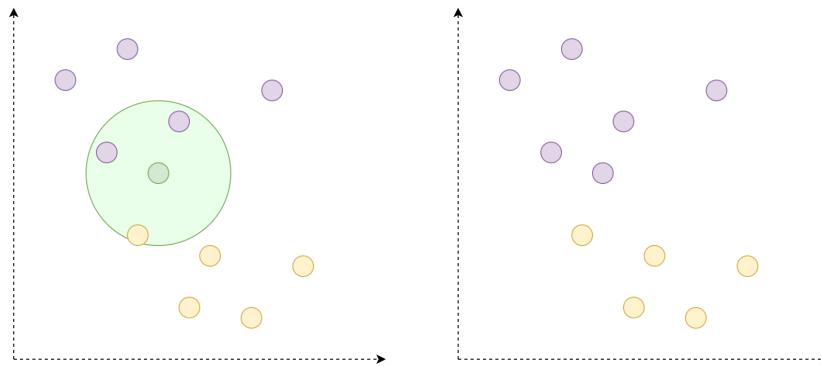
2.2.4 KNN

K-Nearest Neighbors (KNN) ([ALTMAN, 1992](#)) é um método muito simples, porém extremamente eficaz no meio computacional, e tem uso em diversas áreas como aprendizagem de máquina, reconhecimento de padrão, associação de dados, etc.

O KNN é um modelo de classificação com aprendizagem supervisionada, ou seja, dado um conjunto de treino e suas respectivas classes, novas ocorrências são classificadas com base no conjunto pré-estabelecido, e, portanto, é necessário que a princípio seja definido um conjunto inicial de treino $T = \{(x_1, y_1), \dots, (x_n, y_n)\}$, onde x_i é um vetor

de d dimensões, e y_i a classe a qual o elemento x_i pertence. É definido então o valor k de vizinhos mínimos, e após isso, quando um novo elemento é inserido, ele é classificado de acordo com a classe mais comum entre seus k -vizinhos. O processo pode ser visto na Figura 2.11, onde é definido um conjunto A (cor roxa) e um conjunto B (cor amarela), após inserido um novo elemento (cor verde), é buscado os três elementos mais próximos, e com base na classe dos mesmos, é definido a classe do novo elemento.

Figura 2.11 – Exemplo de classificação KNN com $k = 3$



Fonte: Imagem de autoria do autor.

O KNN possui a vantagem de ser um método não paramétrico, necessitando assim somente de um conjunto predefinido de treino. Porém, uma desvantagem do KNN é que associações errôneas se mantêm. Por outro lado, possui a vantagem de possuir uma implementação simples e de baixa complexidade computacional (CHO; BAEG; PARK, 2014).

Outra forma de realizar a classificação através do KNN é atribuir um peso w_{ni} inversamente proporcional a distância de cada vizinho em relação ao elemento, de modo que o elemento é classificado de acordo com o grupo de maior peso.

$$\sum_{i=1}^n w_{ni} = 1$$

2.3 Associação e Rastreamento

A etapa de associação e rastreamento tem como função informar onde o objeto está na cena, tanto espacialmente, como temporalmente. Ou seja, possui a tarefa de associar um objeto o_1 no tempo t_n com outro objeto o_2 no tempo t_{n+1} e confirmar com determinado grau de certeza que o_1 e o_2 são a representação do mesmo objeto captado pelo sensor em instantes de tempo t_i diferentes.

2.3.1 JPDA

Joint Probabilistic Data Association (JPDA) é um método de realizar associação e rastreamento de objetos simultaneamente. É baseado no modelo probabilístico Bayesiano, em que se estima a correspondência entre as características observadas, e os objetos a serem rastreados, de modo que seja gerado uma matriz de hipóteses para todas as possíveis associações.

São escolhidas as associações com maior probabilidade. Cada objeto deve possuir um modelo dinâmico e de medidas lineares, e no caso de um modelo não linear, é recomendado uma linearização como é detalhado em ([COX, 1993](#)).

Esse método é amplamente utilizado em abordagens de DATMO como em ([SCHULZ et al., 2001](#)) e ([ALMEIDA; ALMEIDA; Rui Araujo, 2005](#)), porém, possui algumas desvantagens como, por exemplo, o fato de se considerar somente modelos lineares, e a suposição de linearidade não é realista sempre. Outra desvantagem é a necessidade de que o número de objetos móveis seja conhecido e constante. Quanto à linearização dos dados, esta pode ser realizada, contanto que as não linearidades do modelo sejam fracas na região de interesse.

2.3.2 MHT

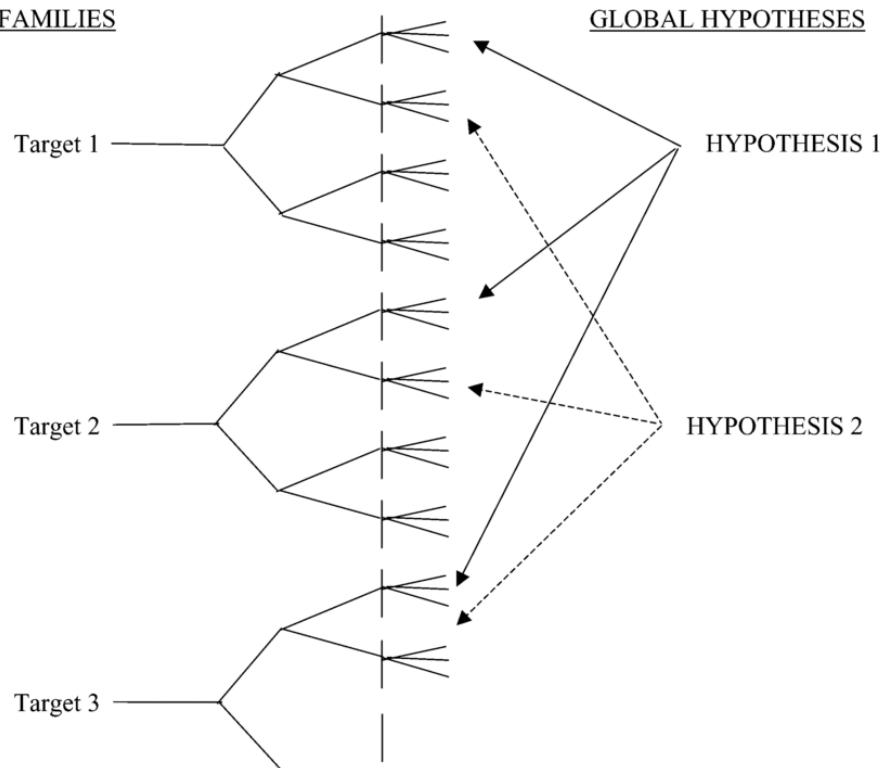
O método *Multiple Hypothesis Tracking* (MHT) possui uma ideia de funcionamento semelhante a do JPDA, porém, nesse caso, são aceitas múltiplas hipóteses de rastreamento de alvos entre objetos e medidas, enquanto que o JPDA assume apenas uma hipótese de associação.

O MHT assume que mais de uma hipótese pode estar correta, e assim todas as possibilidades hipotéticas são armazenadas. A informação sensorial armazenada é utilizada para que se possa realizar alguma tomada de decisão, em relação a quais hipóteses devem ser mantidas e quais devem ser descartadas. Portanto, o MHT utiliza medições passadas para realizar sua conjectura, e novas hipóteses são geradas pela associação dos conjuntos de hipóteses predeterminadas.

As hipóteses são arranjadas em uma estrutura de árvore, conforme visto na Figura [2.12](#). Uma vantagem do método é que se caso uma hipótese errada tenha levado o resultado a um falso positivo, é possível realizar todo o trajeto para definir uma nova hipótese. Porém, uma desvantagem do MHT é o seu alto consumo computacional, o que torna mais complexo seu uso em aplicações de tempo real, visto que o conjunto de dados é muito extenso, e de que o método constantemente gera muitas hipóteses, o tornando mais lento conforme o passar do tempo. O mesmo problema pode ser encontrado caso seja necessário a associação de muitos objetos simultaneamente, pois a árvore de hipóteses se torna bastante extensa e complexa de se processar.

O método MHT é comumente utilizado em trabalhos da área, como por exemplo

Figura 2.12 – Formação da árvore hipóteses



Fonte: Imagem retirada de ([BLACKMAN, 2004](#))

em ([Chieh-Chih Wang; THORPE; THRUN, 2003](#)). A Figura 2.13 demonstra a utilização do método na realização de rastreamento de pedestres no campo de visão de um carro autônomo, onde é mostrada uma visão superior, em que a caixa azul representa o carro, os pontos em vermelho referentes a paredes, e as caixas em verde são conjuntos de pontos de possíveis pedestres.

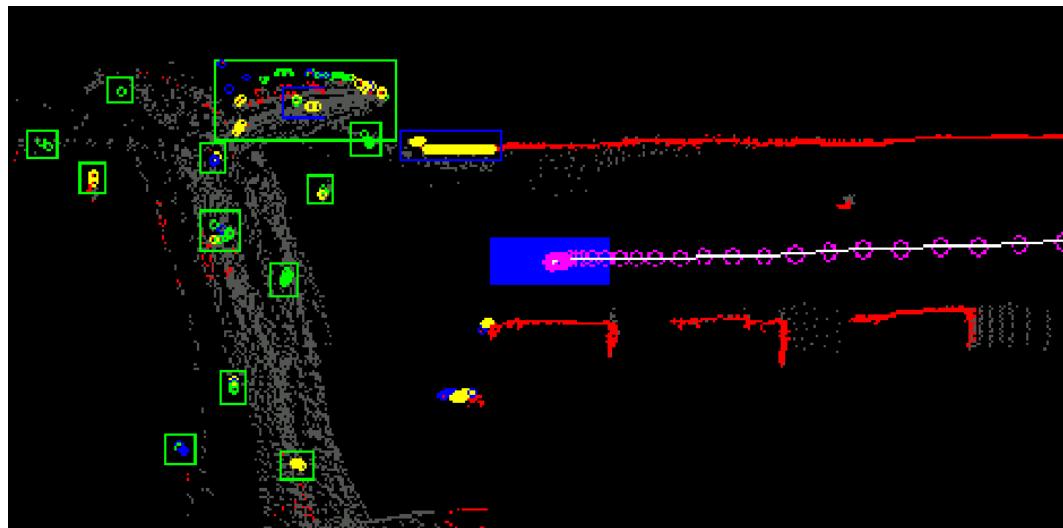
Uma abordagem para permitir seu uso em aplicações de tempo real seria limitar a árvore de hipóteses com algum algoritmo de poda (*pruning algorithm*). Consulte ([COX, 1993](#)) e ([BLACKMAN, 2004](#)) para uma melhor revisão do método.

2.3.3 Filtro de Partículas

O Filtro de Partículas ([MORAL, 1996](#)) pertence a uma classe de algoritmos denominada de *Métodos Sequencias de Monte Carlo*, que é apresentada na literatura por diversas outras nomenclaturas, como filtros *bootstrap*, condensação, filtro de Monte-Carlo. O Filtro de Partículas funciona como um método probabilístico que visa uma resolução para problemas, em que se deve estimar o estado interno de sistemas dinâmicos com observações parciais.

Basicamente, o objetivo do filtro é realizar a inferência de estados a partir de um

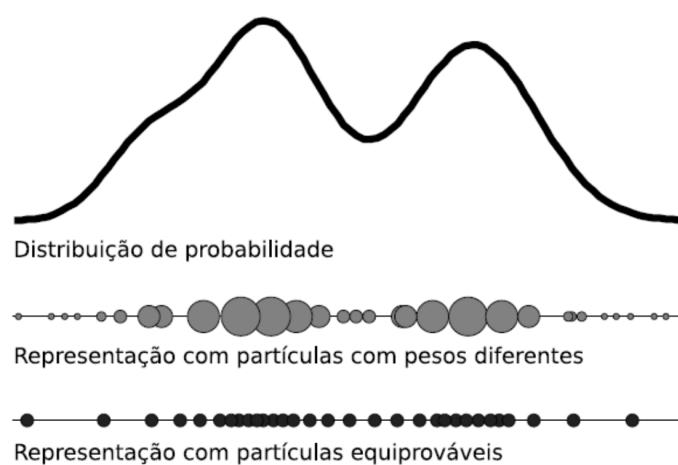
Figura 2.13 – Aplicação do MHT em detecção de pedestres



Fonte: Imagem retirada de ([Chieh-Chih Wang; THORPE; THRUN, 2003](#))

conjunto de observação incompleto. Consiste na utilização de um conjunto de amostragem gerado aleatoriamente, onde cada elemento do conjunto é uma partícula, com probabilidade de observação conhecida para aproximar o valor da função de interesse através de seu valor estimado. A probabilidade a posteriori é representada não mais por uma Gaussiana, mas por uma distribuição discreta de probabilidade, definida através de um conjunto amostral de partículas e seus respectivos pesos, conforme visto na Figura 2.14.

Figura 2.14 – Distribuição de partículas



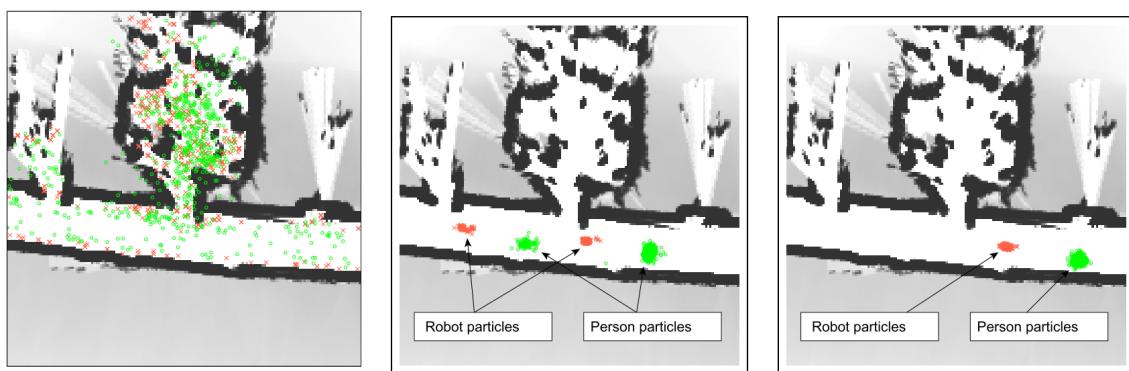
Fonte: Imagem retirada de ([AMARAL et al., 2015](#))

Simplificando o conceito, é realizada uma distribuição aleatória de probabilidade de estimativa do estado do objeto observado. Após cada nova leitura dos dados, novas distribuições são feitas com base nas novas medições e no conjunto de probabilidades do

tempo anterior, fazendo assim com que o algoritmo converja no provável estado do objeto.

Sua aplicação na área da robótica é bastante comum, como mostrado em (THRUN, 2012). Sua aplicação se justifica, pois é capaz de amenizar o problema de oclusão de objetos e leituras falhas de sensores, visto que funciona por inferência probabilística, e possui uma excelente eficácia, devido ao fato de não ser um método determinístico. Um exemplo de aplicação pode ser visto na Figura 2.15, que se trata de uma visão superior de uma rua, onde é visto um conjunto inicial de partículas convergindo para o objeto que se deseja rastrear.

Figura 2.15 – Convergência do filtro de partículas no rastreamento



Fonte: Imagem retirada de (THRUN, 2012)

3 Desenvolvimento

Neste capítulo serão apresentadas as metodologias utilizadas em cada etapa do desenvolvimento do sistema. O sistema proposto utiliza um conjunto de métodos para a realização da identificação e rastreamento de um objeto, representado por uma nuvem de pontos captada pelo campo visual do sensor Kinect da Microsoft. As próximas seções detalham cada etapa deste processo.

Para o desenvolvimento do sistema, foi utilizado o ROS ([ROS, 2016](#)), distribuição *jade*. O ROS é uma coleção de ferramentas, bibliotecas e convenções que visam a simplificação do processo de criação de sistemas robóticos. Possui um sistema interno de troca de mensagens que se baseia em uma arquitetura de rede de *publisher/subscriber* ([ROS, 2016](#)) e foi utilizado na construção dos módulos do sistema proposto. Veja mais detalhes na seção [4.1.2](#).

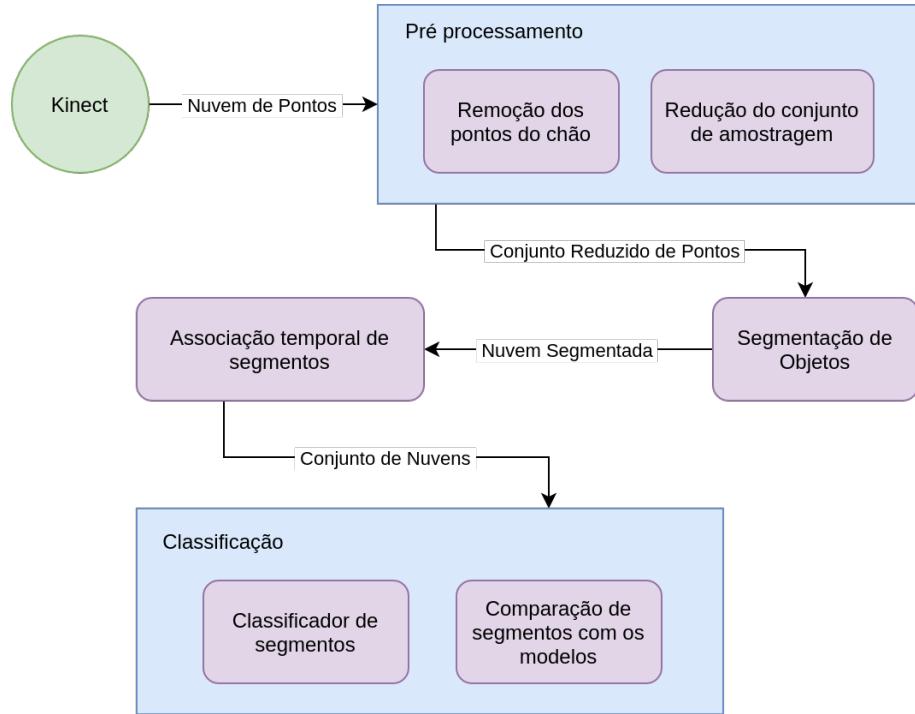
Outra ferramenta utilizada para o desenvolvimento do projeto foi a PCL ([RUSU; COUSINS, 2011](#)) para processamento de nuvem de pontos. A PCL é uma biblioteca de código aberto, e já se encontra nas dependências do ROS por padrão. Mais informações podem ser vistas na seção [4.1.2](#).

3.1 Visão Geral

O sistema proposto desenvolvido neste trabalho opera em quatro etapas. A cada leitura do sensor é realizado o pré-processamento, a segmentação, a associação e a classificação da nuvem de pontos 3D. Na etapa de pré-processamento ocorre a remoção dos pontos 3D do chão e a redução do conjunto de amostragem. Na etapa de segmentação, a nuvem de pontos 3D é segmentada em agrupamentos de pontos usando a distância Euclidiana, sendo que cada agrupamento representa um objeto no ambiente. Na etapa de associação, os objetos observados na varredura atual do sensor são associados aos mesmos objetos observados em varreduras anteriores usando o algoritmo do vizinho mais próximo (*nearest neighbor*). Finalmente, na etapa de classificação, os segmentos são comparados com os modelos predefinidos e após a identificação, o alvo é rastreado. O fluxo de funcionamento do sistema proposto pode ser visualizado na Figura [4.3](#).

Para esse trabalho foi utilizado o Kinect ([Figura 3.2](#)). O Kinect é um sensor RGB-D (*Red Green Blue - Depth*) de baixo custo. Sensores RGB-D se referem a câmeras que além de possuírem o funcionamento convencional, também geram informação de profundidade espacial. Essa informação espacial é denominada de mapa de profundidade (*depth map*) ([Figura 3.3c](#)), que se trata de uma imagem em escala de cinza onde a intensidade da cor

Figura 3.1 – Fluxo de funcionamento do sistema proposto.

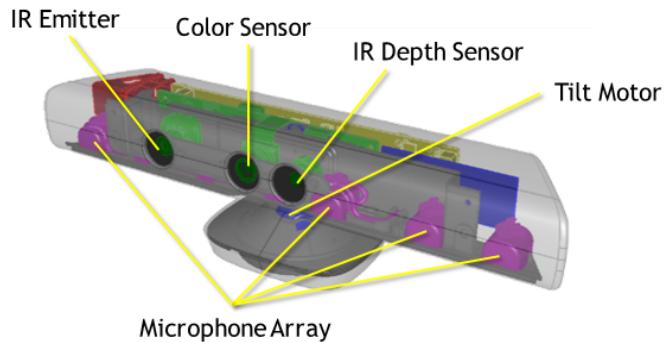


Fonte: Imagem de autoria do autor.

representa sua profundidade em relação ao observador. Portanto, pode ser classificado como um conjunto de pontos p_{ij} que para cada ponto onde $0 \leq i \leq 480$ e $0 \leq j \leq 640$ existe um valor de profundidade d_{ij} onde $0 \leq d_{ij} \leq 255$ em que 0 representa a distância mínima e 255 a máxima.

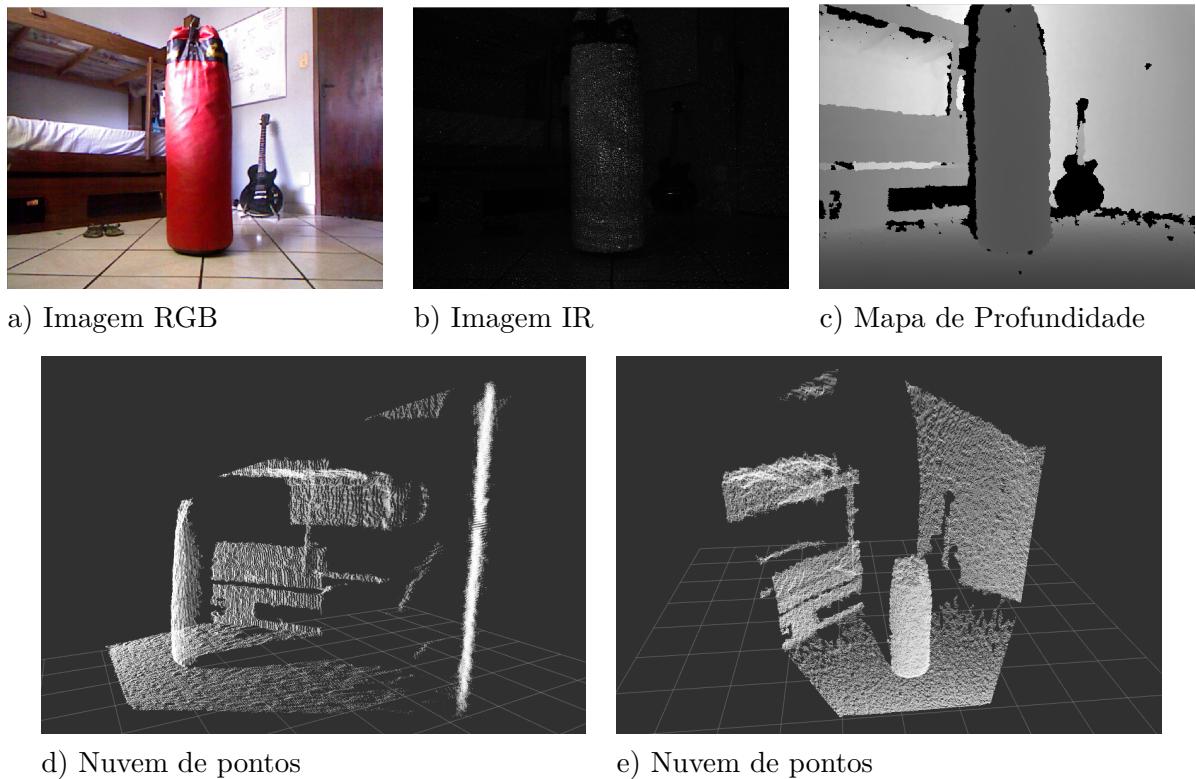
Após uma transformação, o dado é convertido em um conjunto de pontos p_i onde $0 \leq i \leq 307200$. Cada ponto p_i possui uma representação global (x_i, y_i, z_i) , onde o componente z_i é ortogonal ao plano xy e representa a altura de p_i . Esse conjunto de pontos representa uma nuvem de pontos, como visto na Figura 3.3d e Figura 3.3e.

Figura 3.2 – Sensor Kinect da empresa Microsoft.



Fonte: <http://i.microsoft.com/dynimg/IC584396.png>

Figura 3.3 – Processo de extração de dados do Kinect

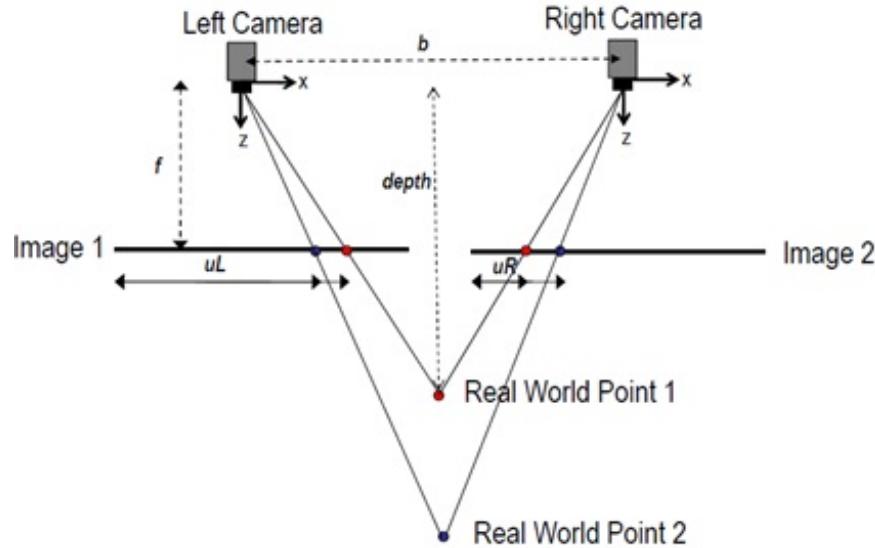


Fonte: Imagem de autoria do autor.

A noção espacial do Kinect funciona através de uma função de disparidade entre o emissor de infravermelho (Figura 3.3b) e o sensor infravermelho de profundidade. Através do ponto de emissão e do ponto de captação é realizado o cálculo da disparidade resultando na profundidade daquele ponto (Figura 3.4). Porém, essa abordagem possui alguns problemas. Devido ao fato de se utilizar infravermelho, ele apresenta um funcionamento ruim em ambientes externos, pois, pelo fato da luz do sol possuir alto índice de infravermelho, o mesmo ilumina o objeto mais que a fonte luminosa do Kinect, causando erro na leitura do sensor. Outro problema é com objetos de superfície reflexiva, onde determinados materiais possuem um coeficiente de reflexão demasiadamente alto, o que ocasiona um cálculo errado de profundidade por parte do sensor, tornando o conjunto de pontos referentes à aquela superfície nulos, como ocorre com a guitarra demonstrado na Figura 3.3c.

Por outro lado, o Kinect é um sensor que pode ser usado em ambientes internos e ambientes noturnos como um excelente nível de precisão e um ótimo custo/benefício se comparado a outros sensores (laser, sonares, etc). Por isso, ele tem sido usado com frequência na robótica, como pode ser visto em (EL-LAITHY; HUANG; YEH, 2012).

Figura 3.4 – Funcionamento da disparidade da câmera.



Fonte: <http://www.depthbiomechanics.co.uk/wp-content/uploads/2012/06/stereo-vision-cams.jpg>

3.2 Pré-processamento

Na etapa de pré-processamento é realizado a modelagem e filtragem do dado, tornando possível as etapas posteriores. Neste trabalho a etapa de pré-processamento é dividida em duas etapas.

1. Remoção do Plano do chão
2. Redução de amostragem do conjunto de dados

O chão é geralmente o plano dominante na maioria das cenas observáveis por um robô terrestre. Como demonstrado por (DOUILlard et al., 2011), a extração do solo melhora significativamente o desempenho da segmentação. A não remoção do chão impossibilita a segmentação de pontos pois é considerado como parte do segmento. A etapa de redução de amostragem do conjunto é opcional, porém se fez necessária devido ao fato de ser essencial a execução do sistema em tempo real.

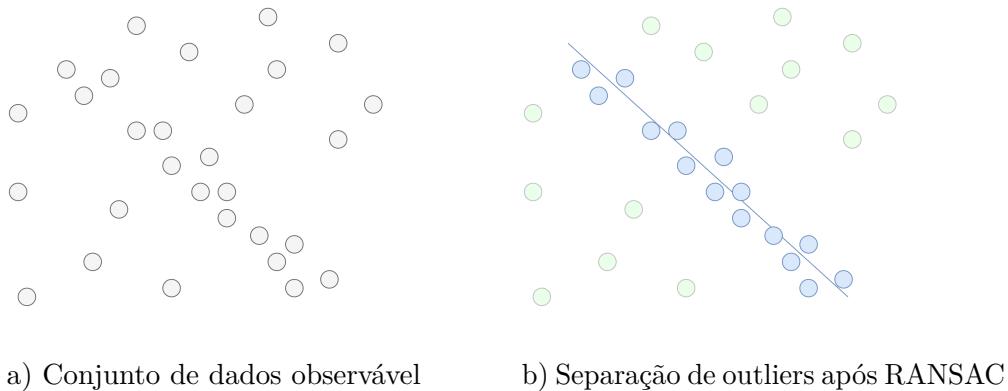
3.2.1 Remoção do Plano do chão

Neste trabalho o *RANSAC* foi utilizado como pré-processamento para a remoção do chão. ***Random Sample Consensus (RANSAC)*** é um método iterativo para estimar parâmetros de um modelo matemático em um conjunto de dados observáveis. O algoritmo foi primeiramente publicado em (FISCHLER; BOLLES, 1981), onde foi utilizado para a

resolução do problema de determinação de localização, onde o objetivo era determinar o ponto no espaço em que uma imagem foi obtida dado um conjunto previamente conhecido de marcadores representados na imagem.

O *RANSAC* é um algoritmo não determinístico, visto que seu resultado depende de determinada probabilidade de acerto, onde essa probabilidade é proporcional ao limite de iterações permitido. O objetivo principal do *RANSAC* é a segmentação de um conjunto de dados observável de modo que seja possível definir os *inliers* (pontos pertencentes ao modelo matemático) e os *outliers* (pontos não pertencentes ao modelo matemático) de certo modelo neste conjunto. Portanto, o *RANSAC* pode ser considerado como um método de detecção de *outliers*, como visto na Figura 3.5, onde demonstra a segmentação de um modelo matemático de uma reta em um conjunto de pontos no espaço.

Figura 3.5 – Exemplo de aplicação do RANSAC.



Fonte: Imagem de autoria do autor.

O *RANSAC* pode ser aplicado em inúmeros conjuntos de dados, e possuir diversas aplicações, desde que o conjunto de dados possua uma forma matemática de representação.

Por se tratar de um método iterativo, a cada iteração o algoritmo converge para a resposta final, sendo ela um falso positivo ou não. O funcionamento do *RANSAC* se baseia em cinco etapas:

1. É selecionado randomicamente um conjunto mínimo necessário para representação do modelo; é chamado esse conjunto de *inliers* hipotéticos.
2. O modelo é ajustado com o conjunto de *inliers* hipotéticos.
3. Todos os outros dados do conjunto são comparados com o modelo ajustado previamente; os que podem ser considerados pertencentes ao mesmo, de acordo com alguma função de perda, são considerados parte do conjunto de consenso.
4. O modelo é considerado razoavelmente bom se uma quantidade suficiente de dados forem classificados como pertencentes ao conjunto de consenso.

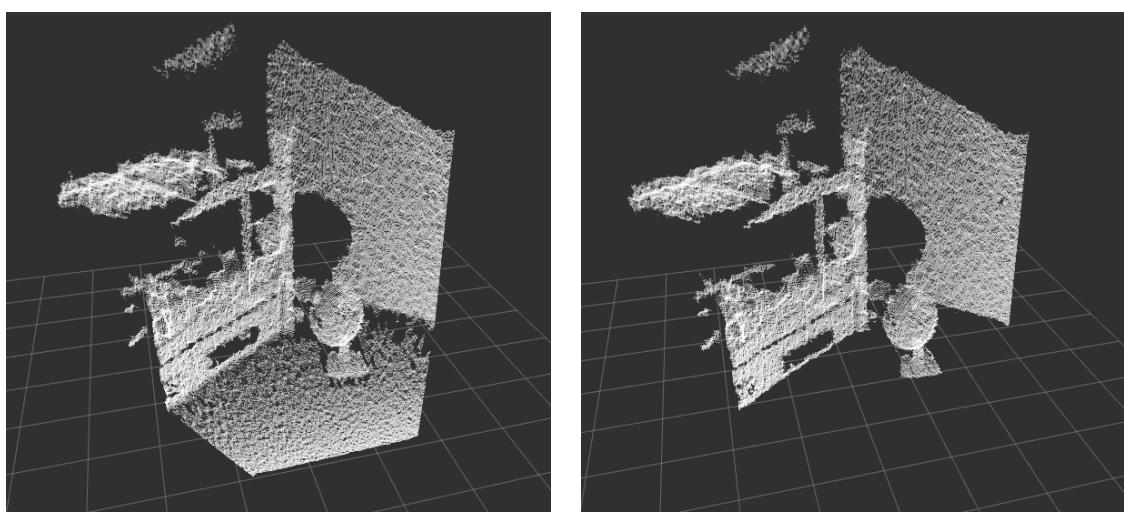
5. Após isso, o modelo é re-estimado utilizando todos os membros do conjunto de consenso.

O método é repetido por um numero predeterminado de vezes, convergindo em um conjunto final que se assemelha ao modelo utilizado. Uma vantagem do *RANSAC* é que ele possui uma capacidade robusta de estimar modelos em conjuntos predefinidos, porém o seu tempo de execução é uma desvantagem devido ao fato de que o algoritmo funciona por exaustão e convergência, fazendo com que um baixo número de iterações possa não gerar o resultado desejado.

Outro problema é que é possível obter apenas um modelo dentro do conjunto observável, portanto caso existam mais de uma representação desse modelo dentro do conjunto é necessário que o método seja executado mais de uma vez.

Como citado anteriormente, é visto em ([DOUILLARD et al., 2011](#)) que a segmentação da nuvem de pontos em grupos que representam objetos no ambiente pode ser melhorada significativamente após a remoção do plano do chão. Por conta disso, foi implementado neste trabalho o processo de remoção do chão, como é demostrado na Figura 3.6. O conjunto de dados observável foi uma nuvem de pontos, e o modelo matemático utilizado foi o de um plano. Dado o fato de que o robô terrestre a ser utilizado navega em ambientes internos, foi assumido que o chão é regular grande parte das vezes, e que portanto sua forma se assemelha a de um plano.

Figura 3.6 – Aplicação do RANSAC para remoção do chão.

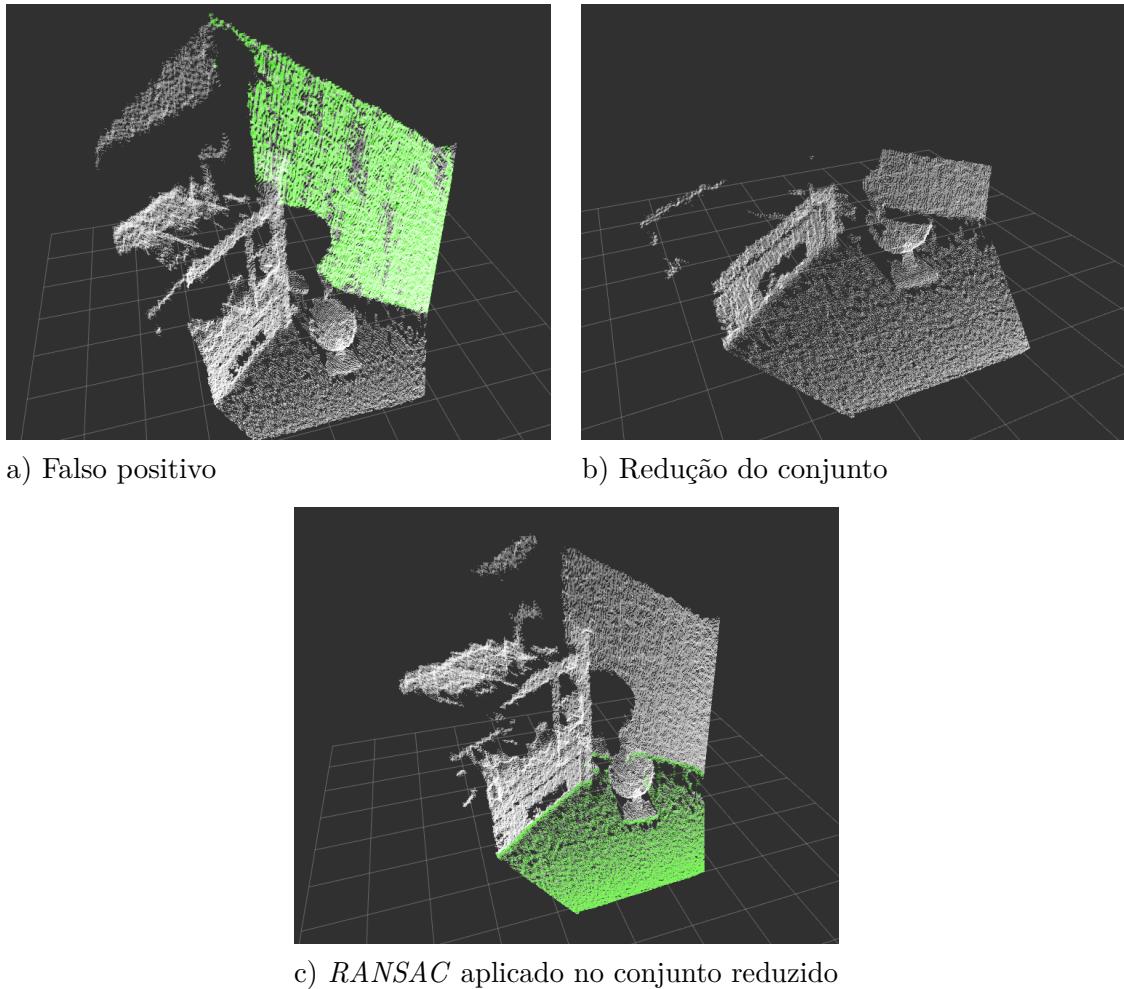


Fonte: Imagem de autoria do autor.

Um problema encontrado na implementação dessa etapa foi a de que em ambientes fechados, paredes entre outros objetos planos que possuem grande quantidade de pontos podem também representar um plano, fazendo com que o *RANSAC* possa convergir em

um falso positivo (Figura 3.7a). Para resolução deste problema foi decidido reduzir o conjunto de dados inicial (Figura 3.7b), de modo que o método converja para o resultado desejado (Figura 3.7c). Para isso foi utilizado somente a metade inferior do conjunto de dados, reduzindo o tempo de execução do método. Desta forma, não é mais necessário iterar sobre todos os pontos 3D, melhorando a precisão, pois estatisticamente, a maior parte dos pontos inferiores da nuvem correspondem ao chão em ambientes planos.

Figura 3.7 – Resolução do problema de falsos positivos na aplicação do *RANSAC*.



Fonte: Imagem de autoria do autor.

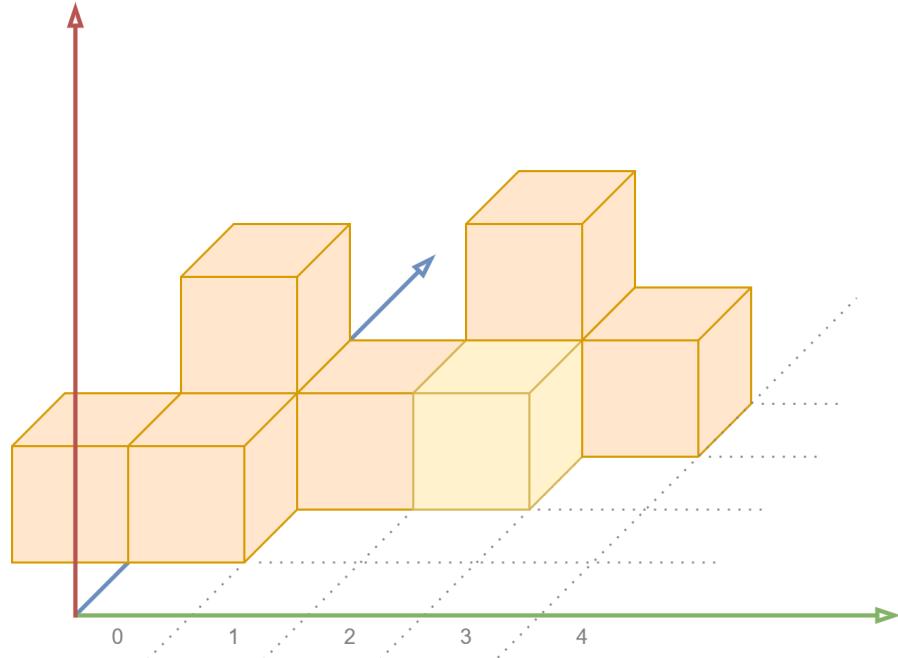
3.2.2 VoxelGrid

O *VoxelGrid* é um método de filtro para nuvens de pontos, onde se é possível diminuir o conjunto de amostragem de forma significativa, mantendo consideravelmente a semelhança do conjunto filtrado com o original. O método de *VoxelGrid* possui implementação na PCL ([RUSU; COUSINS, 2011](#)).

Um *voxel* é uma representação de um valor em um espaço tridimensional, basicamente, uma discretização de um espaço tridimensional contínuo, conforme visto na Figura

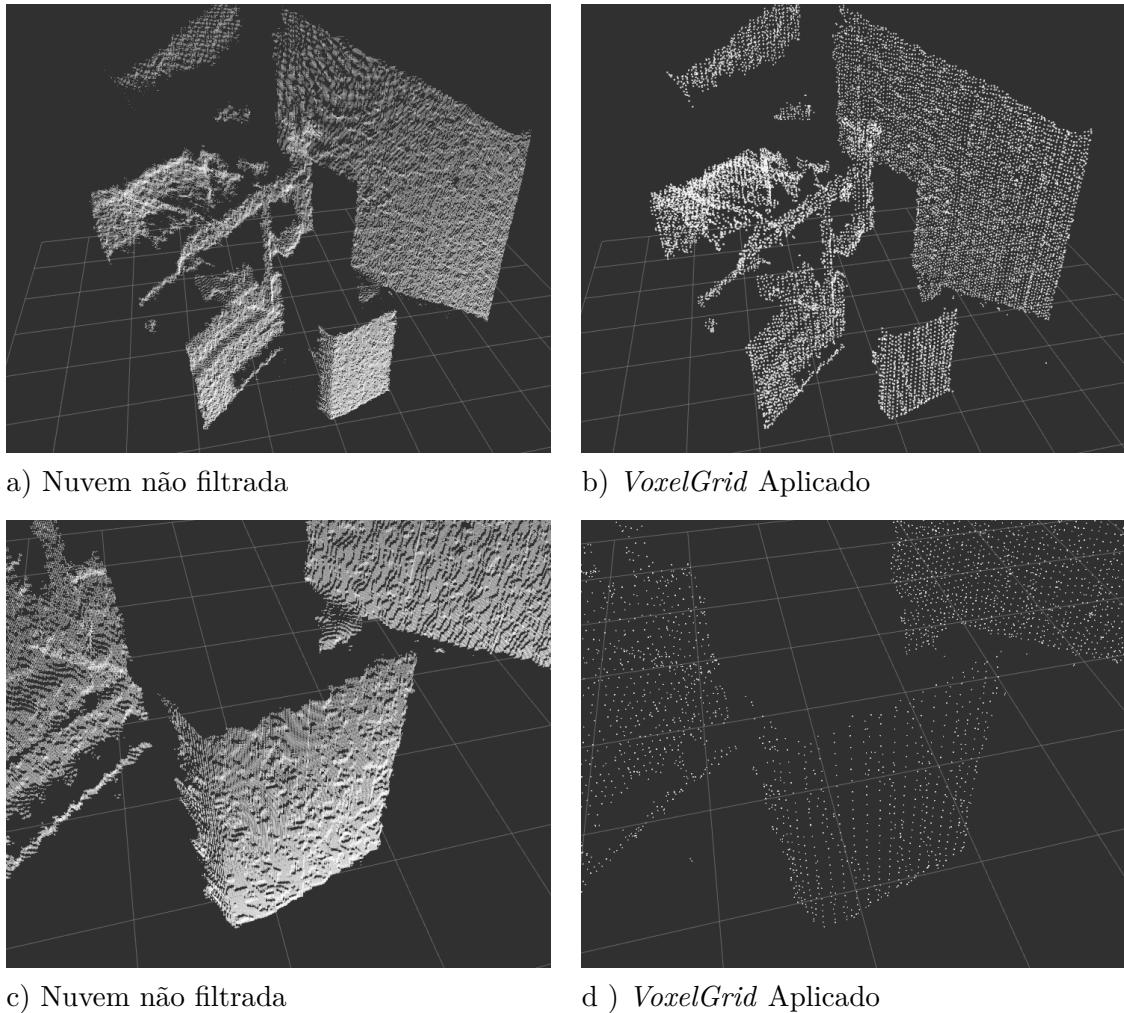
3.8. Seu conceito se assemelha a de um pixel, são igualmente espaçados em um eixo, eixos podem possuir espaçamentos diferentes, e são representados por posições em relação ao centro, ao invés de possuírem uma coordenada central.

Figura 3.8 – Representação de uma grade de *voxel*



Fonte: Imagem de autoria do autor.

O Filtro *VoxelGrid*, ou grade de *voxel* é um método que discretiza os pontos de uma nuvem dentro de um intervalo contínuo. Dado um conjunto de pontos de entrada, o espaço tridimensional é dividido em blocos de tamanho predefinido, onde cada bloco é considerado um *voxel*. E para cada ponto é definido o *voxel* ao qual ele é pertencente. Após isso, é calculado o centroide baseado em todos os pontos próximos, resultando assim em uma nuvem onde cada ponto é referente a um *voxel* e o seu centro uma aproximação da centroide de todos os pontos pertencentes aquele *voxel*. Desta forma, é gerado um conjunto muito menor de dados para processar, conforme visto na Figura 3.9. Para o trabalho em questão foi utilizado *voxels* de 3cm^3 .

Figura 3.9 – Aplicação do *VoxelGrid* na nuvem de pontos

Fonte: Imagem de autoria do autor.

3.3 Segmentação

A segmentação é uma etapa de extrema importância, pois a partir dessa etapa é possível extrair subconjuntos de dados do conjunto principal onde os dados desses subconjuntos possuem características semelhantes. Tais características a serem levadas em conta são predeterminadas pela segmentação, tornando um grande e desconhecido conjunto de dados em inúmeros conjuntos menores, os quais podem ser mais facilmente analisados. No que se refere a projetos de visão computacional, segmentos são caracterizados como grupos de dados gráficos, que possuem características semelhantes, como posicionamento ou cor. E em tarefas de percepção autônoma, esta etapa é essencial, como é demostrado em (BOUCHER, 2012). Outra afirmação verdadeira é a de que uma segmentação a priori melhora a classificação em aplicações de visão computacional, e isso é demonstrado em (MALISIEWICZ; EFROS, 2007). Portanto, o conjunto e seus subconjuntos são definidos como:

Dado uma nuvem de pontos, representado por um Conjunto N de pontos p :

$$N = \{p_1, p_2, \dots, p_{n-1}, p_n\}$$

Ao aplicar a função segmentação $S()$ no conjunto N é obtido um conjunto de segmentos de nuvem sn onde:

$$S(N) = \{sn_1, sn_2, \dots, sn_{k-1}, sn_k\} \Rightarrow (\forall sn)(sn \subset N \Leftrightarrow (\forall p)(p \in sn \Rightarrow p \in N))$$

3.3.1 Segmentação de objetos

Para a segmentação de objetos neste trabalho foi utilizada uma abordagem simples, porém eficiente de agrupamento por distância Euclidiana. Essa abordagem é utilizada em inúmeros trabalho como (BOUCHER, 2012), (AMARAL et al., 2015), (TELLAECHE; MAURTUA, 2014), (ALI; FIGUEROA, 2012), (HAUSMAN et al., 2013), e demonstra ótimos resultados quanto ao objetivo de agrupar pontos pertencentes ao mesmo objeto no campo visual do sensor.

A função de segmentação é constituída de basicamente por duas etapas.

1. Definir uma característica mensurável.
2. Definir uma política de agrupamento.

Primeiramente é necessário mensurar as características, para após isso, agrupar os semelhantes. O modelo de agrupamento por distância Euclidiana, como o próprio nome diz, utiliza o conceito de distância Euclidiana entre dois pontos como característica mensurável, e como política para realizar o agrupamento é utilizado a ideia de menor distância, onde elementos que possuem a menor distância entre si, são definidos como pertencentes ao mesmo segmento. A ideia do modelo Euclidiano de distância é definida do seguinte modo: A distância entre dois elementos p do conjunto N é dado por:

$$Distância = \sqrt{\sum_{i=1}^n (p_i^a - p_i^b)^2}$$

Em que i corresponde a dimensão do ponto. Para a aplicação no trabalho, é definido portanto uma nuvem de pontos como o conjunto de dados, onde cada ponto é o elemento do conjunto com valores internos. Cada ponto possui um conjunto de três coordenadas $\{x, y, z\}$ como mostrado anteriormente na seção 3.1. Assim, a distância entre eles é definida como:

$$Distância = \sqrt{(x_a - x_b)^2 + (y_a - y_b)^2 + (z_a - z_b)^2}$$

Após isso, é feito o agrupamento dos pontos de acordo com o seguinte algoritmo:

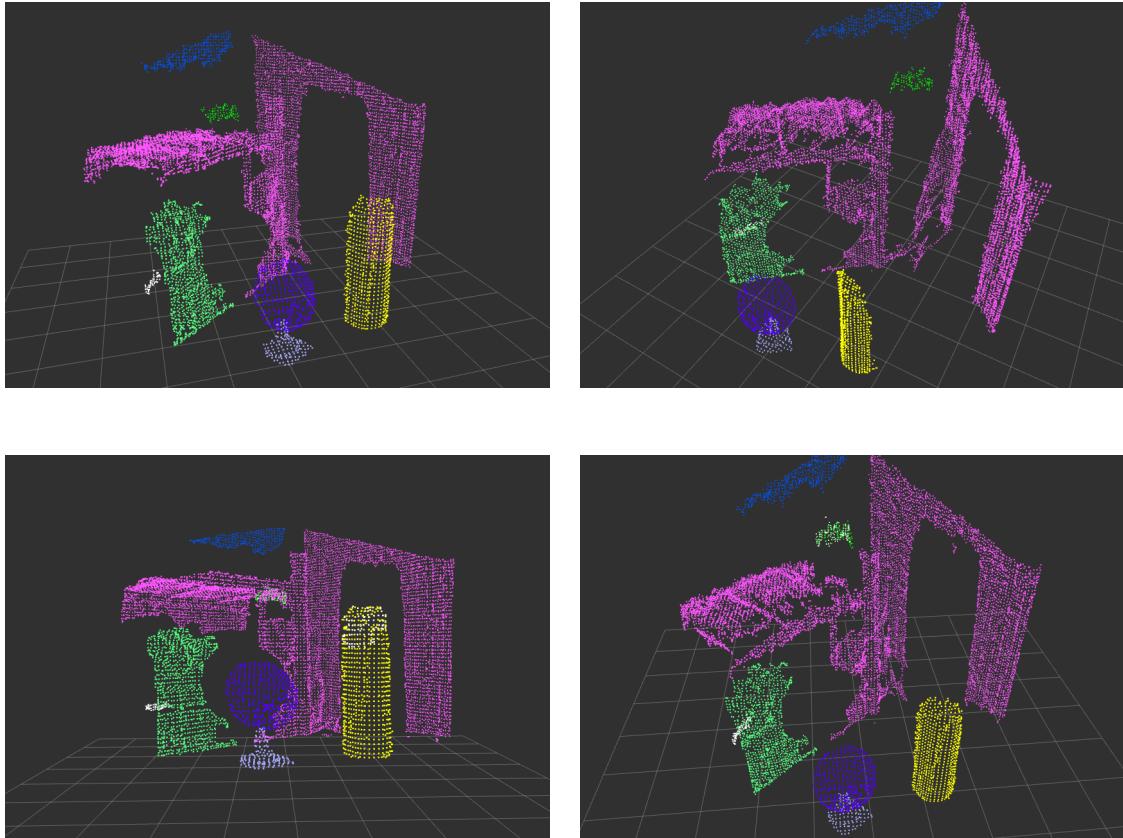
1. É criado uma estrutura de dados de Árvore K-D para o conjunto de dados referente a nuvem de pontos N .
2. É criada uma lista de conjuntos vazia C e uma fila de pontos a serem checados Q .
3. Então, para cada ponto $p_i \in N$ é executado as seguintes etapas:
 - Adiciona p_i a fila de pontos Q .
 - Para cada ponto $p_i \in Q$ é feito o seguinte:
 - É buscado o conjunto de pontos P_k^i vizinhos de p_i dentro de um raio $r < d_{limiar}$, onde d_{limiar} é um parâmetro estático definido anteriormente. Neste caso, d_{limiar} é a distância mínima entre pontos considerados vizinhos.
 - Para cada ponto vizinho $p_k^i \in P_k^i$, verifica se o ponto já foi processado, se não, é adicionado a fila Q .
 - Quando todos os elementos do conjunto Q forem processados, Q é adicionado ao conjunto C e Q é limpa para se tornar uma lista vazia.
4. O algoritmo termina quando todos os pontos $p_i \in N$ são processados e pertence a algum subconjunto $c_i \subset C$.

Árvore K-D (*Árvore K-Dimensões*) é uma estrutura de dados de particionamento espacial desenvolvida com o objetivo de organizar e facilitar operações com pontos em K dimensões. É uma estrutura de árvore binária onde cada nó representa um conjunto de K valores. É muito utilizada em aplicações que envolvem operações de busca multi-dimensionais. Mais informações sobre o método pode ser visto em ([BENTLEY, 1975](#)).

Após a aplicação a priori da redução de amostragem *VoxelGrid* (seção 3.2.2), é conhecido que os pontos estarão espaçados a uma distância média de 3 centímetros. Com isso, para o método de segmentação por distância Euclidiana foi definida a distância mínima entre os pontos de um mesmo grupo em 10 centímetros, obtendo o resultado demonstrado na Figura 3.10.

O método Euclidiano demonstra um ótimo resultado no que se refere a tarefa de segmentar diferentes objetos. Na Figura 3.10 é visto objetos representados por cores diferentes, demonstrando que foi possível evidenciar cada um dos objetos na cena. Porém, foram encontrados alguns problemas, e portanto algumas medidas foram tomadas na etapa de segmentação com o intuito de obter melhores resultados nas próximas etapas.

Figura 3.10 – Aplicação da segmentação na nuvem de pontos



Fonte: Imagem de autoria do autor.

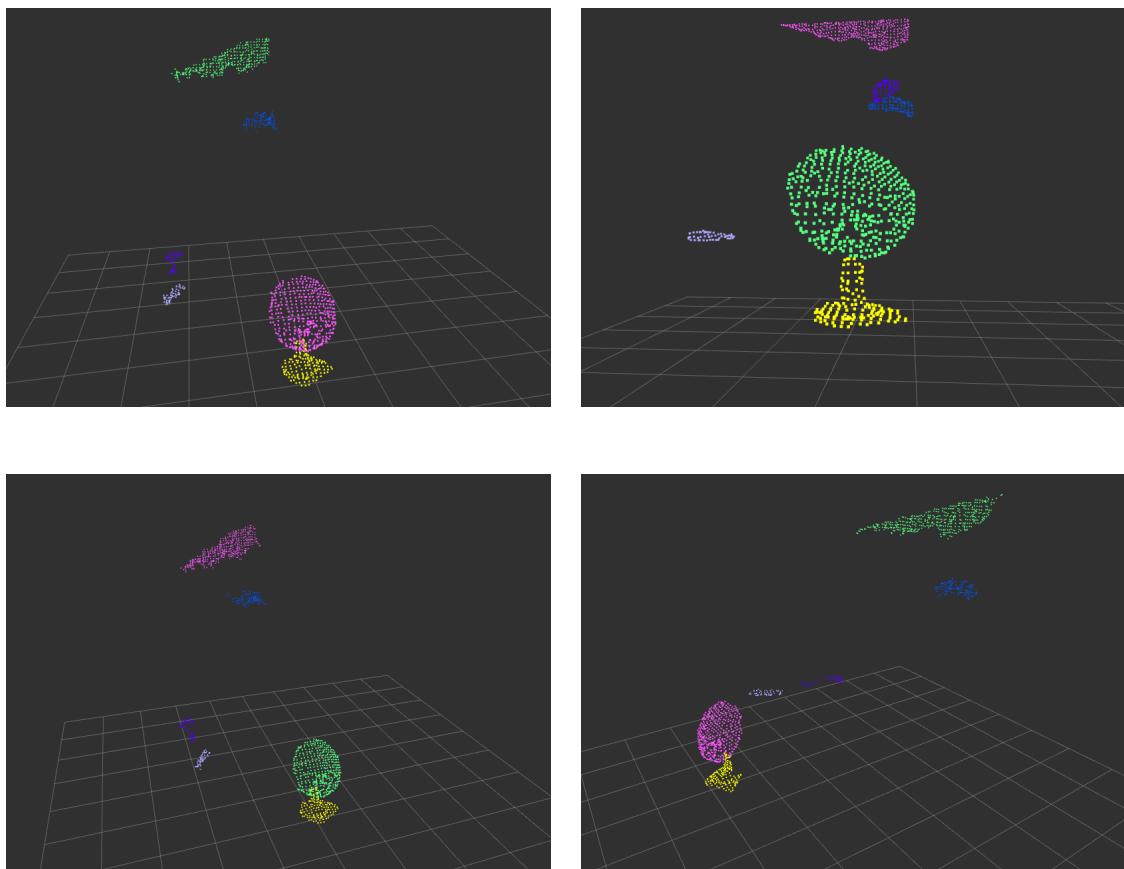
Um dos problemas é o da oclusão de objetos. Isso não é algo decorrido do método Euclidiano, e sim do modo de funcionamento do sensor. Porém, é algo que faz com que o método resulte em um falso positivo, devido ao fato do sensor coletar somente a superfície visível a ele, onde objetos atrás de outros objetos em primeiro plano são representados separadamente, fazendo com que sejam segmentados de maneira errônea, conforme mostrado na Figura 3.10. O objeto roxo (ventilador) obstrui a visão para o objeto após ele, gerando assim dois segmentos, um verde e um rosa, porém ambos pertencem ao mesmo objeto.

Outro problema é o de objetos demasiadamente próximos uns dos outros. Na Figura 3.10 é visto que a cama e a parede são classificadas como o mesmo segmento, isso se da pelo fato de que alguns pontos estão próximos dos dois objetos, interpretando ambos como um só segmento. Nesta etapa é visto a importância da aplicação do *RANSAC* (seção 3.2.1), pois sem sua aplicação todo o conjunto estaria conectado ao plano do chão, tornando tudo em um só grande segmento.

Visto que o objetivo do trabalho está em um método de detecção e rastreamento de objetos no campo de visão de um robô terrestre, foi definido que os objetos utilizados

para testes possuem um tamanho limitado, e sendo assim, foi criado um limite máximo de pontos por segmento. Ou seja, segmentos demasiadamente grandes como paredes, portas, etc, no recinto seriam desconsiderados, já que a quantidade de pontos do conjunto que os representam excedem o limite. O resultado é um conjunto menor de dados a serem processados posteriormente, evidenciando objetos de interesse na cena, conforme visto na Figura 3.11. Porém, a etapa de segmentação não é capaz de resolver o problema da oclusão, e desta forma, ficarão para trabalhos futuros possíveis soluções para esse problema.

Figura 3.11 – Aplicação da segmentação com limitador na nuvem de pontos



Fonte: Imagem de autoria do autor.

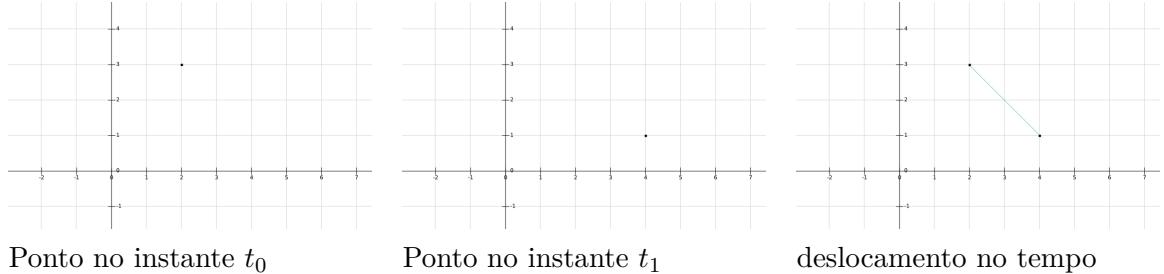
Conforme também demonstrado na Figura 3.11, segmentos não possuem uma identificação, por isso são representados por cores diferentes em cada instante de tempo. Para a aplicação no sistema proposto, era necessário identificar um segmento através do tempo, de modo que se fez necessária a etapa de associação temporal.

3.4 Associação

A etapa de associação é responsável por identificar um segmento de nuvem sn do instante de tempo t_i e associar a outro segmento no instante t_{i+1} , de modo que seja

conhecido que um conjunto de pontos $p \in sn$ em instantes de tempo diferentes represente o mesmo objeto no mundo real, como demostrado na Figura 3.12.

Figura 3.12 – Representação temporal de um ponto



Fonte: Imagem de autoria do autor.

Definido então a associação, é possível extrair outras informações do conjunto de dados, como por exemplo, deslocamento, velocidade, direção de deslocamento do objeto observado, etc. E tendo posse da frequência de leitura do sensor, é possível converter para uma medida de tempo real.

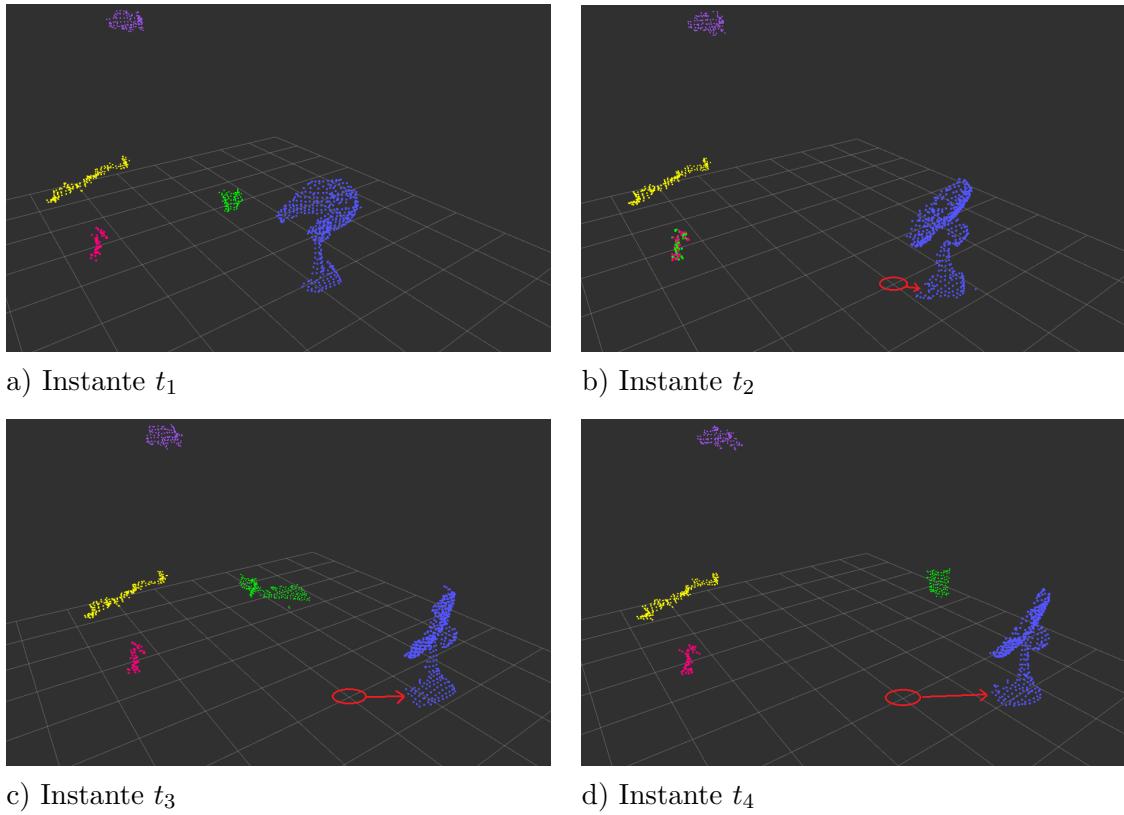
3.4.1 Associação por Vizinhos Próximos

A associação por vizinhos próximos possui aplicação em inúmeros trabalhos da área, como (BECKER et al., 2009), (VU; AYCARD; APPENRODT, 2007), (AZIM; AYCARD, 2012), e demonstra resultados satisfatórios na execução da tarefa.

Neste trabalho foi utilizada uma abordagem pelo centro de massa para realizar a comparação de vizinhos próximos. Para cada objeto da cena $sn_k \subset N$ (seção 3.3), é obtido o centro de massa $cm_k = \{x_k, y_k, z_k\}$ do mesmo. A partir disso, é então definido um conjunto de nuvens associadas $na = \emptyset$, a cada leitura do sensor, uma verificação é realizada, e o centro de massa cm_k de cada segmento $sn_k \subset N$ é comparado com os centros de massa dos objetos $o_j \in na$, procurando-se o vizinho mais próximo. A proximidade de vizinhança é definida como a distância Euclidiana dos centros de massa cm_k do segmento sn_k e do objeto o_j . Caso a distância entre ambos seja menor que um círculo com raio $R < limiar$, onde o $limiar$ é um valor mínimo de distância predefinido no algoritmo, o elemento sn_k substitui o objeto $o_j \in na$, atualizando o conjunto de dados pertencentes ao mesmo, como por exemplo, nuvem de pontos e centro de massa. Caso nenhum elemento do conjunto na seja associado ao segmento sn_k , o mesmo é adicionado ao conjunto na como novo elemento.

Na Figura 3.13 é visto como o objeto em azul é associado a sua representação do tempo anterior, mesmo após ser deslocado para outra posição, nesse caso a associação é representada pelo fato de se ter usado a mesma cor para representação do objeto em

Figura 3.13 – Exemplo de associação no tempo de uma sequencia de nuvens



Fonte: Imagem de autoria do autor.

todos os instantes. O deslocamento pode ser observado por um indicador (em vermelho) nas imagens, e objetos estáticos também são associados.

Porém, um dos problemas dessa abordagem está em quando ocorre oclusão em relação ao sensor, pois o algoritmo perde muitas referências, como por exemplo o centro de massa do objeto, visto que quando parte da representação do objeto é ocluído por outro, a noção de tamanho é prejudicada, tornando a associação do mesmo imprecisa. Outro agravante é o de que se o objeto é totalmente oculto, ao reaparecer poderá não ser associado a sua representação do instante anterior, e portanto sendo categorizado como um novo objeto, como ocorre com um dos segmentos na Figura 3.13b.

3.5 Classificação

A classificação se trata da identificação de um objeto em uma distribuição por classes predefinidas. Basicamente, o processo de classificação está em identificar a qual grupo de características o objeto observável mais se assemelha. Neste trabalho, todos os modelos de treino correspondem a um único objeto, sendo assim, cada objeto por si só constitui uma classe, portanto a classificação neste trabalho é definida como a etapa de

identificação do objeto observado.

3.5.1 VFH

O modelo de classificação VFH (*Viewpoint Feature Histogram*) ([RUSU et al., 2010](#)) ou Histograma de característica de ponto de vista, foi criado em 2010, e se trata de um descriptor para nuvens de pontos tridimensional, capaz de classificar informações de geometria e de ponto de vista de uma nuvem de pontos. Foi desenvolvido para uma aplicação robótica de manipulação de objetos por um braço robótico, e portanto, possui um excelente funcionamento em tempo real.

Para um funcionamento relativamente bom, assume-se que certas etapas de pré-processamento sejam feitas, como a identificação e remoção de planos, a segmentação dos pontos restantes em um espaço Euclidiano, e o cálculo de normais de cada ponto. Como grande parte dessas etapas já foram realizadas, somente a estimativa de normais é necessária para realizar a classificação.

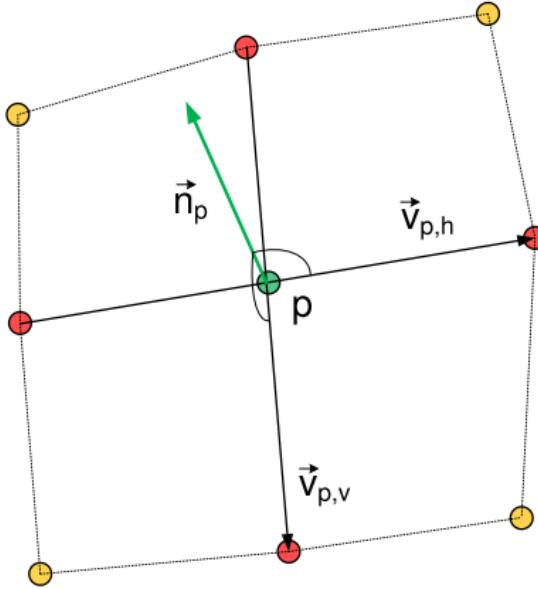
Existe um problema para realizar a estimativa de normais de cada ponto em uma nuvem, não é possível estimar matematicamente o plano tangente em um ponto qualquer de uma superfície tridimensional, sem o conhecimento da equação que a compõe, pois o conjunto de informações é muito vago com somente um ponto. Para uma das possíveis resoluções desse problema, em ([HOLZER et al., 2012](#)) é definido um método para estimativa de normais em nuvem de pontos. O método foi desenvolvido visando aplicações de tempo real, e, resumidamente, seu funcionamento se da através da seguinte maneira:

1. Para cada ponto p_i e seus p_i^k vizinhos é inferido uma superfície.
2. Dada a superfície, é calculado o vetor normal \vec{n} da mesma.
3. É então verificado se a orientação do vetor normal \vec{n} do ponto p_i consiste com a orientação do ponto de visão Pv do observador, caso não esteja o sentido do vetor é invertido.

Por se tratar de uma estimativa, não pode-se assumir o vetor estimado como verdade absoluta, porém o método atinge resultados satisfatórios, segundo os resultados demostrados em ([HOLZER et al., 2012](#)). Uma exemplificação do processo é demonstrada na Figura 3.14.

Após a etapa de estimativa de normais, cada segmento sn_k de nuvem possuirá a informação do vetor normal \vec{n} pra cada ponto $p_j \in sn_k$ (Figura 3.15). Assim como sugerido no trabalho de ([RUSU et al., 2010](#)), foi utilizado o método de *VoxelGrid* para redução do conjunto, de modo a melhorar a performance do algoritmo. E definido o raio de busca da

Figura 3.14 – Estimação de normal por vizinhos próximos.



Fonte: Imagem retirada de ([HOLZER et al., 2012](#)).

estimativa de normal um pouco maior que o tamanho do *voxel*, fazendo com que todos os pontos sejam capazes de estimar uma normal.

A ideia do VFH é semelhante com a dos trabalhos anteriores ([RUSU; BLODOW; BEETZ, 2009](#)) e ([RUSU, 2010](#)), porém adicionando um componente de ponto de vista ao histograma padrão (Figura 3.17). O VFH utiliza a metologia do FPFH com um histograma que coleta um conjunto de informações angulares entre pares de normais na nuvem. Especificamente, para cada par de pontos $\langle p_i, p_j \rangle$ e suas respectivas normais $\langle \vec{n}_i, \vec{n}_j \rangle$, o seu conjunto de divergências angulares pode ser definido como:

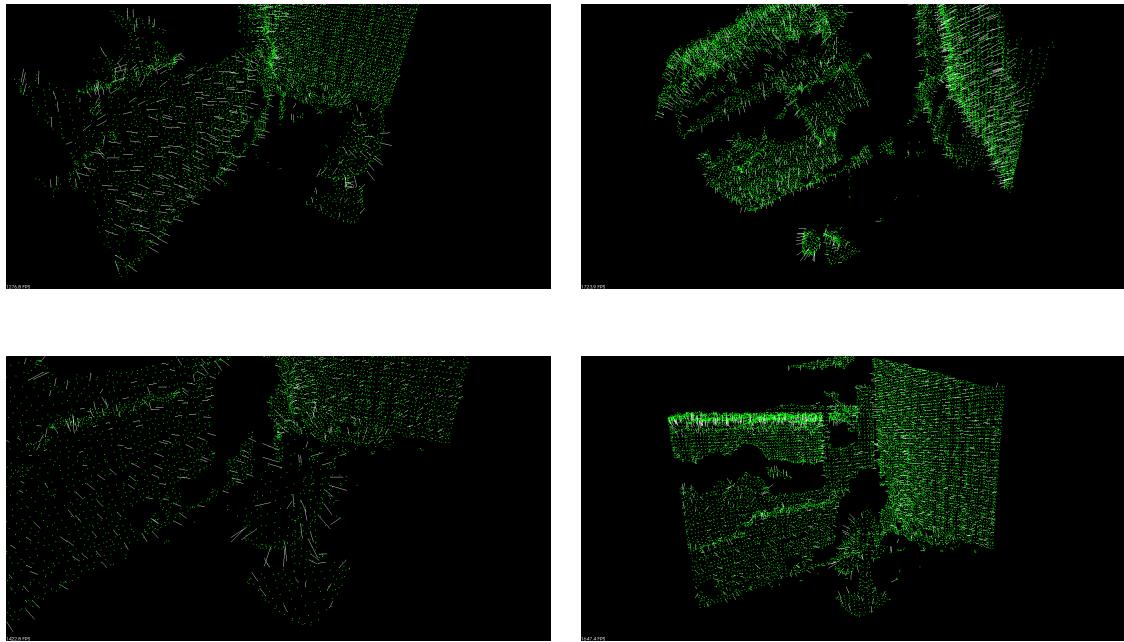
$$\alpha = v \cdot \vec{n}_j$$

$$\phi = \frac{(p_j - p_i)}{d}$$

$$\theta = \arctan(w \cdot \vec{n}_j, u \cdot \vec{n}_j)$$

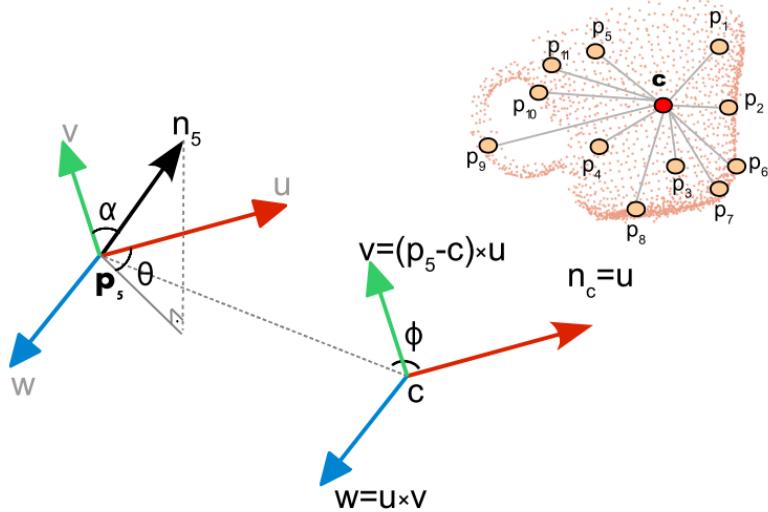
onde u, v, w representam um sistema de coordenadas *Darboux frame* no ponto p_i (Figura 3.16), esse sistema de coordenadas possui três vetores ortonormais centrados no ponto. Uma melhor definição de como funciona o *Darboux frame* e como é realizado a extração dessas informações de uma nuvem de pontos podem ser encontradas em ([HAMEIRI; SHIMSHONI, 2003](#)).

Figura 3.15 – Vetor normal estimado para cada ponto da nuvem



Fonte: Imagem de autoria do autor.

Figura 3.16 – Extração de características entre pontos.

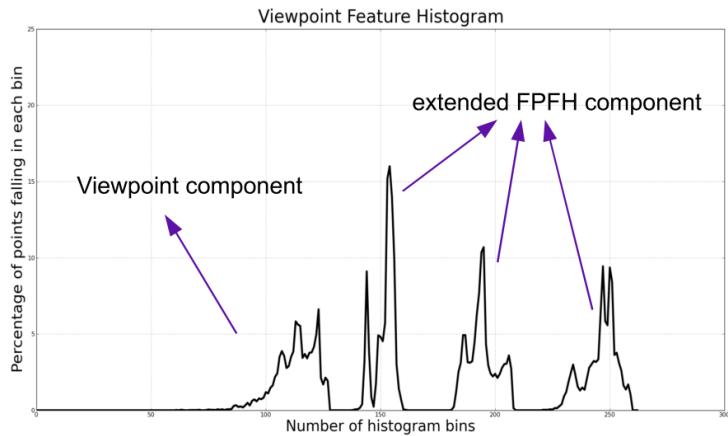


Fonte: Imagem retirada de ([RUSU et al., 2010](#)).

Devido ao fato da complexidade do PFH ser $O(n^2)$, é utilizada uma abordagem FPFH para criação do histograma. Com isso, pra cada par $\langle p_i, p_i^k \rangle$ é computado todos os conjuntos de $\langle \alpha, \phi, \theta \rangle$, onde k é o numero de vizinhos, e então é discretizado o conjunto de $\langle \alpha, \phi, \theta \rangle$ para um histograma, conforme Figura 3.17. Cada histograma possui um

eixo x discreto com um numero finito pré-definido de divisões, onde essas divisões são denominadas *bins*. Neste trabalho foi utilizado um histograma com assinatura de 308 *bins*.

Figura 3.17 – Histograma VFH (*Viewpoint Feature Histogram*).



Fonte: Imagem retirada de ([RUSU et al., 2010](#)).

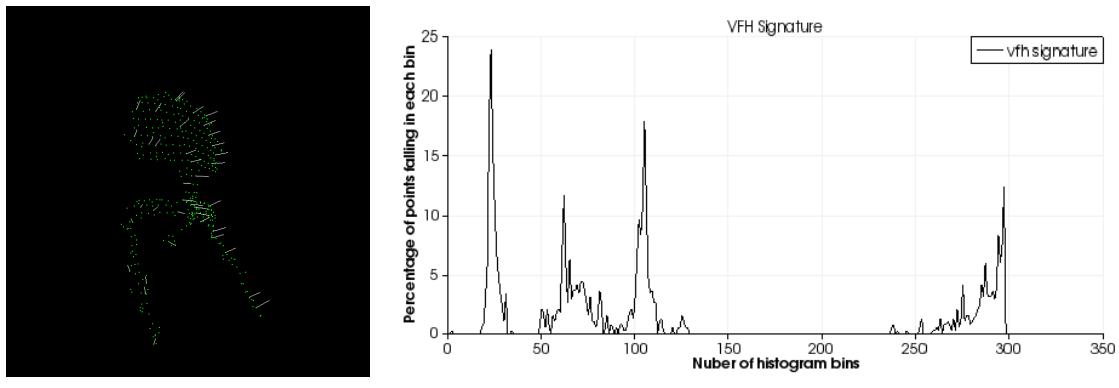
3.5.2 Aplicação do VFH

Uma das partes importantes para realizar a classificação é a etapa de criação de modelos. Neste trabalho, separamos cada tipo de objeto em uma classe, e adicionamos inúmeras amostras e seus respectivos histogramas em diferentes ângulos de visão, conforme visto na Figura 3.18. Essa será a base de conhecimento do classificador, e após isso, cada elemento no campo de visão do sensor será comparado ao conjunto de histogramas de cada objeto na base de conhecimento através de uma classificação por KNN, onde $k = 5$. Para o cálculo de distância de histogramas foi utilizado a distância quadrada de Chi ([PELE; WERMAN, 2010](#)), definida pela seguinte fórmula:

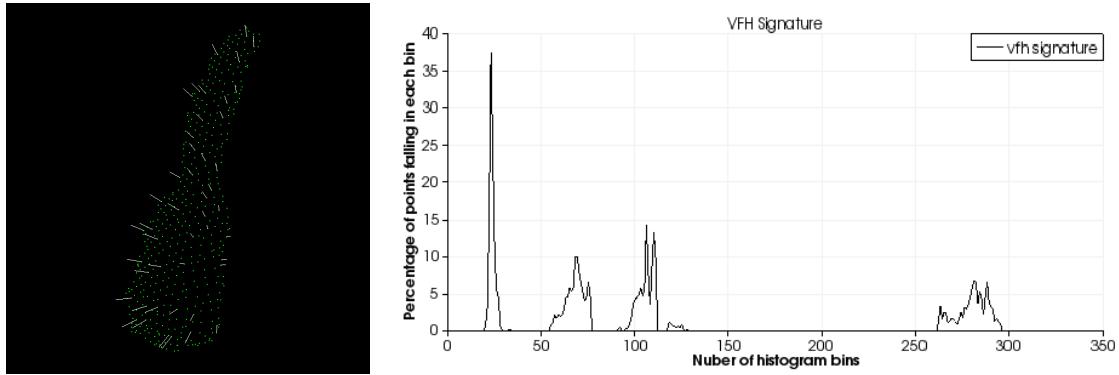
$$\frac{1}{2} \sum_{i=1}^n \frac{(x_i - y_i)^2}{(x_i + y_i)}$$

Onde n é o numero de *bins* (intervalos do eixo x) do histograma e x_i e y_i os valores dos histogramas no índice i . Essa fórmula é utilizada para medir a distância quadrática entre dois histogramas. Ou seja, para realizar uma análise de correspondência entre ambos, obtendo-se então um índice de similaridade, onde quanto menor o valor, mais semelhantes são os histogramas comparados. Sua função possui complexidade de $O(n)$ e seu gráfico de crescimento se assemelha a qualquer função quadrática do tipo $ax^2 + bx + c$ onde $0 \leq x \leq \infty$, sendo x correspondente a distancia entre os histogramas.

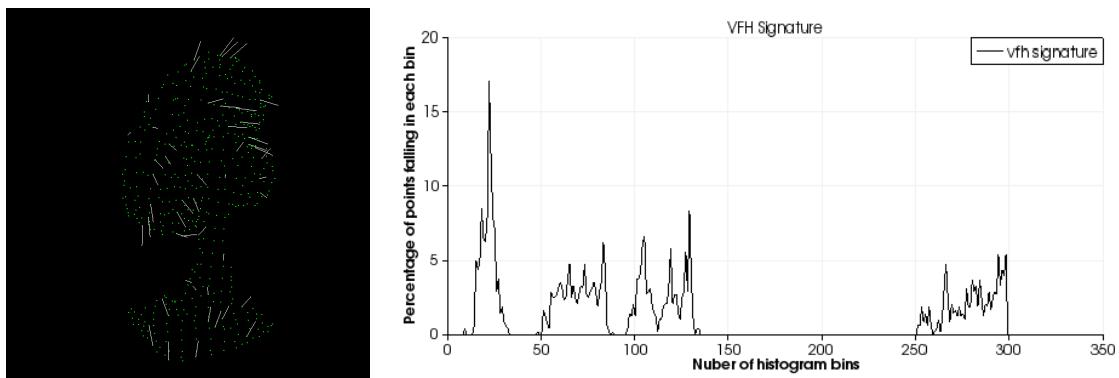
Figura 3.18 – Exemplos de histogramas VFH de modelos aleatórios



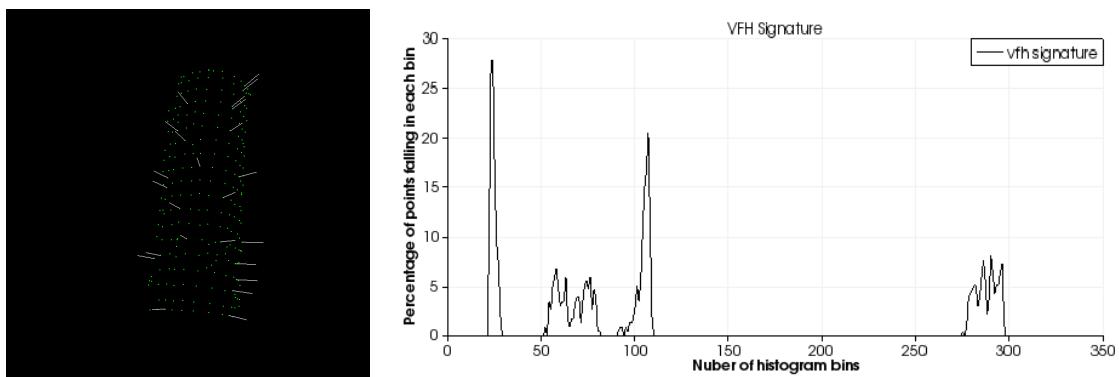
Modelo 1 - Cadeira



Modelo 2 - Case de Guitarra



Modelo 3 - Ventilador



Modelo 4 - Cilindro

Fonte: Imagem de autoria do autor.

4 Experimentos, Resultados e Discussão

Neste capítulo são apresentados os resultados dos métodos utilizados no sistema desenvolvido no trabalho, avaliando-se métricas qualitativas e quantitativas de eficiência. Os testes realizados nesse trabalho foram feitos em ambientes controlados, com objetivos bem definidos e em locais *indoor* (fechados, sem interferência da luz do sol, devido as restrições do sensor), afim de se medir o funcionamento do sistema.

4.1 Materiais e Métodos

Nesta seção são mostrados os materiais e métodos utilizados para a realização do trabalho, a infraestrutura montada e o projeto de desenvolvimento utilizado no sistema.

4.1.1 Infraestrutura

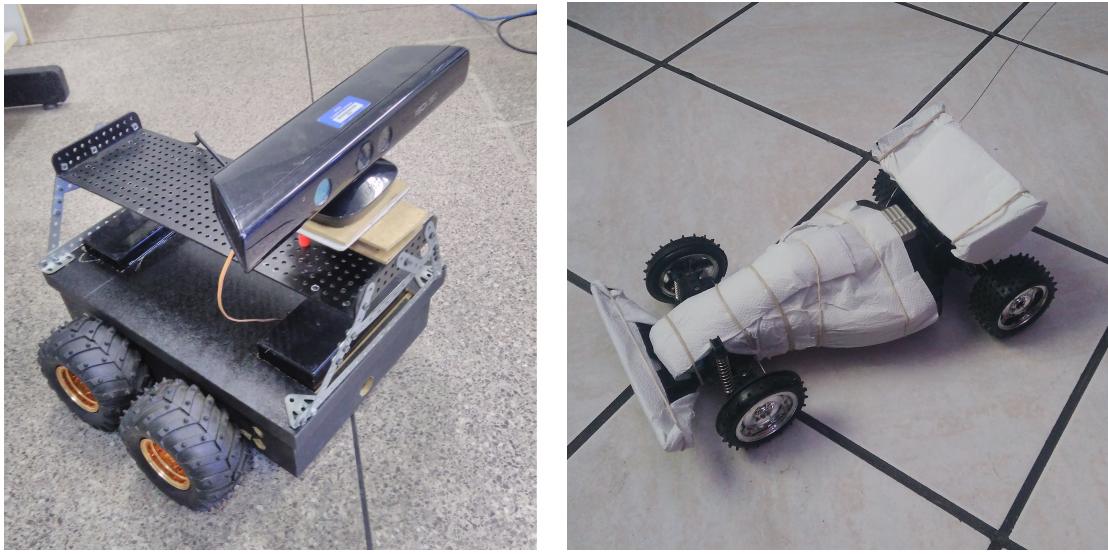
No desenvolvimento deste trabalho e para a execução dos testes foi utilizado um notebook com processador Intel® Core™ i7-3517U CPU @ 1.90GHz × 4, memória 8Gb DDR3 1600Hz em um sistema Ubuntu 15.04 64-bit. O computador fica embarcado em uma Plataforma Robótica experimental (PR). A PR foi utilizada como um robô de patrulhamento capaz de realizar a perseguição à um alvo (robô perseguidor). Como alvo utilizou-se um carro rádio controlado (objeto alvo). O robô e o objeto utilizado como alvo podem ser vistos na Figura 4.1.

A PR usada neste trabalho foi desenvolvida por alunos do LARSE (Laboratório de Robótica e Sistemas Embarcados). A PR está sendo utilizada como plataforma de teste para vários algoritmos relacionados à robótica. Ela é apoiada em uma base de madeira e metal, contendo quatro rodas motorizadas com motores de corrente contínua, duas baterias de 12V e 7A, um Kinect, um notebook, duas pontes h, uma placa microcontrolada e uma *protoboard*. Além disso, a PR suporta outros sensores como: sonar, infravermelho, GPS e unidade de medida inercial, mas que não serão tratados neste trabalho.

4.1.2 ROS e PCL

O ROS (*Robot Operational System*) é um sistema de troca de mensagens baseado em uma arquitetura de rede de *publisher/subscriber* com foco em aplicações robóticas. O conceito básico de seu funcionamento é o de nós independentes que executam tarefas específicas. A troca de dados é gerenciada pelo ROS através de uma especie de servidor central denominado *roscore*. Cada nó ao ser iniciado, diz quais tópicos quer assinar e quais quer publicar, onde tópicos são canais de transmissão de dados. Após isso, os nós

Figura 4.1 – Infraestrutura utilizada no sistema.



a) Robô perseguidor

b) Objeto alvo de teste

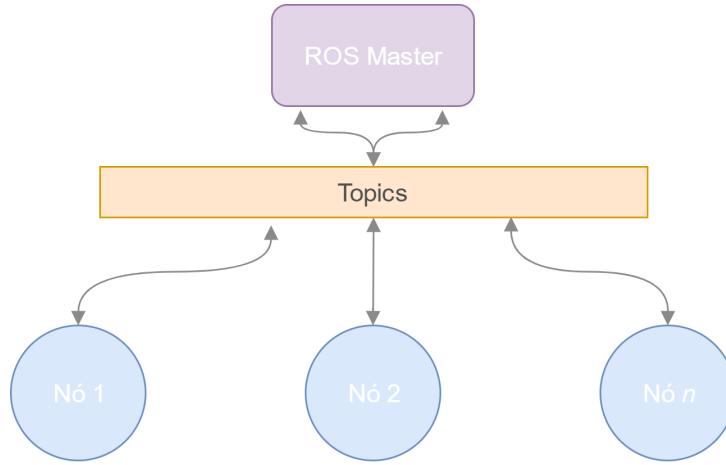
Fonte: Imagem de autoria do autor.

se comunicam por esses canais de modo a realizar o objetivo planejado. Os nós podem ser escritos em C++ ou em *Python*. Uma melhor visualização do fluxo pode ser visto na Figura 4.2.

A PCL ([RUSU; COUSINS, 2011](#)), por outro lado, é uma biblioteca de código aberto, e já se encontra nas dependências do ROS por padrão, e foi utilizada para realização das funções de análise e tratamento dos dados das nuvens de pontos obtidas pelo sensor (Kinect). É desenvolvida na linguagem C e C++, e fornece um conjunto de funções e modelos que visam a simplificação do uso de métodos já consolidados, como por exemplo filtros, estimativa de características, reconstrução de superfícies, registro e segmentação de nuvens de pontos, etc. É suportada por uma comunidade de desenvolvedores tanto da área de pesquisa em robótica quanto na área de percepção artificial.

Nesse trabalho cada uma das funções foi construída separadamente como um nó do sistema, possibilitando uma melhor avaliação individual de cada função. Isso também possibilita a troca de partes do sistema (nós) sem alteração de outros nós. Isso é viável, por exemplo, para troca de abordagens baseado no contexto. Uma visão completa do fluxo de funcionamento adotado nesse trabalho pode ser visto na Figura 4.3.

Figura 4.2 – Fluxo de funcionamento do ROS.



Fonte: Imagem de autoria do autor.

Figura 4.3 – Fluxo de funcionamento dos nós do ROS implementados neste trabalho.



Fonte: Imagem de autoria do autor.

4.2 Experimentos Associados ao Pré-Processamento

Nesta seção, são apresentados os resultados da utilização das funções de pré-processamento, o impacto de sua parametrização e sua influência nas etapas posteriores. O pré-processamento conta com duas etapas: a remoção do plano do chão através do método RANSAC e a redução da nuvem de pontos pelo método *VoxelGrid*. O resultado dessa etapa é um conjunto reduzido e segmentável de pontos em um espaço tridimensional.

4.2.1 Remoção do Plano

Para a remoção do plano foi utilizado o método RANSAC. O mesmo conta com dois parâmetros pré-determinados pela aplicação, que são o número de iterações, e limiar

de distância do modelo. A escolha de qual medida utilizar para cada parâmetro foi definida após a realização de uma avaliação qualitativa para cada par de valores, com base no desempenho avaliado por meio do tempo de execução e do nível de ruído verificado visualmente.

Tabela 4.1 – Desempenho de cada par de parâmetros do RANSAC

Parâmetros		Desempenho Médio
Limiar = 2cm	iterações = 50	5,99 FPS
	iterações = 150	4,48 FPS
	iterações = 500	4,51 FPS
	iterações = 1000	4,42 FPS
Limiar = 4cm	iterações = 50	5,99 FPS
	iterações = 150	5,40 FPS
	iterações = 500	5,22 FPS
	iterações = 1000	5,32 FPS
Limiar = 6cm	iterações = 50	7,21 FPS
	iterações = 150	7,27 FPS
	iterações = 500	7,01 FPS
	iterações = 1000	7,12 FPS

O resultado do desempenho dos testes pode ser observado na Tabela 4.1, onde é visualizada a quantidade média de *frames* por segundo (FPS) para cada par de parâmetros. Nas Figuras 4.4, 4.5 e 4.6 é visto imagens do resultado do RANSAC para cada opção de parâmetro. No caso do RANSAC, o número de iterações não impacta tão diretamente no desempenho do método, mas sim a distância limiar. Isso se da pelo fato de que mais pontos são processados por iteração, para que o método possa convergir mais rapidamente. Todavia, isso não garante maior qualidade do método.

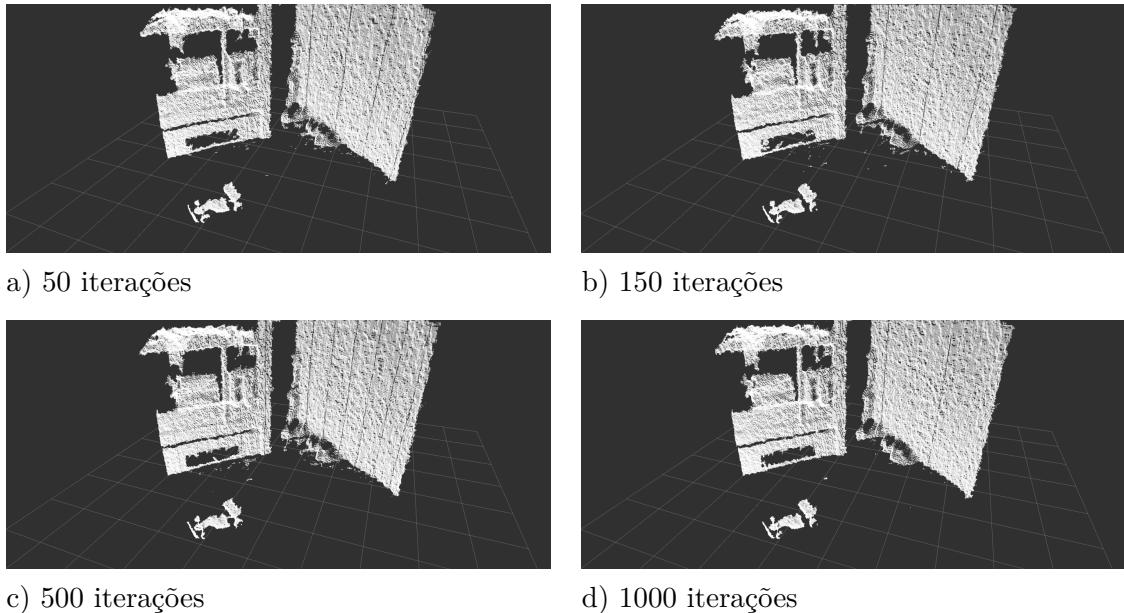
Como é visto nas Figuras 4.4, 4.5 e 4.6, maiores limiares resultaram em um número maior de ruído, e algumas vezes causando erros como na Figura 4.5d. Na Figura 4.6 é visto que, devido ao limiar estar demaisiadamente grande, a parte superior dos objetos ao fundo do ambiente é classificado como parte do chão pelo método.

Desta forma, para esse trabalho foi escolhido utilizar um limiar de 2cm com um limite de 50 iterações, já que após uma avaliação de desempenho verificou-se uma boa média de performance, e após uma avaliação visual observou-se um baixo nível de ruído.

4.2.2 Redução de Amostragem

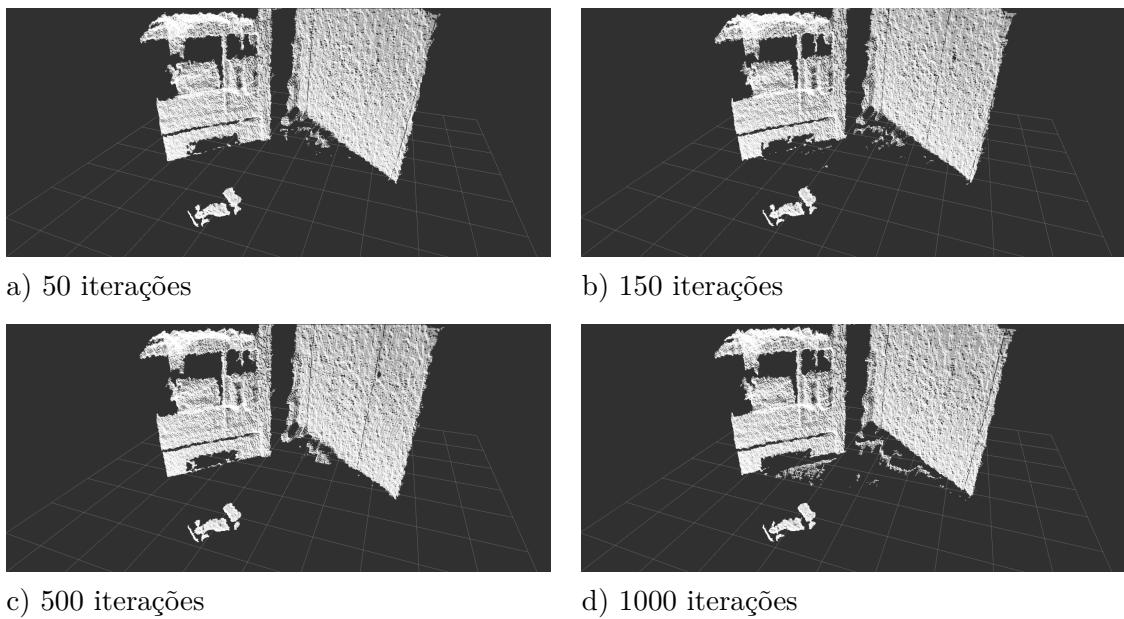
Nos testes executados, o método de *VoxelGrid* demonstrou um excelente resultado, sendo capaz de diminuir o conjunto de dados em taxas extremamente satisfatórias, e mantendo grande parte das características da nuvem, e com isso, sendo possível a realização das próximas etapas com bons resultados. Para demonstrar a capacidade de redução do conjunto na utilização do *VoxelGrid* foram captadas as médias de pontos das nuvens

Figura 4.4 – Nuvem após RANSAC com distância limiar de 2cm



Fonte: Imagem de autoria do autor.

Figura 4.5 – Nuvem após RANSAC com distância limiar de 4cm

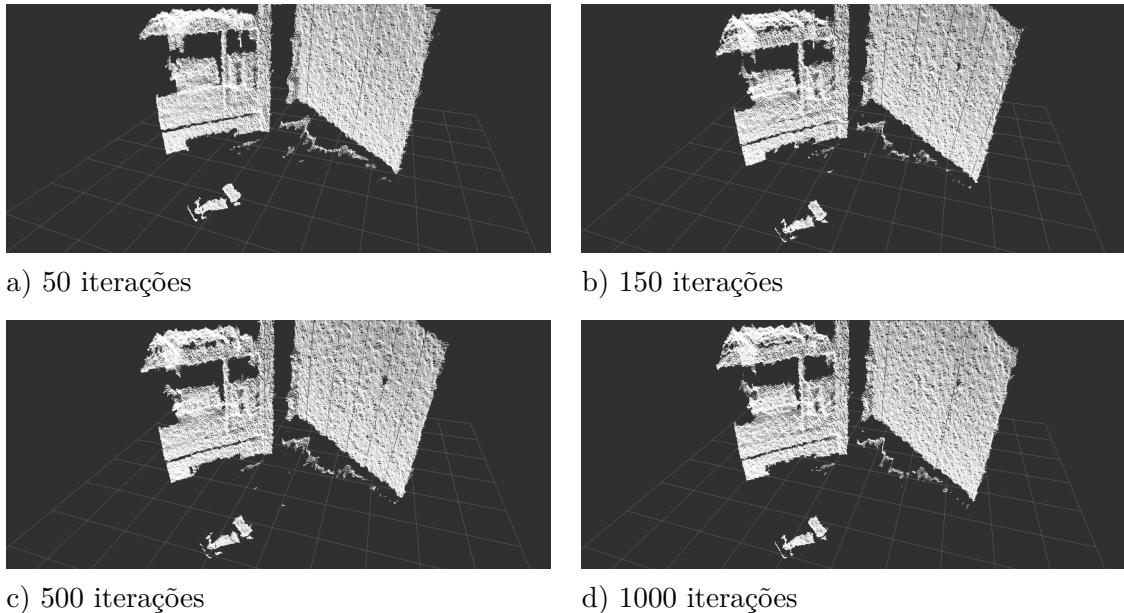


Fonte: Imagem de autoria do autor.

resultantes do processamento por uma tempo de 60 segundos, em três diferentes tamanhos de *voxel*, conforme visto na Tabela 4.2.

A etapa de redução do conjunto de dados é de extrema importância para a execução do sistema em tempo real. O Kinect conta com uma câmera de 30Hz de frequência e uma resolução 640 x 480 resultando em 9.216.000 de pontos por segundo. Como grande parte

Figura 4.6 – Nuvem após RANSAC com distância limiar de 6cm



Fonte: Imagem de autoria do autor.

Tabela 4.2 – Desempenho de redução do *VoxelGrid*

Tamanho do <i>Voxel</i>	Média de pontos por <i>frame</i>	Porcentagem de redução
$1cm^3$	91907	70,08 %
$3cm^3$	16650	94,57 %
$5cm^3$	6366	97,92 %

dos métodos possuem alta complexidade computacional baseado no numero de pontos, um alto conjunto de dados resulta em alto tempo de processamento, tornando a aplicação inviável para execução em tempo real. A Tabela 4.3 demostra como o tamanho do conjunto de dados impacta no desempenho do sistema através da análise do tempo de execução da etapa de agrupamento por distância Euclidiana baseado no tamanho do *voxel*.

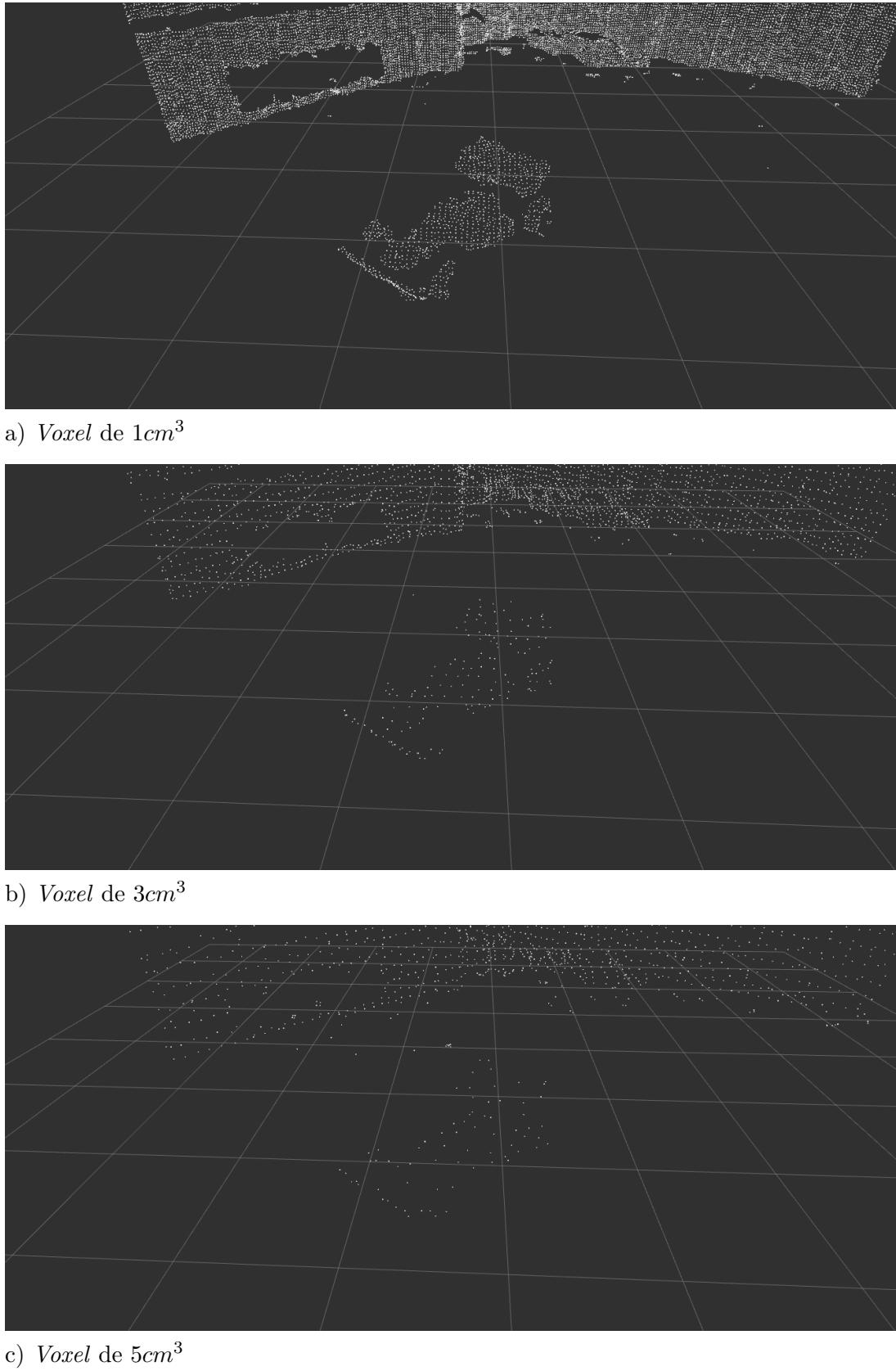
Tabela 4.3 – Impacto do VoxelGrid na segmentação.

Tamanho do <i>Voxel</i>	Média de pontos por <i>frame</i>	Desempenho médio do Agrupamento
$1cm^3$	72480	0,33 frames / s
$3cm^3$	13776	7,51 frames / s
$5cm^3$	5259	44,19 frames / s

Para este trabalho foi adotado o tamanho de $3cm^3$ pois reduz satisfatoriamente o conjunto de dados, mantendo uma alta gama de características do conjunto origem, e possui um desempenho satisfatório que se assemelha ao desempenho do RANSAC. Outro fator é que a inferência do vetor normal de um ponto usa como base os vizinhos próximos,

e portanto com um *voxel* de $3cm^3$ pode ser obtido um número suficiente de vizinhos para a execução de uma estimativa aceitável, sem prejudicar o desempenho. Uma melhor observação da diferença visual entre os tamanhos de *voxel* pode ser vista na Figura 4.7.

Figura 4.7 – Diferença entre nuvens com tamanhos de *voxel* diferentes.



Fonte: Imagem de autoria do autor.

4.3 Experimentos Associados à Classificação

Nesta seção, são apresentados os resultados dos experimentos realizados com os métodos de descrição e classificação aplicados no trabalho. Para a classificação foi utilizada uma abordagem do vizinho mais próximo (KNN). Primeiramente, foi definido o limiar utilizado na execução do método, e após isso foi realizada uma avaliação quantitativa e qualitativa do módulo de classificação. Essa avaliação foi realizada de modo visual por um período pré-determinado de tempo.

As medidas avaliativas de desempenho para o método de classificação utilizado neste trabalho foram divididas em medidas qualitativas e medidas quantitativas. As medidas qualitativas foram realizadas por meio de análise visual das amostras retiradas durante o teste do classificador. Essas amostras podem ser vistas nas Figuras 4.10, 4.12 e 4.11. As medidas quantitativas foram realizadas através do uso de dois índices de avaliação: a precisão e a revocação.

4.3.1 Escolha do Limiar

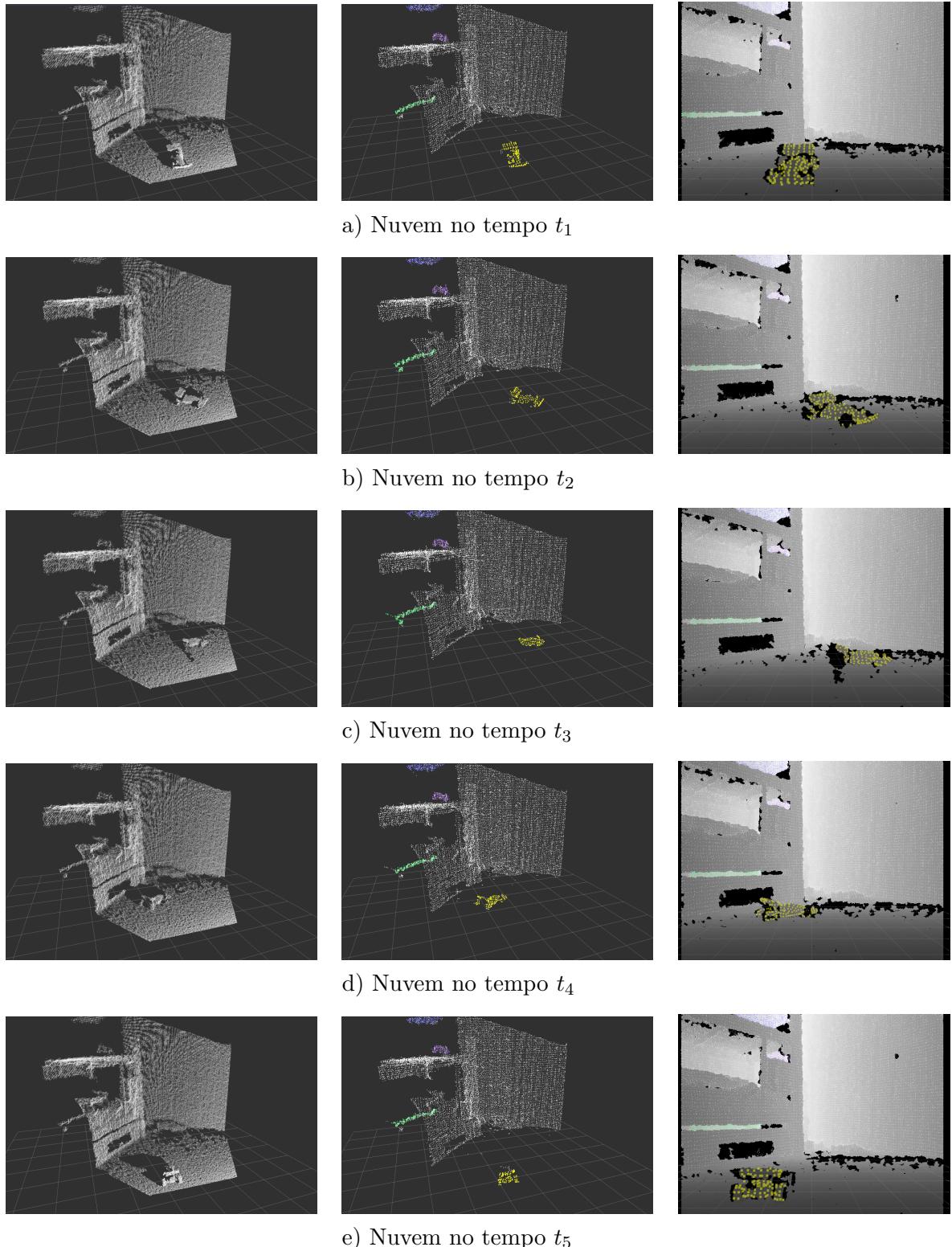
Para realizar a escolha do limiar de semelhança do classificador foi utilizada uma avaliação estatística. Por um período de aproximadamente dois minutos, foram armazenados todos os índices de semelhança do histograma VFH da nuvem que representa um objeto em observação, com os modelos do objeto armazenados. O índice de semelhança corresponde a distância entre ambos os histograma (objeto alvo e modelo) e é obtido através da Distância Quadrática de Chi, explicada na seção 3.5.2. Durante o período de observação, o objeto foi movimentado de modo que fosse capturado grande parte dos ângulos possíveis de visualização. Partes desse processo pode ser visto na Figura 4.8.

A Figura 4.8 apresenta instantes diferentes dos testes, sobre três ângulos de visão diferentes: o primeiro ângulo demonstra a nuvem completa; o segundo a nuvem filtrada, segmentada e associada no tempo; e o último o ponto de visão do sensor, próximo ao chão. O objeto de teste aparece em amarelo nas imagens.

O objeto observado conta com um conjunto de 120 modelos para a realização da comparação, e, ao final da etapa de captura dos dados, foram obtidos um total de 940 avaliações. Os dados analisados foram dispostos em dois gráficos conforme visto na Figura 4.9. O Gráfico 4.9a apresenta um histograma dos índices de semelhança (*score*) entre o objeto analisado e modelo de maior semelhança, obtidos durante os testes, onde o eixo *y* representa o valor de semelhança e o eixo *x* cada uma das leituras. O gráfico 4.9b representa a distribuição de frequências dos *scores*, onde o eixo *y* representa a quantidade de ocorrências e o eixo *x* representa o intervalo.

Existem dois fatos interessantes a serem observados. O primeiro é que, sua distribuição de ocorrência se assemelha grandemente a uma distribuição normal, mostrando nessa

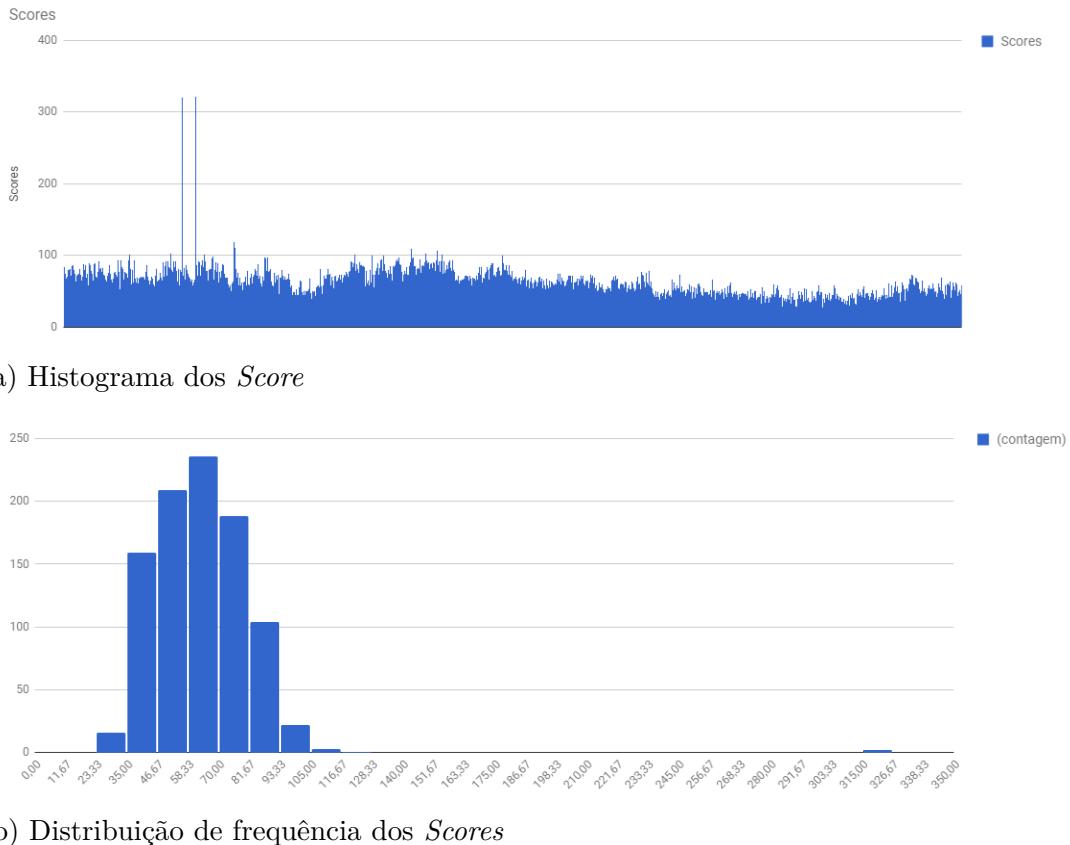
Figura 4.8 – Captura de índices de semelhança de um objeto



Fonte: Imagem de autoria do autor.

primeira etapa que o conjunto de modelos está aceitável para o método de classificação, pois possui boa taxa média de semelhança, não possuindo muitas leituras fora do esperado.

Figura 4.9 – Captura de índices de semelhança de um objeto



Fonte: Imagem de autoria do autor.

Outro fato é que ocorreram somente duas leituras errôneas do sensor, em um conjunto de 940 leituras, demonstrando assim uma taxa de confiabilidade de leitura de aproximadamente 99,78% para esse teste.

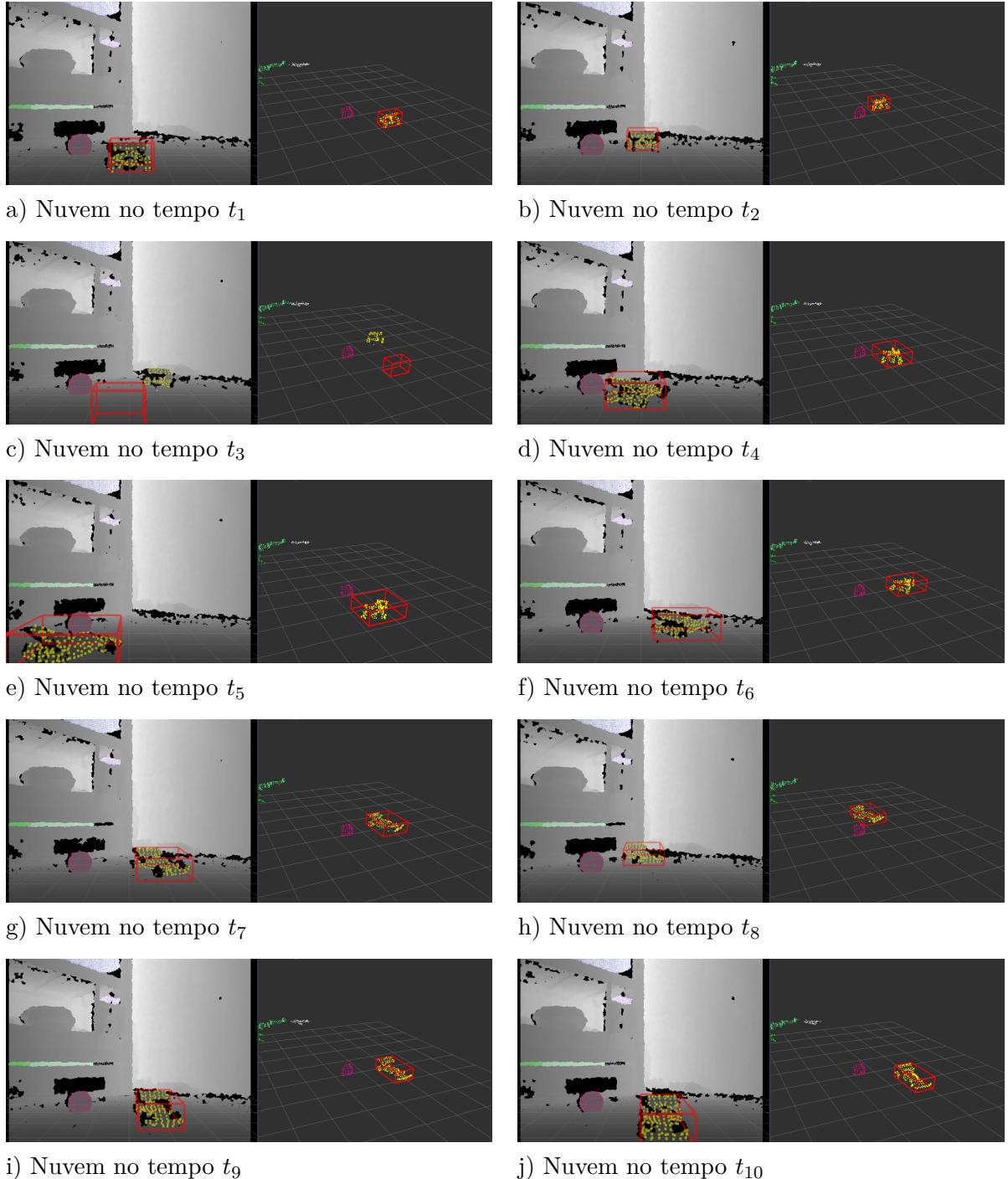
Após a análise dos dados, utilizando uma abordagem de função teto (o teto de um número real x é o valor inteiro maior ou igual a x), o intervalo de maior ocorrência foi o [59, 70], optou-se então por utilizar o 70 como limiar para a classificação. Como aproximadamente 65,95% do conjunto de amostras do objeto alvo atende essa condição, definiu-se como uma taxa satisfatória de acerto, garantindo um menor número de falsos positivos. Nesse caso, um falso positivo é uma associação errada do modelo com um objeto de natureza diferente.

4.3.2 Avaliação Qualitativa

Para avaliação qualitativa do método, foi realizada uma análise visual de momentos capturados na etapa de teste, onde a função do módulo classificador era o de, dado um conjunto de modelos, achar o objeto que representa o modelo no alcance de visão do Kinect, e, representar a classificação visualmente por meio de uma caixa delimitadora

(*bounding box* - caixa vermelha), conforme pode ser visto nas Figuras 4.10, 4.11 e 4.12, que demostram instantes diferentes do processo.

Figura 4.10 – Instantes de tempo da nuvem no período de teste em diferentes ângulos.



Fonte: Imagem de autoria do autor.

A Figura 4.10 demonstra dois pontos de vista de um mesmo instante, a esquerda, uma visão referencial ao sensor, com um mapa de profundidade, e a direita uma visão tridimensional dos dados obtidos e processados. Em 90% das imagens mostradas na Figura

[4.10](#) é visto um bom funcionamento do sistema, onde o método foi capaz de classificar qual o objeto correto. Um detalhe é que a *bounding box* de marcação se limita ao tamanho da superfície captada pelo sensor, e não ao tamanho real do objeto.

Na Figura [4.10c](#) é observado um caso de falso negativo, onde nenhum dos objetos na cena possuiu um índice de semelhança aceitável para a classificação em um conjunto de *frames*, e nesse caso, a *bounding box* fica com a posição e dimensão do último registro classificado. Isso pode ocorrer por inúmeros fatores. Os mais comuns registrados foram: o ângulo de visão não possuía registros de modelos para a realização de uma boa comparação, ou, por conta da distância e de uma falha ocasional do sensor a leitura apresenta dados imprecisos ou inválidos.

As Figuras [4.11](#) e [4.12](#) apresentam uma visão tridimensional superior da nuvem, captando outros instantes do teste. Nelas são vistos o funcionamento correto do método em diferentes ângulos de visão do objeto em análise, invariante a distância ou escala do mesmo.

Outro caso de falso negativo pode ser visualizado na Figura [4.12m](#), assim como o ocorrido em [4.10c](#), onde por conta de uma leitura ruim ou falha do conjunto de treino não é realizada uma boa leitura do objeto, e o mesmo não é classificado. Na Figura [4.12o](#) ocorre um caso de falso positivo, onde outro objeto estático da cena é classificado erroneamente de acordo com o modelo. Um método de filtro de ruído nesse caso (mudanças abruptas podem ser encaradas como ruído de leitura) poderiam melhorar o processo, de modo que os módulos de associação e classificação cubram as falhas um do outro.

Apesar do sistema proposto detectar falsos positivos, ele foi capaz de detectar e rastrear o objeto alvo de forma bem efetiva, como observado nas figuras mostradas.

4.3.3 Avaliação Quantitativa

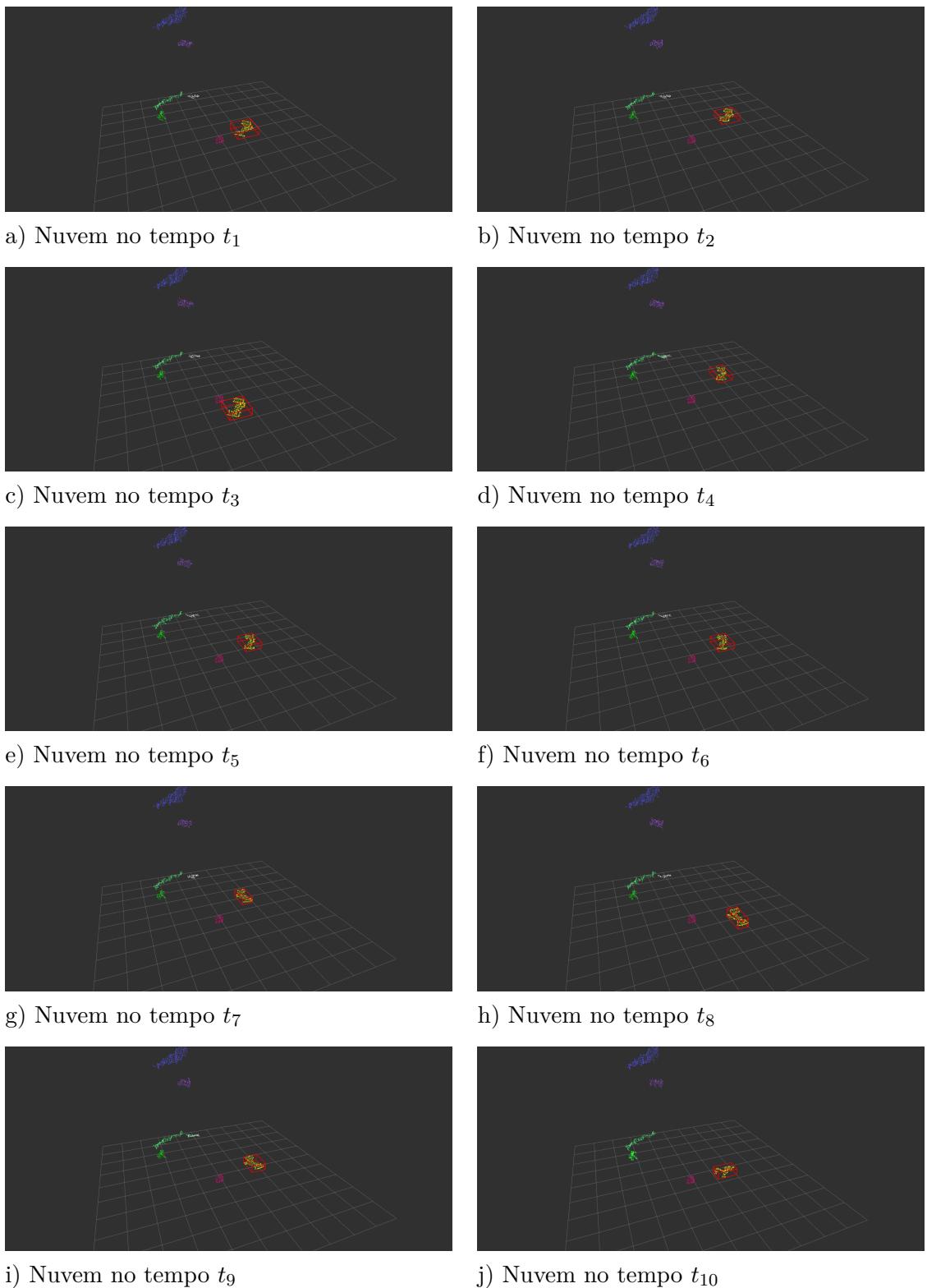
Para realizar à avaliação quantitativa do método de classificação proposto neste trabalho, são utilizados dois índices avaliativos: o índice de precisão e o índice de revocação. O índice de precisão é expresso por uma relação entre a quantidade de acertos (verdadeiros positivos) e a quantidade total de classificações realizadas.

Nesse trabalho uma classificação correta é definida como um verdadeiro positivo (*vp*), uma classificação errada como falso positivo (*fp*), e uma não classificação como um falso negativo (*fn*).

O índice de precisão possui resultado entre zero e um, indicando a porcentagem de precisão e pode ser definido como:

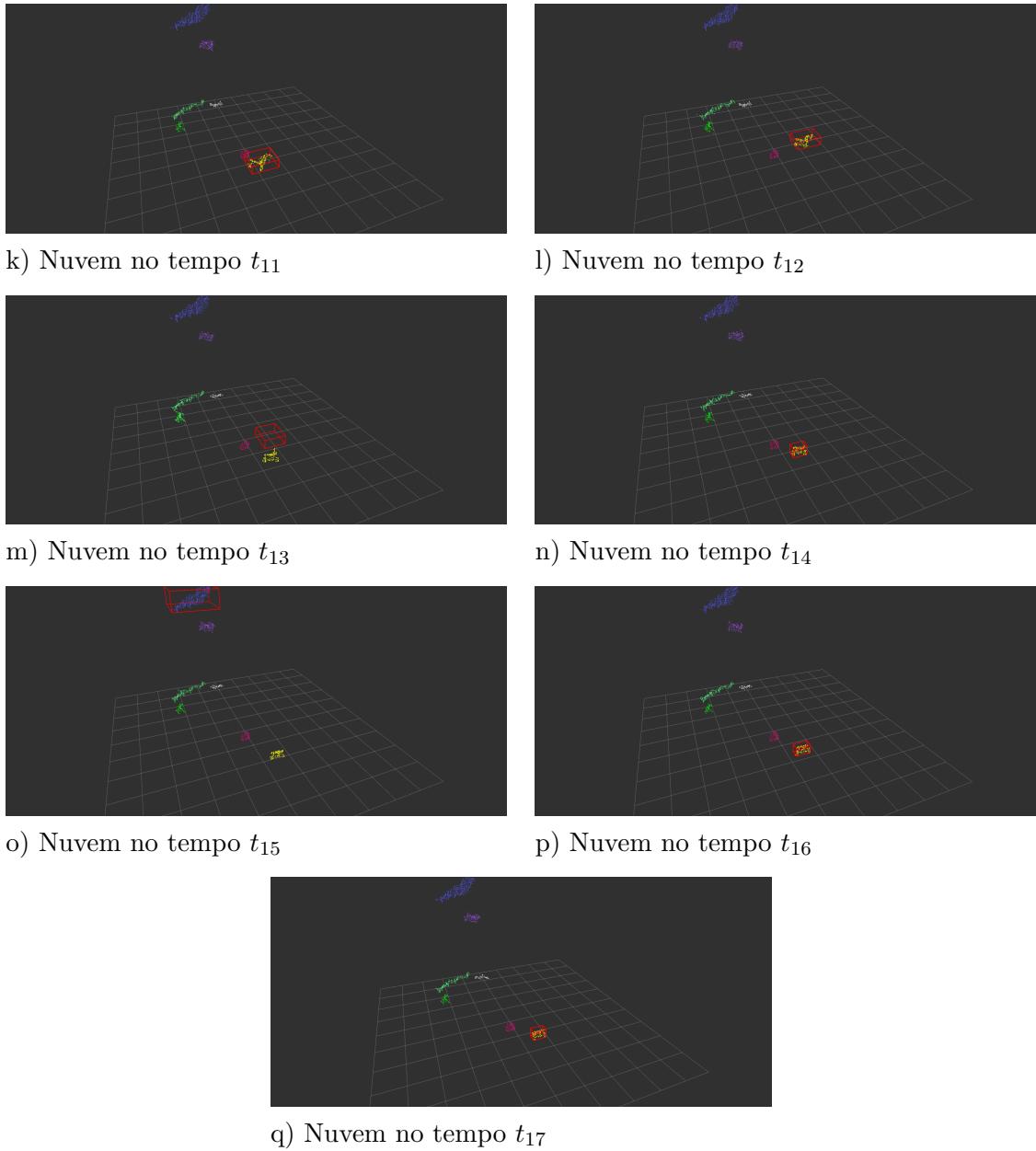
$$\text{precisão} = \frac{vp}{vp + fp}$$

Figura 4.11 – Diferentes instantes de tempo da nuvem no período de teste



Fonte: Imagem de autoria do autor.

Figura 4.12 – Diferentes instantes de tempo da nuvem no período de teste



Fonte: Imagem de autoria do autor.

O índice de revocação, por outro lado, expressa a relação entre a quantidade de classificações corretamente realizadas de um objeto na cena e a quantidade real de amostragem desse mesmo objeto. Também possui um resultado que varia entre zero e um, e sua equação é expressa da seguinte maneira:

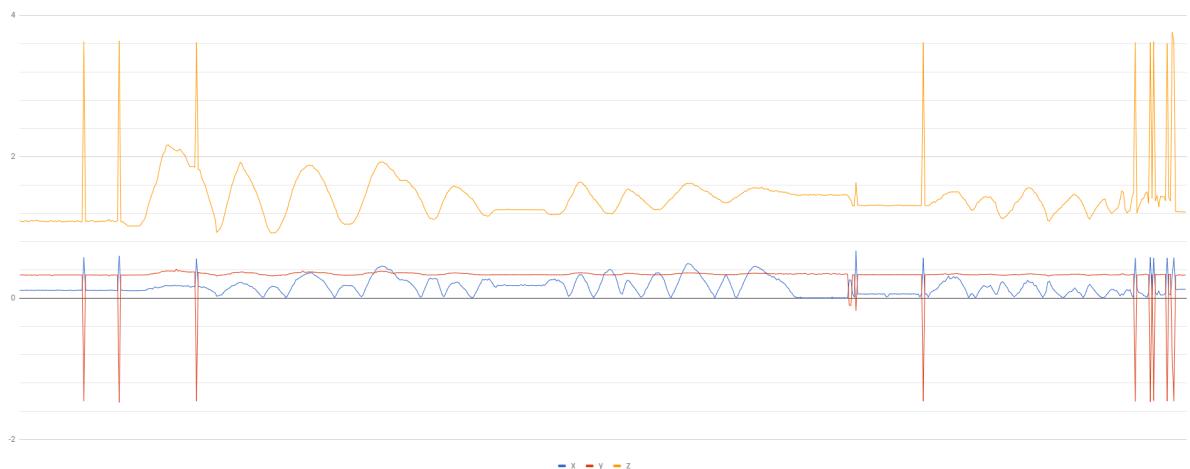
$$\text{revocação} = \frac{vp}{vp + fn}$$

Para a execução do teste quantitativo, neste trabalho, foi realizado o seguinte processo: durante um período de três minutos, o classificador avalia o conjunto de objetos

presentes na nuvem de pontos, calcula a distância entre os histogramas do objeto com os modelos, e obtém o modelo com maior semelhança. Após isso, caso esse índice seja menor que um valor limiar pré-determinado ($\text{limiar} = 70$, seção 4.3.1), uma caixa delimitadora vermelha de dimensões semelhantes ao do objeto é adicionada à cena na mesma posição que o objeto classificado, indicando visualmente qual o objeto classificado, conforme visto nas Figuras 4.11, 4.12 e 4.10. Simultaneamente, é feito um registro da posição do centro de massa do objeto alvo com referência ao sensor (posição do robô).

Ao final do teste foi obtido um conjunto de 1082 leituras (*frames*), sendo que em 694 *frames* foram realizadas classificações e em 388 dos *frames* o objeto em análise não obteve um valor mínimo de semelhança, conforme pode ser visto na Tabela 4.4. Com base nas leituras onde ocorreram classificações, foi gerado um histograma da variação das coordenadas (x, y, z) do objeto classificado no decorrer do tempo, onde a cor amarela representa o eixo z , a cor azul o eixo x e a cor vermelha o eixo y , conforme Figura 4.13.

Figura 4.13 – Variação do deslocamento de objetos classificados.



Fonte: Imagem de autoria do autor.

Para realizar a contagem de verdadeiros positivos vp , de falsos positivos fp e de falsos negativos fn , foi observado o histograma gerado a partir da variação das coordenadas (x, y, z) do objeto classificado (Figura 4.13). Quando um objeto (diferente do objeto alvo) é classificado erroneamente, ocorre uma mudança repentina no centro de massa do objeto classificado, gerando momentos de mudanças abruptas de valores em um dos três eixos, conforme demonstrado no histograma. Essas ocorrências (mudanças abruptas da coordenada do centro de massa do objeto) são classificadas como fp . Todos os outros valores de coordenadas de objetos classificados são quantificados como vp . E, em todos os *frames* que não houve classificação são contabilizados como fn . Durante a avaliação foram quantificados 681 verdadeiros positivos, 388 falsos negativos e 13 falsos positivos.

Tabela 4.4 – Dados obtidos dos testes.

Resultados dos testes		
Leituras = 1082	Classificadas = 694	681 <i>vp</i>
		13 <i>fp</i>
	Não classificadas = 388	388 <i>fn</i>

Desta forma, obteve-se uma taxa de precisão de 98,12% na classificação. Para a taxa de revocação obteve-se um índice de 63,70%, o que condiz com a média de acertos verificada na distribuição de frequência dos *scores* obtida durante a escolha do limiar (seção 4.3.1, Figura 4.9). Os resultados da precisão e revocação podem ser visualizados na Tabela 4.5.

Tabela 4.5 – Índices avaliativos da classificação.

Resultados dos testes	
precisão	98,12%
revocação	63,70%

4.4 Considerações

Após análise dos experimentos, pode-se verificar que o sistema apresentou boa eficiência em detectar e rastrear um objeto alvo através de dados de uma nuvem de pontos, sendo capaz de processar as informações obtidas com um bom desempenho de execução, podendo ser aplicado em sistemas de tempo real.

Quanto a classificação, nos testes é observado que o método apresenta alta taxa de precisão e uma taxa média de revocação. Isso significa que o método classifica corretamente o modelo em 98,12% das vezes, porém só realiza a classificação em 63,70% das leituras (*frame*). O que define o valor dessas taxas é o valor do limiar escolhido para a realização da classificação por meio do KNN. Um maior valor do limiar implica diretamente em aumento da taxa de revocação, pois um maior conjunto de objetos serão classificados, porém, com menor taxa de precisão, já que mais objetos podem ser erroneamente aceitos como pertencentes ao modelo.

Como o sistema proposto neste trabalho tem por objetivo detectar e rastrear um objeto alvo no campo visual de um robô utilizando sensor RGB-D, optou-se por utilizar uma abordagem com maior precisão e menor revocação. O sistema tem a capacidade de realizar uma média de 6 leituras processadas por segundo, e com uma taxa de revocação de 63,70%, ele é capaz de perceber e classificar o objeto em aproximadamente 4 dos 6 *frames* processados, o que pode ser considerado suficientemente bom para a execução da tarefa. Além disso, é mais relevante que o sistema tenha eficiência para perseguir o objeto

correto, do que possuir uma menor precisão, já que isso pode ocasionar uma perseguição ao alvo incorreto.

5 Considerações Finais

Neste trabalho, foi investigado o problema de detecção e rastreamento de um objeto por meio da análise de nuvem de pontos 3D, e foi proposto um sistema de detecção e rastreamento de um objeto alvo no ambiente à frente de um robô autônomo usando o sensor RGB-D Kinect.

O sistema proposto desenvolvido neste trabalho opera em quatro etapas. A cada leitura do sensor é realizado o pré-processamento, a segmentação, a associação e a classificação da nuvem de pontos 3D. Na etapa de pré-processamento ocorre a remoção dos pontos 3D do chão e a redução do conjunto de amostragem. Na etapa de segmentação, a nuvem de pontos 3D é segmentada em agrupamentos de pontos usando a distância Euclidiana, sendo que cada agrupamento representa um objeto no ambiente. Na etapa de associação, os objetos observados na varredura atual do sensor são associados aos mesmos objetos observados em varreduras anteriores usando o algoritmo do vizinho mais próximo (*nearest neighbor*). Em seguida, os objetos são classificados por meio do método de descrição VFH e um classificador KNN através da distância entre os histogramas, com base em um conjunto de treino, indicando se pertencem ou não ao conjunto de modelos do objeto alvo. Por fim, uma caixa delimitadora é inserida sobre a nuvem de pontos 3D usando como referência o centro de massa do objeto classificado como objeto alvo.

O desempenho do sistema proposto foi avaliado por meio de análises qualitativas e quantitativas utilizando dados de um sensor Kinect coletados durante um período de teste em um ambiente *indoor*. Os resultados experimentais mostraram que o sistema proposto foi capaz de detectar, classificar e rastrear o objeto alvo com bom desempenho.

5.1 Trabalhos Futuros

Com base nos testes realizados no sistema desenvolvido neste trabalho e na revisão literária realizada, possíveis trabalhos futuros para melhoria ou comparação podem ser realizados, como:

1. Implementação e comparação de outros métodos de segmentação de nuvem de pontos.
2. Implementação e comparação de outros métodos de associação temporal de uma nuvem de pontos.
3. Implementação de um método probabilístico para rastreamento dos objetos (i.e., filtro de partículas) ou um filtro de *Kalman*.

4. Utilização de métodos mais robustos para realização da classificação dos histogramas dos objetos como SVM ou uma Rede Neural.
5. Realização das etapas sem a redução de amostragem para avaliar o impacto qualitativo da mesma, utilizando sistemas de computação de alto desempenho (i.e., CUDA).
6. Implementação de múltiplos métodos descritivos para nuvem de pontos 3D na realização da classificação.

Referências

- ALI, H.; FIGUEROA, N. Segmentation and pose estimation of planar metallic objects. *Proceedings of the 2012 9th Conference on Computer and Robot Vision, CRV 2012*, n. i, p. 376–382, 2012. Citado na página 43.
- ALMEIDA, A.; ALMEIDA, J.; Rui Araujo. Real-Time Tracking of Moving Objects Using Particle Filters. In: *Proceedings of the IEEE International Symposium on Industrial Electronics, 2005. ISIE 2005*. IEEE, 2005. IV, p. 1327–1332. ISBN 0-7803-8738-4. ISSN 00222275. Disponível em: <<http://ieeexplore.ieee.org/document/1529124/>>. Citado 2 vezes nas páginas 16 e 30.
- ALTMAN, N. S. An Introduction to Kernel and Nearest-Neighbor Nonparametric Regression. *The American Statistician*, v. 46, n. 3, p. 175–185, aug 1992. ISSN 0003-1305. Disponível em: <<http://www.tandfonline.com/doi/abs/10.1080/00031305.1992.10475879>>. Citado 2 vezes nas páginas 24 e 28.
- AMARAL, E. M. A. et al. Detecção e rastreamento de veículos em movimento para automóveis robóticos autônomos. *XII Simpósio Brasileiro de Automação Inteligente (SBAI)*, n. 2015, p. 801–806, 2015. Citado 4 vezes nas páginas 16, 19, 32 e 43.
- AZIM, A.; AYCARD, O. Detection, classification and tracking of moving objects in a 3d environment. *Intelligent Vehicles Symposium (IV), 2012 IEEE*, p. 802–807, 2012. ISSN 1931-0587. Citado na página 47.
- BAY, H. et al. Speeded-Up Robust Features (SURF). *Computer Vision and Image Understanding*, v. 110, n. 3, p. 346–359, 2008. ISSN 10773142. Citado 2 vezes nas páginas 16 e 23.
- BECKER, M. et al. 2D laser-based probabilistic motion tracking in urban-like environments. *Journal of the Brazilian Society of Mechanical Sciences and Engineering*, v. 31, n. 2, p. 83–96, 2009. ISSN 16785878. Disponível em: <<http://ieeexplore.ieee.org/document/6568193/>>. Citado na página 47.
- BĘDKOWSKI, J. et al. Intelligent Mobile System for Improving Spatial Design Support and Security Inside Buildings. *Mobile Networks and Applications*, v. 21, n. 2, p. 313–326, apr 2016. ISSN 1383-469X. Disponível em: <<http://link.springer.com/10.1007/s11036-015-0654-8>>. Citado na página 24.
- BENTLEY, J. L. Multidimensional binary search trees used for associative searching. *Commun. ACM*, ACM, New York, NY, USA, v. 18, n. 9, p. 509–517, set. 1975. ISSN 0001-0782. Disponível em: <<http://doi.acm.org/10.1145/361002.361007>>. Citado na página 44.
- BLACKMAN, S. Multiple hypothesis tracking for multiple target tracking. *IEEE Aerospace and Electronic Systems Magazine*, v. 19, n. 1, p. 5–18, jan 2004. ISSN 0885-8985. Disponível em: <<http://ieeexplore.ieee.org/document/1263228/>>. Citado na página 31.
- BOUCHER, S. Obstacle Detection and Avoidance Using TurtleBot Platform and XBox Kinect. 2012. Citado 2 vezes nas páginas 42 e 43.

- Chieh-Chih Wang; THORPE, C.; THRUN, S. Online simultaneous localization and mapping with detection and tracking of moving objects: theory and results from a ground vehicle in crowded urban areas. In: *2003 IEEE International Conference on Robotics and Automation (Cat. No.03CH37422)*. IEEE, 2003. v. 1, p. 842–849. ISBN 0-7803-7736-2. Disponível em: <<http://ieeexplore.ieee.org/document/1241698/>>. Citado 3 vezes nas páginas 16, 31 e 32.
- CHO, K.; BAEG, S.; PARK, S. Object tracking with enhanced data association using a 3D range sensor for an unmanned ground vehicle. *Journal of Mechanical Science and Technology*, v. 28, n. 11, p. 4381–4388, nov 2014. ISSN 1738-494X. Disponível em: <<http://link.springer.com/10.1007/s12206-014-1005-6>>. Citado na página 29.
- CORTES, C.; VAPNIK, V. Support-vector networks. *Machine Learning*, v. 20, n. 3, p. 273–297, sep 1995. ISSN 0885-6125. Disponível em: <<http://link.springer.com/10.1007/BF00994018>>. Citado 3 vezes nas páginas 24, 27 e 28.
- COX, I. J. A review of statistical data association techniques for motion correspondence. *International Journal of Computer Vision*, v. 10, n. 1, p. 53–66, feb 1993. ISSN 0920-5691. Disponível em: <<http://link.springer.com/10.1007/BF01440847>>. Citado 2 vezes nas páginas 30 e 31.
- DOUILLARD, B. et al. On the segmentation of 3D LIDAR point clouds. *2011 IEEE International Conference on Robotics and Automation*, p. 2798–2805, 2011. ISSN 10504729. Disponível em: <<http://ieeexplore.ieee.org/document/5979818/>>. Citado 2 vezes nas páginas 37 e 39.
- DROST, B. et al. Model globally, match locally: Efficient and robust 3D object recognition. In: *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE, 2010. p. 998–1005. ISBN 978-1-4244-6984-0. ISSN 10636919. Disponível em: <<http://ieeexplore.ieee.org/document/5540108/>>. Citado na página 24.
- EL-LAITHY, R. A.; HUANG, J.; YEH, M. Study on the use of Microsoft Kinect for robotics applications. In: *Proceedings of the 2012 IEEE/ION Position, Location and Navigation Symposium*. IEEE, 2012. p. 1280–1288. ISBN 978-1-4673-0387-3. ISSN 2153-358X. Disponível em: <<http://ieeexplore.ieee.org/document/6236985/>>. Citado na página 36.
- ESTER, M. et al. A density-based algorithm for discovering clusters a density-based algorithm for discovering clusters in large spatial databases with noise. In: *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*. AAAI Press, 1996. (KDD'96), p. 226–231. Disponível em: <<http://dl.acm.org/citation.cfm?id=3001460.3001507>>. Citado na página 20.
- FISCHLER, M. a.; BOLLES, R. C. Paradigm for Model. *Communications of the ACM*, v. 24, n. 6, p. 381–395, 1981. Citado na página 37.
- GRITTI, A. P. et al. Kinect-based people detection and tracking from small-footprint ground robots. *IEEE International Conference on Intelligent Robots and Systems*, n. Iros, p. 4096–4103, 2014. ISSN 21530866. Citado 2 vezes nas páginas 16 e 19.
- HAMEIRI, E.; SHIMSHONI, I. Estimating the principal curvatures and the darboux frame from real 3-d range data. *IEEE Transactions on Systems, Man, and Cybernetics*,

- Part B (*Cybernetics*), v. 33, n. 4, p. 626–637, Aug 2003. ISSN 1083-4419. Citado na página 50.
- HAUSMAN, K. et al. Tracking-based interactive segmentation of textureless objects. *Proceedings - IEEE International Conference on Robotics and Automation*, p. 1122–1129, 2013. ISSN 10504729. Citado na página 43.
- HOLZER, S. et al. Real-Time Surface Normal Estimation from Organized Point Cloud Data Using Integral Images. *Computer*, p. 2684–2689, 2012. ISSN 21530858. Citado 2 vezes nas páginas 49 e 50.
- LI, S. *Robots are becoming security guards. 'Once it gets arms ... it'll replace all of us.'*. 2016. Disponível em: <<http://www.latimes.com/business/la-fi-robots-retail-20160823-snap-story.html>>. Citado na página 15.
- LOWE, D. G. Object recognition from local scale-invariant features. *Proceedings of the Seventh IEEE International Conference on Computer Vision*, v. 2, n. [8, p. 1150–1157, 1999. ISSN 0-7695-0164-8. Disponível em: <<http://ieeexplore.ieee.org/document/790410/>>. Citado 2 vezes nas páginas 16 e 23.
- MALISIEWICZ, T.; EFROS, a. a. Improving Spatial Support for Objects via Multiple Segmentations. *Procedings of the British Machine Vision Conference 2007*, p. 55.1–55.10, 2007. Disponível em: <<http://www.bmva.org/bmvc/2007/papers/paper-304.html>>. Citado na página 42.
- MORAL, P. del. Non {Linear Filtering: Interacting Particle Solution}. *Markov Processes and Related Fields*, v. 2, n. 4, p. 555–580, 1996. Citado na página 31.
- MOSANYA, L. *'Crazy violence' in Brazilian state during police strike*. 2017. Disponível em: <<http://www.bbc.co.uk/newsbeat/article/38942911/crazy-violence-in-brazilian-state-during-police-strike>>. Citado na página 14.
- PANIAGUA, F. S. I. Object Recognition using the Kinect. 2011. Citado na página 16.
- PELE, O.; WERMAN, M. The Quadratic-Chi Histogram Distance Family. In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. [s.n.], 2010. v. 6312 LNCS, n. PART 2, p. 749–762. ISBN 3642155510. Disponível em: <http://link.springer.com/10.1007/978-3-642-15552-9_54>. Citado na página 52.
- PETROVSKAYA, A. et al. Awareness of Road Scene Participants for Autonomous Driving. In: *Handbook of Intelligent Vehicles*. London: Springer London, 2012. p. 1383–1432. ISBN 978-0-85729-084-7. Disponível em: <<https://cs.stanford.edu/people/petrovsk/dn/publications/anya-datmo-handbook2012.pdf>>. Citado na página 19.
- PRIBERAM. *O Dicionário Priberam da Língua Portuguesa (DPLP) é um dicionário de português contemporâneo que contém mais de 110 000 entradas lexicais, incluindo locuções e fraseologias, cuja nomenclatura compreende o vocabulário geral, bem como os termos mais comuns das principais áreas científicas e técnicas*. 2016. Disponível em: <<https://www.priberam.pt/dlpo/>>. Citado na página 14.

- ROGERS, J. *Robot patrol: Israeli Army to deploy autonomous vehicles on Gaza border*. 2016. Disponível em: <<http://www.foxnews.com/tech/2016/09/01/robot-patrol-israeli-army-to-deploy-autonomous-vehicles-on-gaza-border.html>>. Citado na página 15.
- ROS. *Is a flexible framework for writing robot software*. 2016. Disponível em: <<http://www.ros.org/>>. Citado na página 34.
- RUSU, R. B. Semantic 3D Object Maps for Everyday Manipulation in Human Living Environments. *KI - Kunstliche Intelligenz*, v. 24, p. 345–348, 2010. ISSN 09331875. Citado na página 50.
- RUSU, R. B.; BLODOW, N.; BEETZ, M. Fast Point Feature Histograms (FPFH) for 3D registration. In: *2009 IEEE International Conference on Robotics and Automation*. IEEE, 2009. p. 3212–3217. ISBN 978-1-4244-2788-8. ISSN 1050-4729. Disponível em: <<http://ieeexplore.ieee.org/document/5152473/>>. Citado 4 vezes nas páginas 16, 26, 27 e 50.
- RUSU, R. B. et al. Fast 3D recognition and pose using the Viewpoint Feature Histogram. In: *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2010. p. 2155–2162. ISBN 978-1-4244-6674-0. ISSN 2153-0858. Disponível em: <<http://ieeexplore.ieee.org/document/5651280/>>. Citado 5 vezes nas páginas 16, 26, 49, 51 e 52.
- RUSU, R. B.; COUSINS, S. 3D is here: point cloud library. *IEEE International Conference on Robotics and Automation*, p. 1 – 4, 2011. ISSN 1050-4729. Disponível em: <<http://pointclouds.org/>>. Citado 4 vezes nas páginas 16, 34, 40 e 55.
- RUSU, R. B. et al. Learning informative point classes for the acquisition of object model maps. In: *2008 10th International Conference on Control, Automation, Robotics and Vision*. IEEE, 2008. p. 643–650. ISBN 978-1-4244-2286-9. Disponível em: <<http://ieeexplore.ieee.org/document/4795593/>>. Citado 3 vezes nas páginas 16, 24 e 25.
- SCHILLER, B. *meet the scary little security robot thats patrolling silicon valley*. 2015. Disponível em: <<https://www.fastcompany.com/3049708/meet-the-scary-little-security-robot-thats-patrolling-silicon-valley>>. Citado na página 15.
- SCHULZ, D. et al. Tracking multiple moving objects with a mobile robot. In: *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*. IEEE Comput. Soc, 2001. v. 1, p. I-371–I-377. ISBN 0-7695-1272-0. ISSN 1063-6919. Disponível em: <<http://ieeexplore.ieee.org/document/990499/>>. Citado 2 vezes nas páginas 16 e 30.
- SPINELLO, L. et al. A layered approach to people detection in 3d range data. In: *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence*. AAAI Press, 2010. (AAAI'10), p. 1625–1630. Disponível em: <<http://dl.acm.org/citation.cfm?id=2898607.2898866>>. Citado 2 vezes nas páginas 22 e 23.
- STEDER, B. et al. Point feature extraction on 3D range scans taking into account object boundaries. *Proceedings - IEEE International Conference on Robotics and Automation*, p. 2601–2608, 2011. ISSN 10504729. Citado na página 16.

- TELLAECHE, A.; MAURTUA, I. 6DOF pose estimation of objects for robotic manipulation. A review of different options. *19th IEEE International Conference on Emerging Technologies and Factory Automation, ETFA 2014*, 2014. Citado na página 43.
- THRUN, S. Particle Filters in Robotics (Invited Talk). dec 2012. ISSN 00222275. Disponível em: <<http://arxiv.org/abs/1301.0607>>. Citado na página 33.
- THRUN, S. et al. Stanley: The robot that won the DARPA Grand Challenge. *Journal of Field Robotics*, v. 23, n. 9, p. 661–692, sep 2006. ISSN 15564959. Disponível em: <<http://doi.wiley.com/10.1002/rob.20147>>. Citado na página 19.
- THRUN WOLFRAM BURGARD, D. F. S. *Probabilistic Robotics: Intelligent robotics and autonomous agents*. [S.l.]: MIT Press, 2005. Citado na página 14.
- VU, T.-D.; AYCARD, O.; APPENRODT, N. Online Localization and Mapping with Moving Object Tracking in Dynamic Outdoor Environments. In: *2007 IEEE Intelligent Vehicles Symposium*. IEEE, 2007. p. 190–195. ISBN 1-4244-1067-3. ISSN 1931-0587. Disponível em: <<http://ieeexplore.ieee.org/document/4290113/>>. Citado na página 47.
- WAHL, E.; HILLENBRAND, U.; HIRZINGER, G. Surflet-pair-relation histograms: a statistical 3D-shape representation for rapid classification. In: *Fourth International Conference on 3-D Digital Imaging and Modeling, 2003. 3DIM 2003. Proceedings*. IEEE, 2003. v. 2003-Janua, n. October, p. 474–481. ISBN 0-7695-1991-1. ISSN 15506185. Disponível em: <<http://ieeexplore.ieee.org/document/1240284/>>. Citado na página 25.

Apêndice A – Código Fonte

O código fonte do projeto pode ser encontrado em:

<https://github.com/leoCamilo/ObjectRecognition>