CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE MINAS GERAIS

Rainara Araújo Mateus

UTILIZANDO NUVENS DE PONTOS NA DETECÇÃO E CLASSIFICAÇÃO DE OBSTÁCULOS EM SUPERFÍCIE

Timóteo

2017

Rainara Araújo Mateus

UTILIZANDO NUVEM DE PONTOS NA DETECÇÃO DE OBSTÁCULOS EM SUPERFÍCIE

Monografia apresentada ao Curso de Engenharia de Computação do Centro Federal de Educação Tecnológica de Minas Gerais para a obtenção do título de Engenheiro de Computação.

Orientador: Odilon Corrêa da Silva

Timóteo
2017

Rainara Araújo Mateus

UTILIZANDO NUVEM DE PONTOS NA DETECÇÃO DE OBSTÁCULOS EM SUPERFÍCIE

Monografia apresentada ao Curso de Engenharia de Computação do Centro Federal de Educação Tecnológica de Minas Gerais para a obtenção do título de Engenheiro de Computação.

Odilon Corrêa da Silva
Orientador

Prof. João Batista Queiroz Zuliane
Professor Convidado

Prof. Douglas Nunes de Oliveira
Professor Convidado

Timóteo
2017

AGRADECIMENTOS

A presente monografia representa o fim de uma das fases mais importantes da minha vida. Passei por caminhos desafiadores, mas o apoio e suporte de algumas pessoas foram fundamentais ao longo dessa etapa.

Primeiramente, gostaria de agradecer aos meus pais, Elvira e José Raimundo, que sempre me apoiaram e batalharam arduamente durante toda a minha graduação, acreditando na importância de uma educação melhor para seus filhos. Aos meus irmãos, Renata, Rafael e Raíssa, pela paciência e companheirismo. Aos meus padrinhos Igor e Amanda por terem sempre acreditado indubitavelmente no meu potencial mesmo quando eu mesma não acreditava, me apresentado ao método Kumon e me dado todo o suporte escolar desde o meu Ensino Fundamental.

Á todos os professores do curso de Engenharia de Computação, em especial aos professores Éder Rodrigues, Marcelo Balbino, Deisymar Botega, Rutyele Moreira, Júlio César, João Batista, Rodrigo Gaiba e Douglas Oliveira, os quais são profissionais de alta competência e que sempre tive um enorme respeito e admiração.

Aos meus preciosos amigos Matheus Assunção, Débora Cristina, Walace Andrade, Beatriz Vieira, Ramon Meira e Priscila Carolini pela consideração, apoio, carinho e amizade nos momentos que mais precisei. Ao aluno Josué Rocha por ter sido uma contribuição ímpar nos momentos em que tive dúvida.

A todas pessoas incríveis que conheci durante o período em que fiz intercâmbio as quais tenho o prazer de poder chamar de amigos, em especial a Patrícia Guedes, Amanda Lima, Thales Beraldo, David Guimarães, Indianara Valaski, Juliana Pace, Daniela Milreu, Edson Pinheiro, Aline Fedoce e André Fontolan.

Ao meu melhor amigo Marcos Guimarães pela sua contínua motivação e apoio durante em todo esse período, sempre procurando me aconselhar em todas as situações e me ajudando a encontrar soluções nos meus momentos de incerteza, sendo o meu escudo durante esses sete anos de amizade.

Por fim, ao CEFET-MG campus Timóteo que me possibilitou passar por tantas experiências engrandecedoras, tanto no âmbito profissional quanto pessoal, e que tenho muito orgulho de fazer parte do corpo docente. Em particular, agradeço ao meu orientador, Prof. Odilon Corrêa da Silva, o qual considero um grande amigo, por todo o empenho em me auxiliar ao longo de todo o processo com muita paciência e dedicação.

“There is never good enough, there must always be something better – there is no progress when we think: *“This is good enough.” Even if you currently do not have confidence in yourself, you should have confidence in the you of tomorrow, believing you will be able to instruct better tomorrow by continuing to learn today.”*

Toru Kumon

RESUMO

Devido a crescente busca pela compreensão e extração de informações de imagens tridimensionais, surgiram diversas formas responsáveis por facilitar a manipulação desses dados, dentre as quais tem-se as nuvens de ponto. Considerando a aplicabilidade das nuvens de ponto, este trabalho propõe a utilização desse recurso para a detecção e classificação de obstáculos, avaliando o desempenho de duas técnicas de segmentação distintas: *Euclidean Segmentation* e *Region Growing*. Para tanto, foram desenvolvidos e implementados dois modelos de classificação para cada técnica, utilizando os recursos da biblioteca PCL. A validação dos modelos foi realizada através de 3 casos de testes, onde os objetos no cenário foram distribuídos de forma distintas e classificados de acordo com a altura. Por meio do estudo de caso, pôde-se constatar resultados similares entre os modelos (quando o ângulo de suavidade apresenta valor de 7π rad). Isso mostrou que a adoção e especificação de uma técnica de segmentação adequadamente permite a extração e manipulação dos dados capturados.

Palavras-chave: Nuvens de Ponto - Euclidean Segmentation - Region Growing - PCL

ABSTRACT

Due to the increasing search for the understanding and extraction of information from three-dimensional images, several methods have appeared to facilitate the manipulation of this data, among them there is the called point cloud. Considering the applicability of the point clouds, this work proposes the use of this resource for the detection and classification of obstacles, evaluating the performance of two different segmentation techniques: *Euclidian Segmentation* and *Region Growing*. Thus, two classification models were developed and implemented for each technique, using PCL library features. The models validation was accomplished through 3 test cases, where the objects of the scenery were distributed in different forms and classified according to their heights. Through the case study, it was possible to verify similar results between the models (when the angle of softness presents value of 7π rad). This showed that adopting and specifying a segmentation technique properly allows the extraction and manipulation of the captured data.

Keywords: Point Clouds - Euclidian Segmentation - Regional Growing - PCL

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

PCL	*Point Cloud Library*
RGBD	*Red-Green-Blue-Depth*
ROS	*Robot Operating System*

SUMÁRIO

1 INTRODUÇÃO

As habilidades de extração, processamento e interpretação de dados visuais permitem-nos perceber o mundo de objetos tridimensionais com muitas propriedades invariantes de forma coerente (BALLARD & BROWN, 1982). A partir de sua capacidade de captar, processar e interpretar grandes volumes de dados, o sistema visual humano consegue interpretar e/ou impor ações em dados de entrada caóticos (PEDRINI & SCHWARTZ, 2008).

Segundo Teixeira (2011), antigamente a inspeção visual e definição de rotas eram os principais requisitos para métodos de captura tridimensional. Entretanto, com o aumento da demanda por conteúdo 3D criou novos desafios para a análise da visão computacional no registro e mapeamento de superfícies/objetos, onde a qualidade da informação da imagem se tornou o requisito primordial nas aplicações.

Com a evolução da aplicação de recursos computacionais e a crescente necessidade de compreensão das informações visuais obtidas em diversas atividades, a visão computacional surgiu com a finalidade de emular a visão humana, incluindo o aprendizado e a capacidade de fazer inferências e tomar decisões com base nas informações contidas em imagens (GONZALES & WOODS, 2010). Por conseguinte, um dos grandes desafios da visão computacional é prover aos sistemas computacionais a habilidade de interpretar e reagir aos ruídos ou outras imperfeições intrínsecas das imagens.

De acordo com Gois (2004), existem diversos algoritmos de reconstrução de superfície que apresentam diversas aplicações. Esses algoritmos podem ser subdivididos em duas classes: *Reconstrução a partir de Seções Planares* e *Reconstrução a partir de Nuvens de Pontos*. A primeira classe de algoritmos surgiu na área de Imagens Médicas, as quais adotam equipamentos que capturam sequências bidimensionais de um objeto tridimensional. A segunda classe, de origem mais recente, abrange algoritmos que capturam uma amostragem de pontos através de dispositivos, como *scanners 3D*.

Segundo Palnick (2014), uma nuvem de pontos (*Point* Cloud), consiste em um

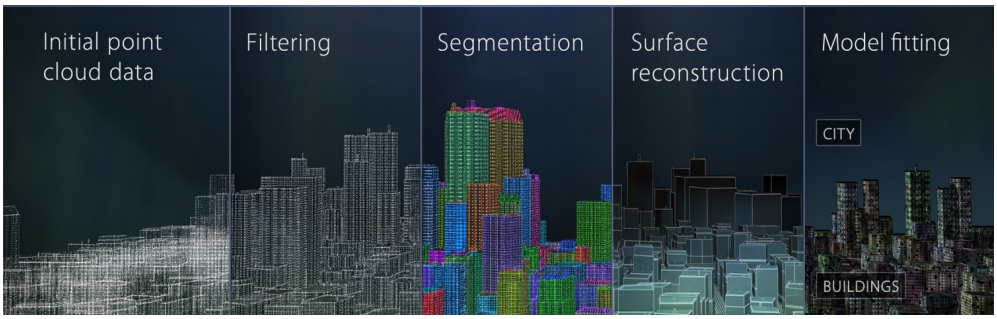
conjunto de dados atribuídos a um sistema de coordenadas, o qual, além da posição de coordenada XYZ de cada ponto da área processada, permite a inclusão de informações adicionais como cor e intensidade. Essa estrutura tem sido amplamente adotado para o registro tridimensional devido aos recentes avanços das tecnologias de aquisição volumétrica, tais como, *Kinect*, *Xton* e *Scanners 3D*, que utilizam algoritmos de reconhecimento e extração de informações e propriedades dos dados registrados. Pelo fato da nuvem de pontos possuir um formato simples e flexível para a representação de geometrias complexas, um sistema computacional que utilize nuvens de pontos pode ser utilizado na detecção e classificação de obstáculos em superfície plana e aplicado a diversos contextos. Entre outros, podem ser citados:

- Detecção e classificação de obstáculos (NGUYEN, 2012);
- Percepção em Sistemas Autônomos (QUEIRÓS, 2012);
- Registro e classificação de objetos em ambientes internos (HUANG, HSIEH, & YEH, 2015);
- Aquisição e processamento de modelos tridimensionais faciais (MARTINS, 2014);
- Reconhecimento de objetos baseado em visão artificial (DUARTE, 2015);
- Detecção em tempo real de regiões planas em nuvens de ponto desorganizadas (LIMBERGER & OLIVEIRA, 2015);
- Robótica Autônoma e reconstrução 3D (SANTOS, 2016);

1.1 O PROBLEMA E SUA IMPORTÂNCIA

Segundo Aldoma (2012), nuvens de ponto podem ser utilizadas para reconhecer e estimar a posição de objetos em diversos cenários. As etapas do processamento e análise de uma nuvem de pontos são ilustradas na figura abaixo e descritas a seguir.

Figura 1 - Etapas para reconstrução de uma superfície a partir de uma nuvem de pontos



Fonte: PCL (2011)

As etapas de aquisição da nuvem de pontos e filtro estão englobadas no pré-processamento das nuvens de ponto, o qual consiste no estágio de preparação e reconstrução da nuvem de pontos, mantendo as informações de características relevantes. Em sequência, as etapas de segmentação, reconstrução e ajuste de modelo representam o processamento das nuvens de ponto, no qual aplicam-se algoritmos aos dados pré-processados de modo a permitir a análise avançada da nuvem (GRUPO DE ROBÓTICA, 2015).

A captura dos dados de um objeto 3D demanda o armazenamento de milhares de pontos os quais são armazenados em uma única nuvem de pontos. Deste modo, extrair atributos e segmentar uma nuvem de pontos pré-processada para se obter os modelos de objetos individuais da cena (*clusters*) é uma tarefa desafiadora (SANTOS, 2016). Devido a esse fato, é necessário que os algoritmos implementados para a classificação sejam adequados para o tratamento otimizado das nuvens de pontos da cena registrada. Nesse contexto, surge a seguinte questão: como utilizar nuvem de pontos na detecção e classificação de obstáculos em superfície plana?

1.2 OBJETIVOS

1.2.1 Objetivo Geral

Este trabalho tem como objetivo geral especificar, implementar e avaliar um método de detecção e classificação de obstáculos, que atuará em ambientes internos e superfícies planas. Para tanto, foram utilizados técnicas de pré-processamento e segmentação de nuvem de pontos com o auxílio de ferramentas computacionais.

Considerando a relevância da etapa de segmentação para a classificação de objetos, o trabalho em questão prevê a realização de uma análise comparativa do desempenho de duas técnicas avançadas de segmentação, *Euclidean Segmentation* e *Region Growing*, adotando a detecção e classificação de obstáculos como estudo de caso para validação dos resultados.

1.2.2 Objetivos Específicos

Para alcançar o objetivo geral proposto, foram definidos os seguintes objetivos específicos:

- Investigar técnicas de visão computacional e manipulação de nuvem de pontos;
- Analisar os algoritmos de segmentação de nuvem de pontos;
- Especificar um processo de detecção de obstáculos em superfície plana;
- Adaptar o processo para classificar os obstáculos detectados;
- Implementar o processo com o auxílio de ferramentas computacionais;
- Avaliar o processo implementado proposto por meio de um estudo de caso.

2 MATERIAIS E MÉTODOS

Para a realização da pesquisa a respeito de conteúdos teóricos/práticos das nuvens de pontos e suas aplicações utilizou-se os seguintes materiais: dissertações, livros, artigos técnicos, teóricos e científicos acessíveis no portal da Capes, na internet e na biblioteca da própria instituição.

Para o desenvolvimento do modelo prático proposto para realizar a análise comparativa dos métodos de segmentação da nuvem de pontos, foram usados os seguintes recursos:

- Notebook com as seguintes especificações:
  - Processador: Intel® Core™ i5-2450M CPU @ 2.5GHz;
  - Memória RAM: 4GB, Single Channel DDR3, 1333MHz (1x4Gb);
  - Disco Rígido: 750GB, SATA (5400 RPM);
  - Sistema Operacional: Windows 10 Pro (Sistema Operacional de 32 bits, processador com base em x64);
- Plataforma utilizada: Visual Studio 2015
  - Microsoft Visual Studio Community 2015;
- Versão 14.0.25422.01 Update 3
- Linguagem de programação: C ++;
- Biblioteca: PCL (*Point Cloud Library*) versão 1.8.0 associado à API *OppeNI* 2.0;
- Sensor RGBD: Kinect Xbox 360 (Microsoft).

Atividades realizadas durante a elaboração do projeto:

1. Estudo direcionado sobre visão computacional e manipulação de nuvem de pontos;
2. Pesquisa e avaliação de sensores de profundidade de baixo custo que realizam a manipulação de nuvem de pontos;
3. Pesquisa e avaliação das etapas de processamento de nuvem de pontos;
4. Pesquisa e avaliação de trabalhos que realizam a manipulação de nuvem de pontos;
5. Estudo da ferramenta computacional de manipulação de nuvens de pontos;
6. Estudo e avaliação de métodos de segmentação de nuvem de pontos utilizando ferramentas computacionais;

7. Estudo e elaboração de um modelo para a implementação em tempo real do método de segmentação *Euclidian Segmentation* com a utilização de ferramentas computacionais e sensores de profundidade de baixo custo;
8. Estudo e elaboração de um modelo para a implementação em tempo real do método de segmentação *Region Growing Segmentation* com a utilização de ferramentas computacionais e sensores de profundidade de baixo custo;
9. Elaboração e realização de testes do estudo de caso para a realização da análise comparativa dos dois métodos de segmentação implementados para a detecção de obstáculos;
10. Documentação das atividades desenvolvidas;
11. Apresentação do trabalho para a banca examinadora;
12. Aplicação das correções do trabalho sugeridas pela banca examinadora.

Cronograma do Projeto:

Tabela 1 – Cronograma do Projeto

	Atividades											
Mês	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
Ago/2016	x											
Set/2016	x	x										
Out/2016			x									
Nov/2016			x									
Dez/2016				x								
Jan/2017					x	x						
Fev/2017							x					
Mar/2017							x	x				
Abr/2017								x	x			
Mai/2017									x	x		
Jun/2017									x		x	x

Fonte: elaborado pelo próprio autor

3 ESTADO DA ARTE

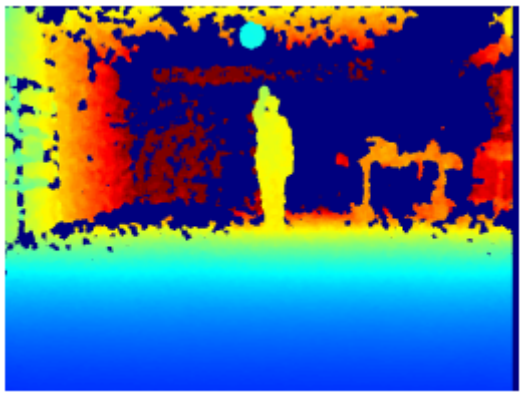
A análise de superfícies tridimensionais em alta qualidade pode ser considerada uma atividade dispendiosa uma vez que para a aquisição de dados 3D de alta qualidade são utilizados equipamentos e montagens sofisticados que, por consequência, têm custos elevados e aplicação limitada (JOSÉ, 2008). No entanto, com a evolução das técnicas de reconhecimento de imagem, houve o surgimento das câmeras RGB-D, como o Kinect (Microsoft) ou o Xtion PRO Live (Asus), os quais acabaram se popularizando devido ao fato de seus sensores serem de baixo custo e adquirirem os dados a uma frequência elevada contendo informação de profundidade e imagem.

As câmeras RGB-D incorporam na sua gênese o processamento de nuvens de pontos adquiridos pelos próprios dispositivos. Devido a esse fato, as características dos dados capturados pelo Kinect têm atraído a atenção de pesquisadores de outros campos, incluindo mapeamento e modelagem 3D (KHOSHELHAM & ELBERINK, 2012). Considerando a aplicação de técnicas e algoritmos no processamento de nuvens de pontos a partir da utilização de sensores de profundidade de baixo custo para o registro tridimensional, foram encontrados relevantes trabalhos na literatura, os quais são discutidos a seguir.

Duarte (2015) propôs a realização do reconhecimento de uma bola num jogo de futebol de robôs em dois diferentes casos, bolas aéreas e não-aéreas. Após a realização de um levantamento bibliográfico dos métodos de detecção e reconhecimento de objetos, o autor adotou o método *K-Means Clustering* para o caso aéreo e segmentação do campo baseado em cor para o caso não-aéreo. No primeiro caso, o algoritmo realiza a quantificação de um vetor para separar todas as observações e, posteriormente, agrupá-las num número definido de *cluster* onde um ponto pertence a um dado grupo de pontos minimizando a soma dos quadrados (Figura 2). No segundo caso, a partir da utilização de um espaço de cores YUV (Y - transportam a quantidade de luminosidade da imagem / UV - transportam informação da cor), é realizada a extração dos pontos que pertencem ao plano do chão através do método RANSAC e o método de *clustering* para a remoção do plano do chão (Figura 3). Após a segmentação dos objetos, foi aplicado um algoritmo de reconhecimento para realizar a análise das propriedades geométricas da bola de

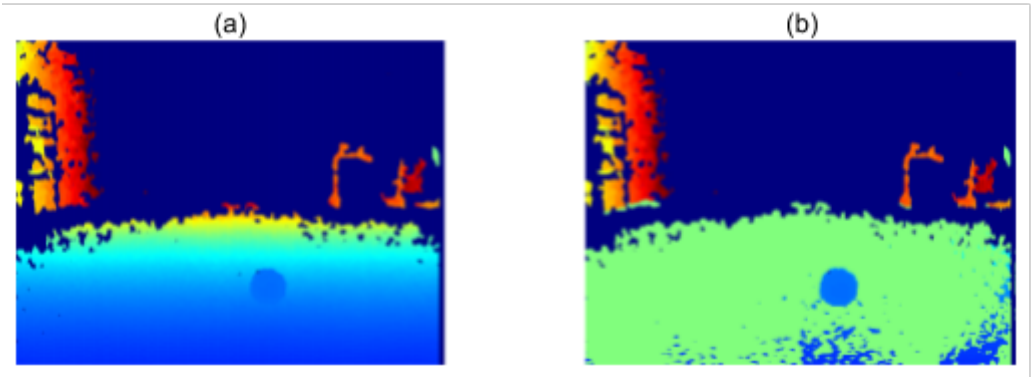
futebol, impondo ao sistema as seguintes condições: a área, perímetro, circularidade e o raio.

Figura 2 - Imagem de profundidade com bola aérea.



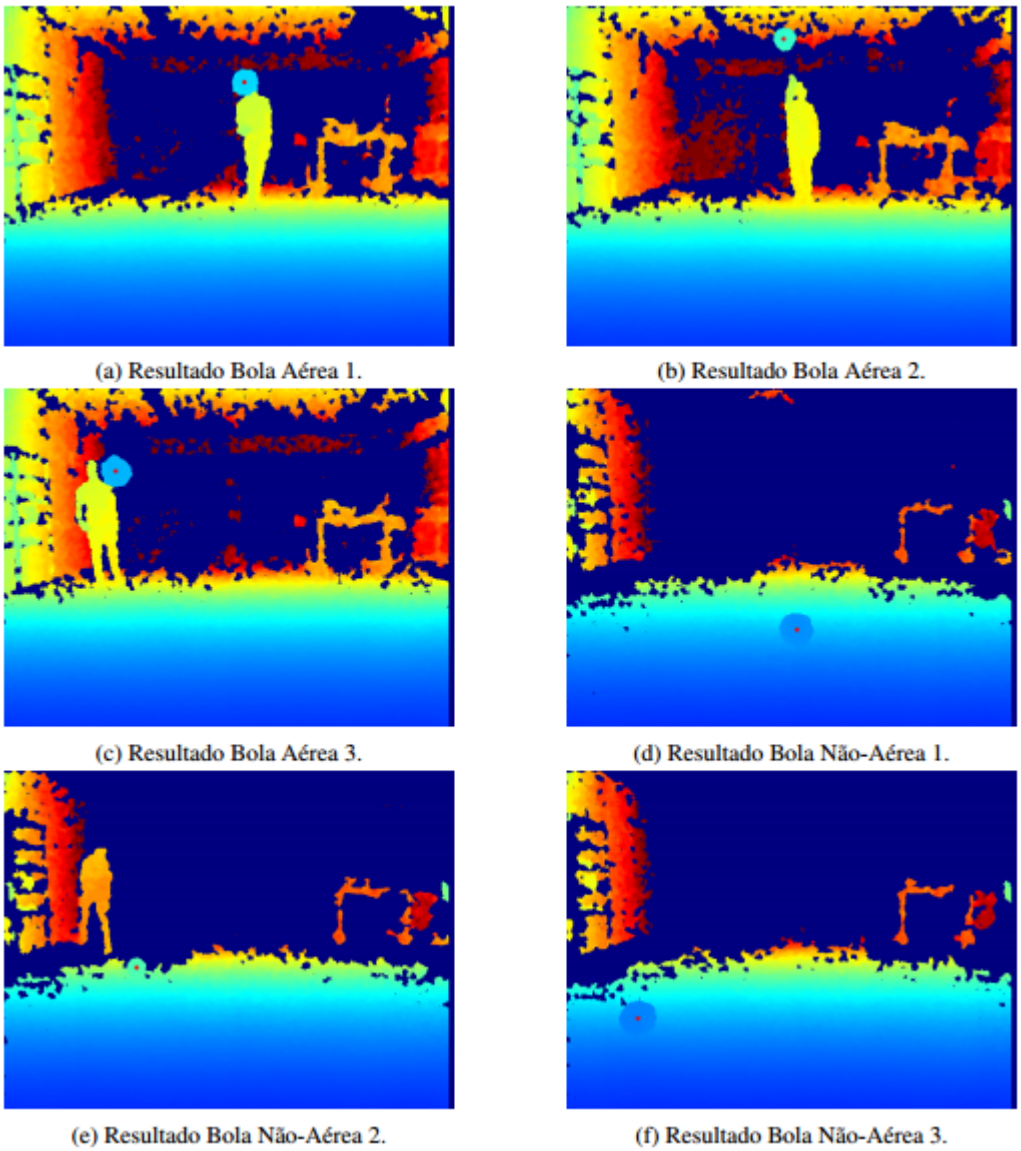
Fonte: Duarte (2015)

Figura 3 - Segmentação do plano do chão.
(a) Imagem de profundidade original. (b) Imagem com plano do chão a verde



Fonte: Duarte (2015)

Figura 4 - Resultado do Algoritmo de Reconhecimento



(a) Resultado Bola Aérea 1. (b) Resultado Bola Aérea 2.

(c) Resultado Bola Aérea 3. (d) Resultado Bola Não-Aérea 1.

(e) Resultado Bola Não-Aérea 2. (f) Resultado Bola Não-Aérea 3.

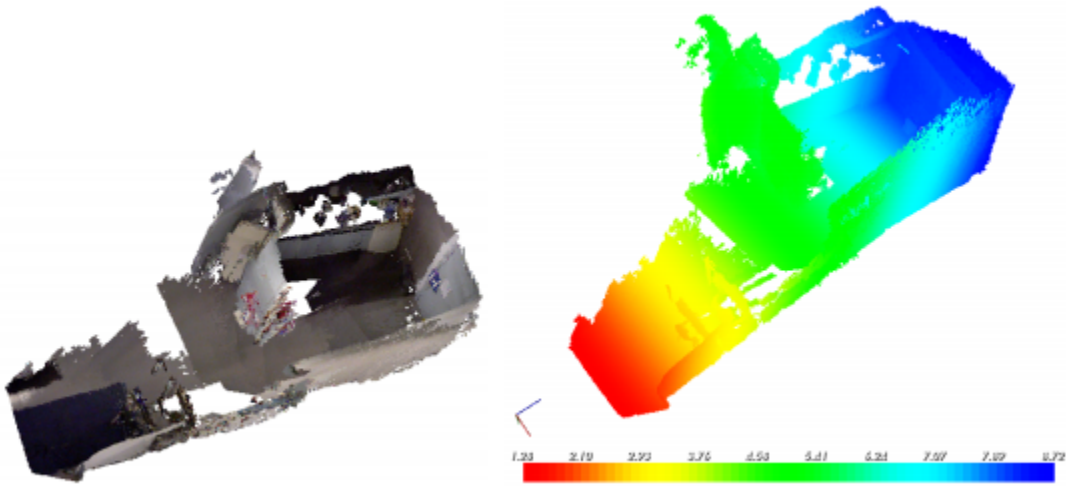
Fonte: Duarte (2015)

Os resultados atingidos no trabalho descrito podem ser considerados satisfatórios, uma vez que a análise das características da bola de futebol e o reconhecimento foram realizados em tempo total de processamento suficiente para o acompanhamento do movimento da bola. Além disso, foi aplicado a técnica denominada *Filtro de Kalman* para se estimar a posição da bola em cada quadro (*frame*). A análise da descrição do projeto desenvolvido permite perceber a importância do processo de segmentação de objetos para a detecção da bola. No entanto, apesar da eficiência descrita, não há

evidências do desempenho do mesmo em cenários de bola não-aérea e com múltiplos objetos.

Estudos relacionados ao ramo da visão computacional propõem a utilização de ferramentas computacionais para o registro e mapeamento de superfícies, como por exemplo a biblioteca PCL (*Point Cloud Library*), de modo a viabilizar o fornecimento de blocos de construção comuns necessários para o processamento das nuvens de ponto de uma aplicação tridimensional. No trabalho de Queirós (2012), tem-se a avaliação da utilização da biblioteca PCL para a construção de mapas multidimensionais e para a interpretação das nuvens de pontos registradas em ambientes internos. A partir da análise dos módulos fornecidos pela biblioteca PCL, desenvolveu-se uma arquitetura para permitir a obtenção de um mapa global consistente para cenários internos não estruturados. Apesar da arquitetura desenvolvida ter apresentado resultados qualitativos satisfatórios, não houve a implementação do mesmo para a avaliação de reconhecimento e classificação dos objetos existentes nos ambientes internos.

Figura 5 - Resultante do mapeamento de um dos cenários mapeados utilizando arquitetura elaborada pelo autor.

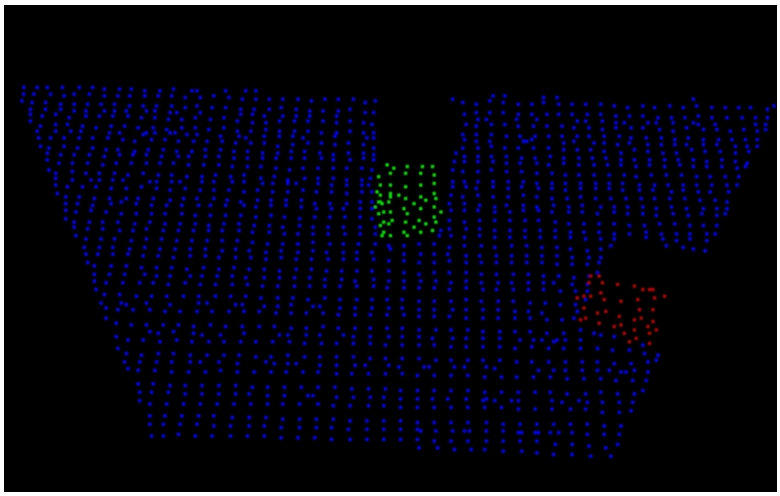


Fonte: Queirós (2012)

Semelhante ao trabalho anterior, Nguyen (2012) também utiliza o sensor de profundidade de baixo custo, o Kinect (Microsoft), associado à biblioteca PCL com a finalidade de detectar obstáculos. O autor propôs um método para a detecção de

obstáculos por um robô através do uso do Kinect com o suporte da biblioteca PCL. Para análise dos objetos foi empregada a segmentação do tipo Euclidiana para a extração e classificação dos *clusters* por cor (Figura 6). Após o emprego dessa técnica, foi possível se obter as dimensões dos obstáculos detectados. Assim, o robô conseguiu extrair com sucesso informações precisas do ambiente e dos obstáculos para a realização do movimento plano. A análise do modelo desenvolvido permite perceber a relevância do emprego de métodos de segmentação adequados para a classificação e separação dos *cluster*. Apesar disso, considera-se que uma comparação entre técnicas de segmentação para a classificação dos obstáculos comprovaria sua viabilidade.

Figura 6 - Nuvem de Pontos filtrada e segmentada pelo método Euclidiano



Fonte: Nguyen (2012)

A análise dos trabalhos descritos anteriormente, permite a percepção da importância da implementação de métodos de segmentação para a classificação de objetos e superfícies de maneira eficiente em diversas aplicações.

4 REFERENCIAL TEÓRICO

4.1 Câmeras RGBD

Segundo Sousa (2015), câmeras RGBD são câmeras que apresentam recursos de cor e profundidade. Uma câmera RGBD combina dados da imagem para gerar a representação em uma nuvem de pontos. Dentre as câmeras RGBD mais conhecidas, o Kinect, desenvolvido pela empresa Microsoft, se popularizou devido a seu baixo custo e resultados satisfatórios na representação de superfícies (XU, 2014).

As principais peças que compõem o sensor são: 2 sensores de profundidade (emissor e sensor de profundidade infravermelho); sensor de cores (câmera RGB); uma rede de microfones; motor de movimento (MICROSOFT, 2017). A imagem abaixo ilustra os componentes do sensor Kinect:

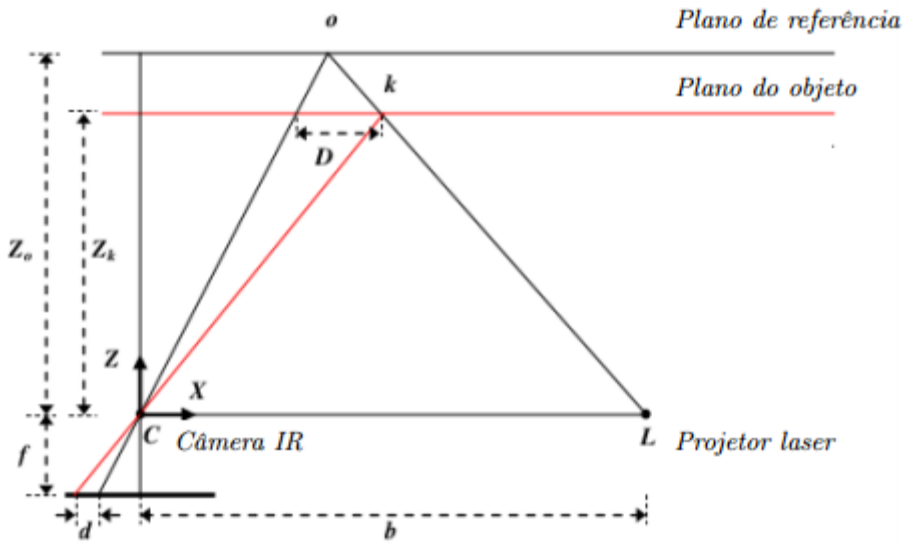
Figura 7 - Componentes do sensor Kinect

Fonte: Martins (2014)

i. **Sensores de profundidade 3D (Câmera IR e Projetor IR):** Composto por um emissor de luz estruturada IR e a câmera IR (infravermelho), o sistema de detecção de profundidade (*depth*) consiste na emissão de um padrão estruturado de luz infravermelha e na leitura da reflexão dos raios pela câmara (CORREIA, 2013). A câmera IR realiza a interpretação da deformação da

projeção e conversão da informação em valores de profundidade através da triangulação do padrão conhecido de pontos e o observado (Figura 8).

Figura 8 - Sistema de detecção de profundidade do sensor Kinect



Fonte: Martins (2014)

ii. Câmera RGB: A câmera RGB (vermelho, verde, azul) auxilia na identificação do indivíduo. A câmera apresenta resolução VGA de 8 bits com uma taxa de quadros de filtragem de cores de 30 Hz (NGUYEN, 2012).

iii. Vários microfones: Uma série de microfones na parte dianteira inferior do sensor Kinect é usada para reconhecimento de voz e bate-papo (MICROSOFT, 2017).

iv. Inclinação mecanizada: Unidade mecânica na base do sensor Kinect inclina a cabeça do sensor automaticamente para cima e para baixo em no máximo com uma amplitude de 27º (NGUYEN, 2012).

4.2 Nuvem de Pontos

Uma nuvem de pontos é uma estrutura de dados estruturados formados por uma coleção de pontos multidimensionais invariantes e é comumente usada para representar dados tridimensionais (PCL, 2011).

Na equação (1) tem-se a representação de uma nuvem de pontos Pcom um conjunto de p_i, os quais são usados para representar as informações tridimensionais de nD pontos (NGUYEN, 2012).

$$P = \{p_1, p_2, \ldots, p_i, \ldots, p_n\}, p_i = \{x_i, y_i, z_i\} \qquad (1)$$

A equação (2) representa os casos em que a informação da cor está presente e, consequentemente, a nuvem de pontos torna-se 4D.

$$N = \{q_1, q_2, \ldots, q_m\}, q_i = \{x_i, y_i, z_i, (R_i, G_{i,} B_i)\} \qquad (2)$$

Fonte: Sousa (2012) - Adaptado pelo Autor

No cálculo de mapeamento de superfícies, os pontos de nuvem são gerados pela câmera e são adotados para a extração de dados relevantes referentes a profundidade da região analisada. De acordo com Machado (2013), o Kinect funciona através do trabalho em conjunto do emissor e do sensor IR. Feixes de luz infravermelho são emitidos e refletidos de volta para o sensor que realiza a leitura dos mesmos, convertendo os dados em informações de profundidade. Assim, é feita uma comparação entre nuvem de pontos previamente conhecida e a nuvem de pontos extraída do ambiente permitindo a detecção de qualquer perturbação dentro das posições do padrão conhecido.

4.3 PCL (Point Cloud Library)

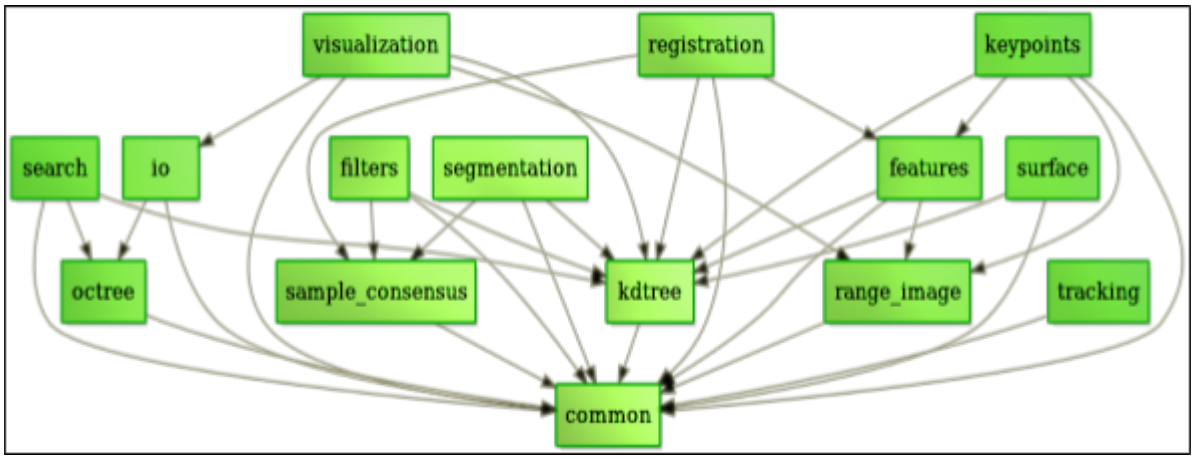
Considerada uma das mais recentes ferramentas na área de percepção 3D, a biblioteca *Point Cloud Library*[1] (PCL) engloba todos os principais algoritmos utilizados no processamento de nuvens de ponto (RUSU & COUSINS, 2011).

[1] http://pointclouds.org/

Desenvolvida por cientistas e pesquisadores de diversas organizações, a biblioteca PCL é um software de código aberto em linguagem C++, gratuita tanto para fins comerciais quanto acadêmicos.

Compatível com os principais sistemas operacionais (Linux, OSX, Windows e Android/iOS), PCL contém uma série de módulos pequenos, os quais podem ser compilados separadamente. As relações de operações apresentadas na Figura 9 representam a forma como os módulos de PCL se relacionam com as nuvens de pontos processadas:

Figura 9 - Módulos da biblioteca PCL



Fonte: PCL (2011)

O trabalho com nuvem de pontos permite realizar o mapeamento da cena tridimensional capturada e utilizá-lo como referência para a realização de diversas atividades, tais como, medição de distâncias e navegação de sistemas. Para se conseguir extrair informações essenciais do mapa da cena, é essencial que as etapas de pré-processamento e processamento sejam realizadas.

4.3.1 Pré-Processamento

O pré-processamento consiste no estágio de preparação e reconstrução da nuvem de pontos, encarregando-se de atividades essenciais para a manipulação das nuvens, como erros e remoção de ruídos através do uso de filtros, suavização e redução da contagem de pontos, mantendo as informações de características relevantes (GRUPO DE ROBÓTICA, 2015). Conforme o diagrama apresentado a seguir, o pré-processamento é composto pelas etapas a seguir:

Figura 10 - Etapas de pré-processamento de nuvem de pontos realizado pela biblioteca PCL

Fonte: Adaptado pelo próprio autor

I. **Nuvem de Pontos (*Point clouds*):** Os sensores de profundidade retornam um conjunto de pontos tridimensionais, nuvem de pontos, a partir do princípio da luz estruturada. A biblioteca PCL engloba nessa etapa: (1) geração da nuvem de pontos; (2) carregamento e visualização da nuvem; (3) concatenação e transformações matriciais; (4) remoção de *NaN* (*Not a Number*) - valores imprecisos devido a erros na medida da distância realizada pelo sensor, cálculo das coordenadas do centróide da nuvem de pontos (GRUPO DE ROBÓTICA, 2015).

II. **Estimativa de Características (*Feature Estimation)*:** Extração de características mais específicas de um ponto, por exemplo, como as normais, que permitam diferenciá-lo de um ponto diferente (GRUPO DE ROBÓTICA, 2015).

III. **Decomposição de Dados (*Decomposition*):** Corresponde a criação de uma estrutura de dados a partir dos pontos de nuvem de modo organizado e ordenado. Essa etapa é composta pelos seguintes métodos de decomposição:

- ***KdTree* (ou árvore *k*-dimensional)**: é uma estrutura de dados usada na ciência da computação para organizar um certo número de pontos em um espaço com *k* dimensões (PCL, 2011).

- **Octree**: é uma estrutura de dados de árvore hierárquica úteis para pesquisas, mas também compressão ou redução da resolução (GRUPO DE ROBÓTICA, 2015).

IV. **Filtro de Dados (*Filtering)*:** Corresponde ao emprego de métodos de polimento em nuvens de pontos no intuito de corrigir erros recorrentes nos dados retornados pelo sensor (ruídos, buracos nas superfícies, medidas incorretas, entre outros). Essa etapa constitui-se dos seguintes métodos de filtragem:

- ***PassThrough:*** Remove qualquer nuvem de pontos cujos valores não estão dentro de um determinado intervalo informado pelo usuário, permitindo, assim, que objetos desnecessários da nuvem sejam descartados (GRUPO DE ROBÓTICA, 2015).

- ***Conditional Removal:*** Método que permite a construção condicional para os valores da nuvem de pontos de modo a se obter uma filtragem similar ao método *PassThrough* (GRUPO DE ROBÓTICA, 2015).

- ***Outlier Removal:*** Sensores podem registrar medições onde não há necessidade de se haver nenhuma. Devido a isso, geram-se pontos solitários espalhados na nuvem, conhecidos como ruídos (*outliers*), os quais podem introduzir erros nos cálculos (GRUPO DE ROBÓTICA, 2015). Para removê-los pode-se determinar um raio de pesquisa e o número mínimo de vizinhos de um ponto para ser considerado um ruído (*radius-based*) ou através da análise estatística das distâncias entre pontos vizinhos (*statistical*).

- ***Downsamplig:*** permite reduzir o número de pontos da nuvem utilizando o processo de *Voxel-grid.*

- ***Upsampling:*** Permite a interpolação dos pontos da nuvem já conhecidos, sendo aplicável para quando se tem menos pontos do que o necessário.

V. **Reconstrução:** Consiste na utilização do algoritmo de suavização MLS (*Moving Last Squares*), o qual corresponde a um método de reconstrução de superfície utilizado para a suavização e reamostragem de dados ruidosos (GRUPO DE ROBÓTICA, 2015).

4.3.2 Processamento

O processamento corresponde as técnicas que podem ser empregadas as nuvens de ponto pré-processadas de modo a permitir a extração/análise avançada dos dados (GRUPO DE ROBÓTICA, 2015). Conforme diagrama apresentado a seguir, o processamento é composto pelas etapas a seguir:

Figura 11 - Etapas de processamento de nuvem de pontos realizado pela biblioteca PCL

Fonte: Adaptado pelo próprio autor

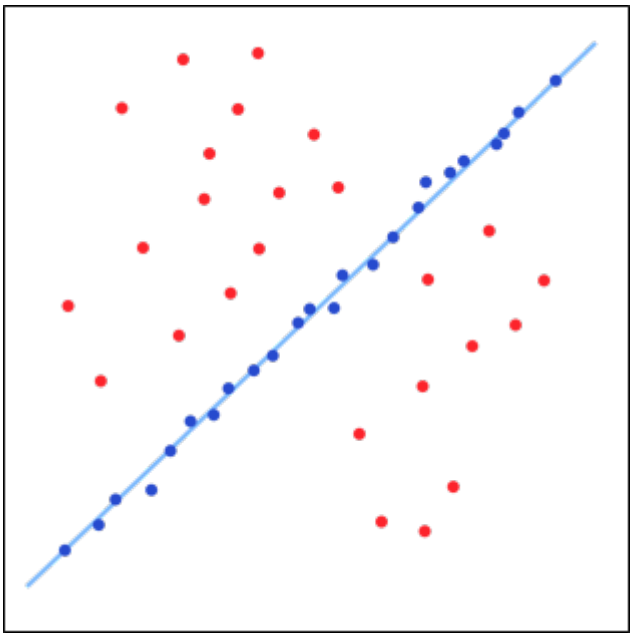
I. **Registro:** Aplicação de um algoritmo para realizar o alinhamento de duas nuvens pontuais. O algoritmo deve encontrar um conjunto de correspondências através da aplicação de transformações lineares nas nuvens. Essa técnica é muito útil, pois, permite a recuperação de modelos completos e contínuos de uma cena ou objeto. Os algoritmos mais conhecidos para se realizar transformações são:

A. ***ICP (Iterative Closest Point):*** Algoritmo que, através da interação entre as várias correspondências de pontos de interesse, procura encontrar a melhor transformação para minimizar a distância entre dois conjuntos de

pontos gerados através de digitalizações sucessivas (ALEXANDRA, 2013).

B. RANSAC (RANdom SAmple Consensus): Algoritmo não determinístico que melhora o resultado das nuvens de ponto a cada iteração (GRUPO DE ROBÓTICA, 2015). Segundo Borges (2013), um conjunto de pontos é selecionado de forma aleatória e, após ter os pontos é definido um modelo, é criada uma linha entre os vários pontos contidos numa dada área, possibilitando a identificação dos pontos que estão dentro da área (*inliers*) e os que estão fora (*outliers*), exemplificados na Figura 12.

Figura 12 - Conjunto de pontos *inliers* e *outliers* após a aplicação do algoritmo RANSAC

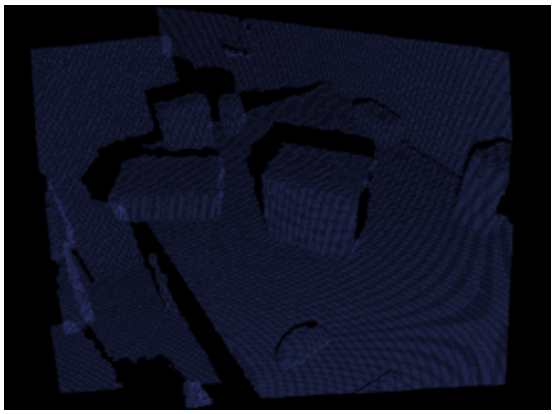
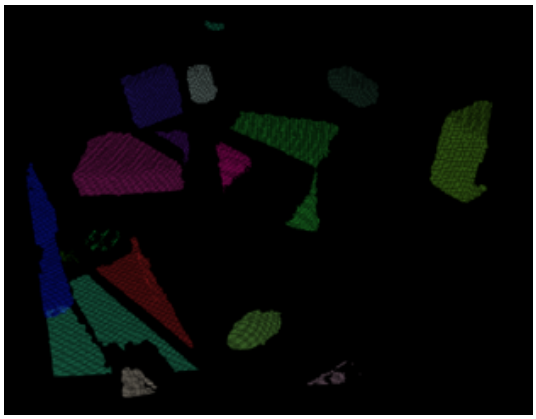


Fonte: GRUPO DE ROBÓTICA (2015)

II. Segmentação: Consiste na divisão da nuvem em diferentes seções ou grupos de pontos chamados *clusters* no intuito de permitir que a nuvem seja processada de forma independente (GRUPO DE ROBÓTICA, 2015). Apesar da existência de diversas técnicas de segmentação englobadas pela biblioteca PCL, no presente trabalho serão destacados apenas os métodos abaixo:

A. **Segmentação Euclidiana (*Euclidean Segmentation*):** A ideia dessa segmentação assemelha-se ao algoritmo de preenchimento de inundação, pois, um ponto na nuvem é "marcado" como "escolhido" para *cluster*, espalhando-se para todos os pontos próximos. Após, inicia-se novamente o mesmo processo por um ponto que não foi atingido.

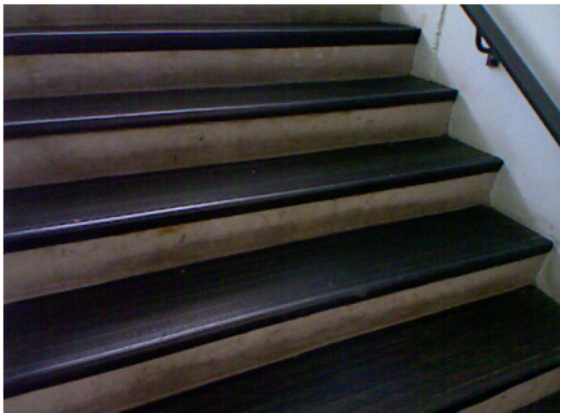
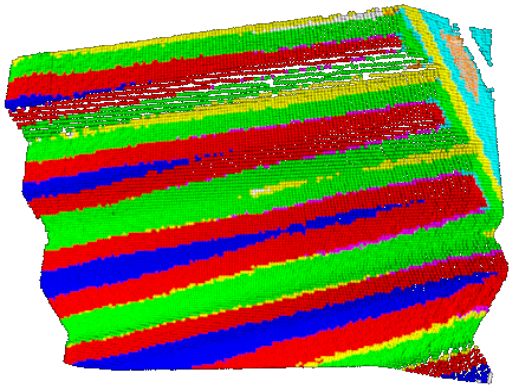
Figura 13 - Visualização dos Clusters gerados após aplicação de segmentação Euclidiana

Nuvem de Pontos Original	*Clusters* segmentação Euclidiana
	

Fonte: GRUPO DE ROBÓTICA (2015)

B. **Crescimento de Região (*Region Growing*):** A proposta desse algoritmo é agrupar pontos próximos em termos de restrição de suavidade de modo a considerá-los do mesmo plano (PCL, 2011). Assim, através da comparação entre as normais e as curvaturas dos pontos, um conjunto de pontos será considerado como parte da mesma superfície lisa.

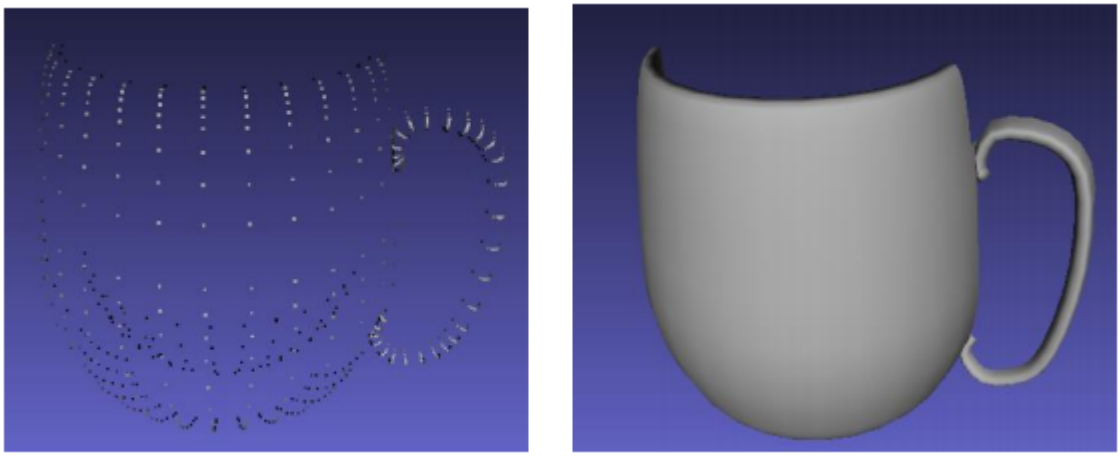
Figura 14 - Visualização dos Clusters gerados após aplicação de segmentação *Region Growing*

Nuvem de Pontos Original	*Clusters* segmentação *Region Growing*
	

Fonte: GRUPO DE ROBÓTICA (2015)

III. **Reconstrução:** Para permitir a reconstrução de uma superfície suave com eliminação de dados ruidosos, a biblioteca PCL apresenta recursos que permitem a estimativa aproximada da superfície capturada pela nuvem de pontos (GRUPO DE ROBÓTICA, 2015). Dentre esses métodos, o algoritmo de triangulação destaca-se por possibilitar a projeção da vizinhança local de um ponto ao longo do ponto normal e conectando pontos não conectados.

Figura 15 – Reconstrução de objeto registrado por método de triangulação



Fonte: Ozbay & Cinar (2013)

5 DESENVOLVIMENTO DOS MODELOS DE CLASSIFICAÇÃO

Nessa seção tem-se na elaboração de dois modelos de classificação, adotando respectivamente os métodos de segmentação *Euclidean* e *Region Growing*.

5.1 Modelo 1 - *Euclidean Segmentation*

No modelo desenvolvido para a implementação da segmentação do tipo *Euclidean Segmentation*, foram adotados os seguintes métodos nas etapas de pré-processamento e processamento da nuvem de pontos em tempo real:

5.1.1 Pré-processamento

- **Aquisição de mapa de profundidade em tempo real:**
  - **Openni2_grabber.h:** Biblioteca incorporada à PCL que permite a aquisição de dados *streams* (em tempo real) de câmeras compatíveis (*Primesense Reference Design*, *Microsoft Kinect* e *Asus Xtion Pro*).

Figura 16 – Pseudo-código *openni2_grabber*:

```
// create a new grabber for OpenNI devices
pcl::Grabber* interface = new pcl::OpenNIGrabber();

// make callback function from member function
boost::function<void (const pcl::PointCloud<pcl::PointXYZRGBA>::ConstPtr&)> f =
  boost::bind (&SimpleOpenNIProcessor::cloud_cb_, this, _1);

// connect callback function for desired signal. In this case its a point cloud with color values
boost::signals2::connection c = interface->registerCallback (f);

// start receiving point clouds
interface->start ();

// wait until user quits program with Ctrl-C, but no busy-waiting -> sleep (1);
while (true)
  boost::this_thread::sleep (boost::posix_time::seconds (1));

// stop the grabber
interface->stop ();
```

Fonte: elaborado pelo próprio autor

- **Filtro (*Filtering*):**
  - **Passthrough.h:** Cria-se um objeto de filtro PassThrough e define-se seus parâmetros. O nome do campo do filtro é definido como a coordenada z e os valores de intervalo

aceitos são definidos como (0,0; 1.3), a saída é computada e armazenada em *cloud_filtered*.

Figura 17 – Pseudo-código *PassThrough*:

```
//FILTERING PROCESS TO ELIMINATE POINTS
pcl::PassThrough<pcl::PointXYZRGBA> pass;
pass.setInputCloud(cloud);
pass.setFilterFieldName("z");
pass.setFilterLimits(0, 1.3);
pass.filter(*cloud_filtered);
```

Fonte: elaborado pelo próprio autor

- **Downsampling:**
  - **Voxel_Grid.h:** Cria-se um filtro pcl::VoxelGrid com tamanho de folha de 1cm, os dados de entrada são passados e a saída é computada e armazenada em *cloud_filtered2*.

Figura 18 – Pseudo-código *VoxelGrid*:

```
//DOWNSAMPLING RESULTING POINT CLOUD
pcl::VoxelGrid<pcl::PointXYZRGBA> sor;
sor.setInputCloud(cloud_filtered);
sor.setLeafSize(0.01f, 0.01f, 0.01f);
sor.filter(*cloud_filtered2);
```

Fonte: elaborado pelo próprio autor

5.1.2 Processamento

- **Registro:**
  - **SAC_Segmentation.h:** Cria-se uma estrutura da nuvem de pontos, para armazenar os coeficientes do modelo plano. Cria-se o objeto de segmentação e configura-o para o modelo de segmentação *RANSAC*, em conseguinte, aplica-se o referido método, indicando-se a distância máxima (*setDistanceThreshold*) e habilita-se o modelo de refinamento.

Figura 19 – Pseudo-código *RANSAC*

```
//SEGMENT SURFACE
pcl::ModelCoefficients::Ptr coefficients(new pcl::ModelCoefficients);
pcl::PointIndices::Ptr inliers(new pcl::PointIndices);
// Create the segmentation object
pcl::SACSegmentation<pcl::PointXYZRGBA> seg;
seg.setOptimizeCoefficients(true);
seg.setModelType(pcl::SACMODEL_PLANE);
seg.setMethodType(pcl::SAC_RANSAC);
seg.setDistanceThreshold(0.01);
```

Fonte: elaborado pelo próprio autor

- **Decomposição de dados:**
  - **Kdtree.h:** Cria-se um objeto *Kd-Tree* para o método de busca do algoritmo de extração. Posteriormente, cria-se um vetor de *PointIndices*, que contém os índices de cada cluster detectado salvos no vetor *cluster_indices* que contém uma instância de *PointIndices* para cada cluster detectado.

Figura 20 – Pseudo-código *KdTree*

```
//Creating the KdTree object for the search method of the extraction
pcl::search::KdTree<pcl::PointXYZRGBA>::Ptr tree(new pcl::search::KdTree<pcl::PointXYZRGBA>);
tree->setInputCloud(cloud_filtered2);
std::vector<pcl::PointIndices> cluster_indices;
```

Fonte: elaborado pelo próprio autor

- **Segmentação:**
  - **Extract_Clusters.h:** Cria-se um objeto do tipo *EuclideanClusterExtraction* com pontos do tipo *PointXYZRGBA*. Determina-se a tolerância de tamanho para cada *cluster* (*SetClusterTolerance*) determinando o número máximo e mínimo de pontos (*SetMaxClusterSize, SetMinClusterSize).* Aplica-se o método de busca (*kdtree*), na nuvem *cloud_filtered2*, e extrai-se os índices que são salvos no objeto *cluster_indices*. Na Figura 21 têm-se representado um modelo de definição de parâmetros e variáveis para a extração. Na Figura 22 têm-se o funcionamento do algoritmo para cada *cluster*.
  -

Figura 21 – Pseudo-código *Euclidean Clustering*

```
//Euclidean clustering
pcl::EuclideanClusterExtraction<pcl::PointXYZRGBA> ec;
ec.setClusterTolerance(0.05); // 2cm
ec.setMinClusterSize(100);
ec.setMaxClusterSize(25000);
ec.setSearchMethod(tree);
ec.setInputCloud(cloud_filtered2);
ec.extract(cluster_indices);
```

Fonte: elaborado pelo próprio autor

Figura 22 – Algoritmo do método *Euclidean Segmentation* para cada *cluster*

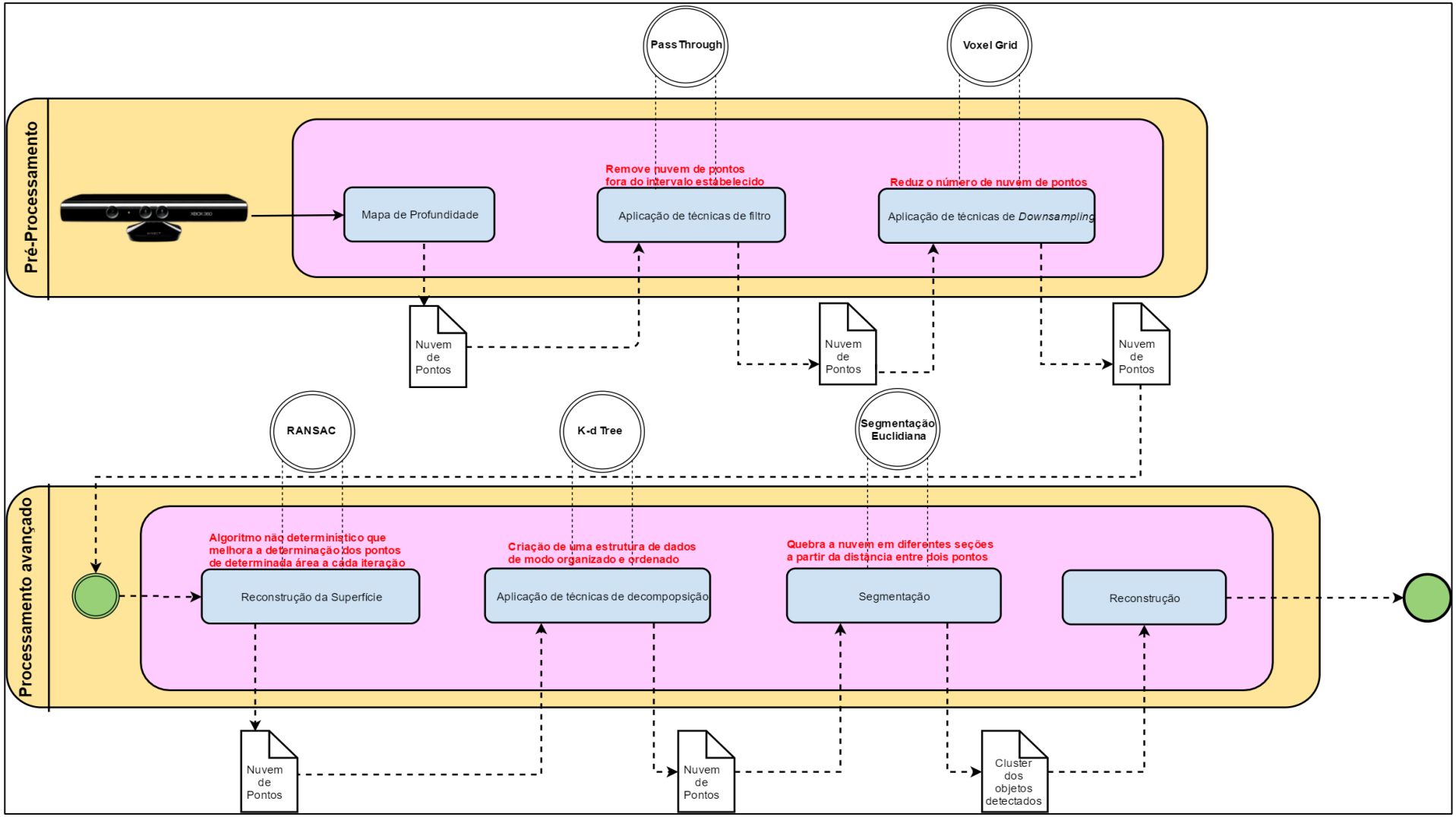
- P ⟶ Nuvem de Pontos resultante do método de pesquisa *KdTree*
- p_i ⟶ Pontos pertencentes à nuvem P
- C ⟶ Lista de clusters vazia
- Q ⟶ Fila de pontos a serem verificados pelo método
- d_{th} ⟶ Distância/Tamanho do Cluster

1. *Para cada ponto pertencente a P* $\boldsymbol{p}_i \in P$, realiza-se os seguintes passos:
   - *Adiciona o ponto* $\boldsymbol{p}_i$ a atual Fila Q;
   - *Para cada ponto pertencente a Fila,* $\boldsymbol{p}_i \in Q$, faz-se*:*
     - *Procura o conjunto* P_k^i de vizinhos de $\boldsymbol{p}_i$ em uma esfera de raio r menor que o definido pela tolerância, $r < d_{th}$;
     - *Para cada vizinho* $\boldsymbol{p}_i^k \in P_i^k$, checa-se se o ponto já foi processado, se não ele é adicionado à Q;
   - *Quando todos os pontos da lista* Q *tiverem sido processados, adicione* Q *à lista de clusters* C*, e resete* Q *para uma lista vazia novamente.*
2. *O algoritmo se encerra quando todos os pontos* $\boldsymbol{p}_i \in P$ *tiverem sido processados e se tornarem partes da lista de clusters* C.

Fonte: PCL (2011) - Adaptado pelo autor

Na figura abaixo tem-se a representação do modelo implementado pelo autor com a utilização do método *Euclidean Segmentation*:

Figura 23 – Modelo *Eulidean Segmentation*



Fonte: Elaborado pelo próprio autor

5.2 Modelo 2 - *Region Growing Segmentation*

No modelo desenvolvido para a implementação da segmentação do tipo *Region Growing Segmentation*, foram adotados os seguintes métodos nas etapas de pré-processamento e processamento da nuvem de pontos apresentados anteriormente nas seções 5.1.1 e 5.1.2, diferenciando-se apenas na etapa de Processamento.

5.2.1 Processamento

- **Segmentação:**
  - **Region_Growing.h:** Cria-se um objeto do tipo *RegionGrowing* com pontos do tipo *PointXYZ*. Determina-se o número máximo e mínimo de pontos para cada *cluster* (*SetMaxClusterSize, SetMinClusterSize).* Aplica-se o método de busca (*kdtree*), na nuvem *cloud*, e extrai-se as normais. Estabelece-se o número de vizinhos (*setNumberofNeighbours*). Define-se o ângulo limiar de suavidade em radianos (*SetSmoothnessThreshold*), que corresponde ao desvio máximo permitido das normais, e o limite de curvatura (*SetCurvatureThreshold*). Na Figura 24 têm-se representado um modelo de definição de parâmetros e variáveis para a extração. Na Figura 25 têm-se o funcionamento do algoritmo para cada *cluster*.

Figura 24 – Pseudo-código *Region Growing*

```
// Region growing clustering object.
pcl::RegionGrowing<pcl::PointXYZ, pcl::Normal> clustering;
clustering.setMinClusterSize(100);
clustering.setMaxClusterSize(10000);
clustering.setSearchMethod(kdtree);
clustering.setNumberOfNeighbours(30);
clustering.setInputCloud(cloud);
clustering.setInputNormals(normals);
clustering.setSmoothnessThreshold(7.0 / 180.0 * M_PI); // 7 degrees.
clustering.setCurvatureThreshold(1.0);
std::vector <pcl::PointIndices> clusters;
clustering.extract(clusters);
```

Fonte: elaborado pelo próprio autor

Figura 25 – Algoritmo do método *Region Growing* para cada *cluster*

Definição:

- *Nuvem de Pontos* = $\{P\}$
- *Pontos Normais* = $\{N\}$
- *Pontos de Curvatura* = $\{c\}$
- *Função para localizar pontos vizinhos* $\Omega(.)$
- *Limite de Curvatura* c_{th}
- *Ângulo limite* θ_{th}
- *Lista de Região* $R \leftarrow \emptyset$
- *Pontos disponíveis na lista* $\{A\} \leftarrow \{1, ..., |P|\}$

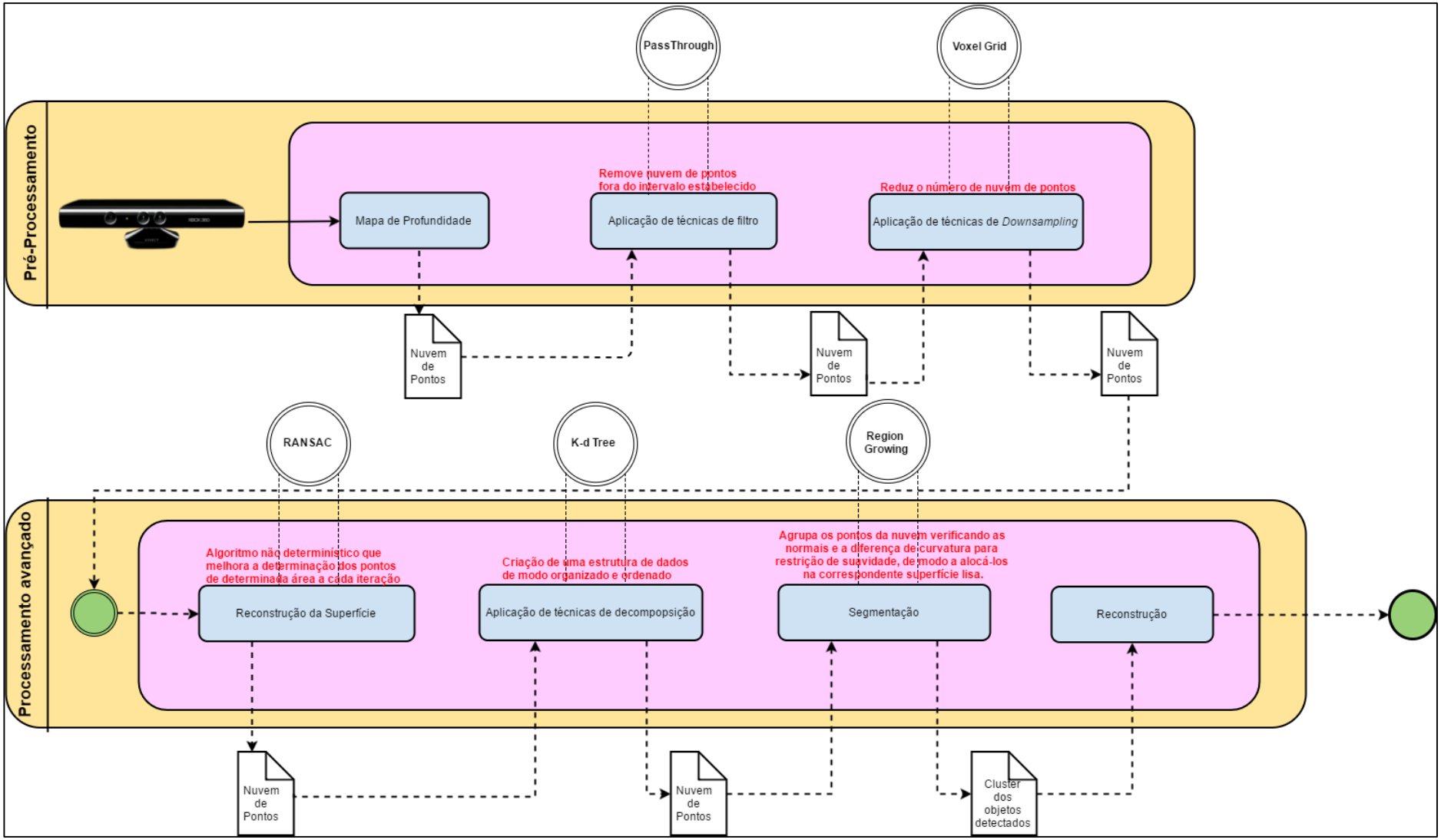
1. **Enquanto** $\{A\}$ não está vazio, faça:
    - *Região atual* $\{R_c\} \leftarrow \emptyset$
    - *Nuvem de pontos atual* $\{S_c\} \leftarrow \emptyset$
    - *Pontos com mínima curvatura em* $\{A\} \rightarrow P_{min}$
    - $\{S_c\} \leftarrow \{S_c\} \cup P_{min}$
    - $\{R_c\} \leftarrow \{R_c\} \cup P_{min}$
    - $\{A\} \leftarrow \{A\} \setminus P_{min}$
    - **Para** $i = 0$ *até o tamanho de* ($\{S_c\}$) **faça:**
        - *Encontra os vizinhos mais próximos da nuvem atual* $\{B_c\} \leftarrow \Omega(S_c\{i\})$
        - **Para** $j = 0$ *até o tamanho de* ($\{B_c\}$) **faça:**
            - *Pontos vizinhos atuais* $P_j \leftarrow B_c\{j\}$
            - **Se** $\{A\}$ *contém* P_j *e* $cos^{-1}(|(N\{S_c\{i\}\}, N\{S_c\{j\}\})|) < \theta_{th}$ **então**
                - $\{R_c\} \leftarrow \{R_c\} \cup P_j$
                - $\{A\} \leftarrow \{A\} \setminus P_j$
                - **Se** $c\{P_j\} < c_{th}$ então
                    - $\{S_c\} \leftarrow \{S_c\} \cup P_j$
                - **Fim do laço Se**
            - **Fim do laço Se**
        - **Fim do laço Para**
    - **Fim do laço Para**
    - *Adiciona a região atual à lista de segmentos globais* $\{R\} \leftarrow \{R\} \cup \{R_c\}$

Fim do laço Enquanto

Retorna $\{R\}$

Fonte: PCL (2011) - Adaptado pelo autor

Figura 26 – Modelo *Region Growing Segmentation*



Fonte: Elaborado pelo próprio autor

6 ESTUDO DE CASO – DETECÇÃO E CLASSIFICAÇÃO DE OBSTÁCULOS

Durante a realização do estudo de caso, delimitou-se os parâmetros para a classificação dos obstáculos de acordo com a altura. Definiu-se uma cor específica para classificar cada um dos intervalos especificados de acordo com a tabela abaixo:

Tabela 2 – Informações para cada intervalo

	Intervalos da altura *h* (*m*)	**Cor estabelecida para o intervalo**
Intervalo 1	$0<h\leq0,20$	Verde
Intervalo 2	$0,20<h\leq0,50$	Amarelo
Intervalo 3	$h>0,50$	Vermelho

Fonte: elaborado pelo próprio autor

Além disso, foram estabelecidas as seguintes configurações para cada modelo, conforme as tabelas a seguir:

Tabela 3 – Configuração modelo 1 (*Euclidean Segmentation*)

Parâmetros da extração	**Valores estabelecidos**
Tolerância do *Cluster (Cluster Tolerance)*	0,02 (*2cm*)
Tamanho mínimo do *Cluster (Min Cluster Size)*	50
Tamanho máximo do *Cluster (Max Cluster Size)*	10000000

Fonte: elaborado pelo próprio autor

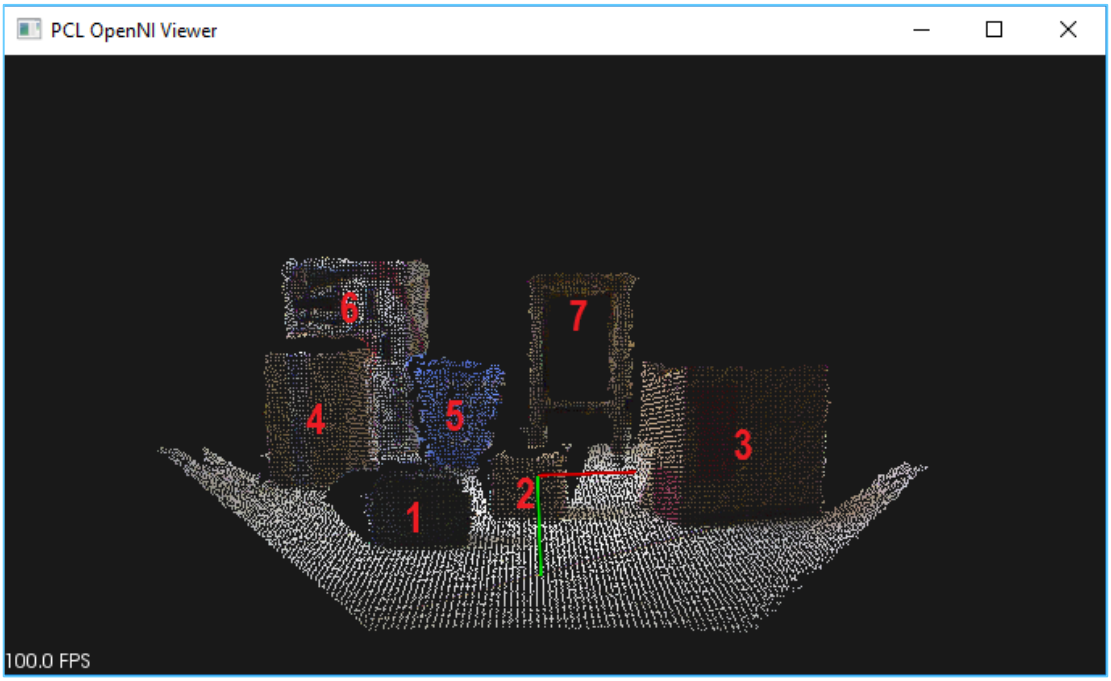
Tabela 4 – Configuração modelo 2 (*Region Growing*)

Parâmetros da extração	**Valores estabelecidos**
Tolerância do *Cluster (Cluster Tolerance)*	0,02 (*2cm*)
Tamanho mínimo do *Cluster (Min Cluster Size)*	50
Tamanho máximo do *Cluster (Max Cluster Size)*	1000000
Número de pontos vizinhos *(Number of Neighbours)*	30
Ângulo de suavidade *(Smoothness Threshold)*	3π rad e 7π rad
Limite de curvatura (*Curvature Threshold)*	1.0

Fonte: elaborado pelo próprio auto

Para a realização do estudo, foram utilizados 7 objetos para a detecção e classificação dos objetos de acordo com a altura *h* de cada um. Considerando que a superfície do chão foi adotada como referencial para o estudo de caso, aplicou-se a coloração azul no mesmo para facilitar a visualização do cenário. A figura abaixo representa os objetos utilizados e a tabela seguinte contém os dados referentes à altura dos mesmos:

Figura 27 – Objetos adotados para a implementação dos modelos de segmentação



Fonte: elaborado pelo próprio autor

Tabela 5 – Informações do Objeto

Objetos	**Altura *h* (*cm*)**	**Cor desejada para o objeto**
Objeto 1	13,1	Verde
Objeto 2	13,4	Verde
Objeto 3	30,3	Amarelo
Objeto 4	29,8	Amarelo
Objeto 5	24,2	Amarelo
Objeto 6	54,6	Vermelho
Objeto 7	53,0	Vermelho

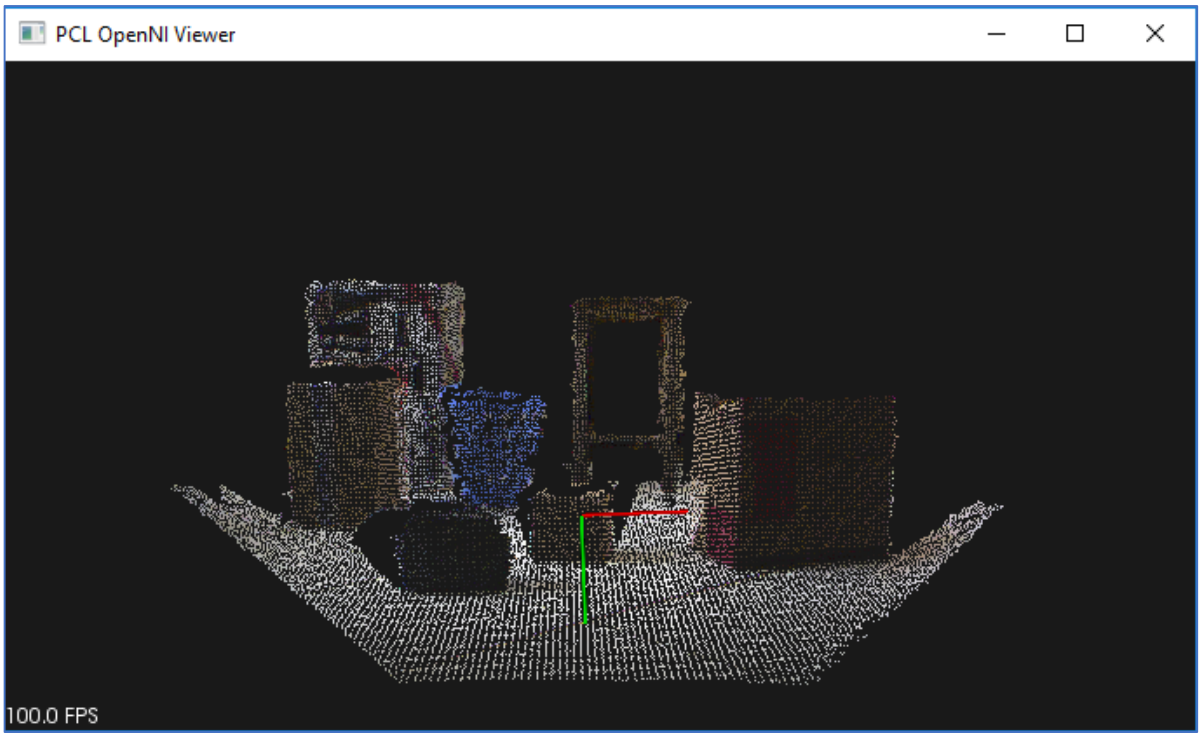
Fonte: elaborado pelo próprio autor

Foram estabelecidos 3 casos de teste, nos quais os objetos foram organizados em diferentes posições no cenário de modo a permitir a avaliação o desempenho da detecção e classificação de obstáculos de cada modelo.

6.1 Caso 1: Obstáculos distribuídos de forma dispersa

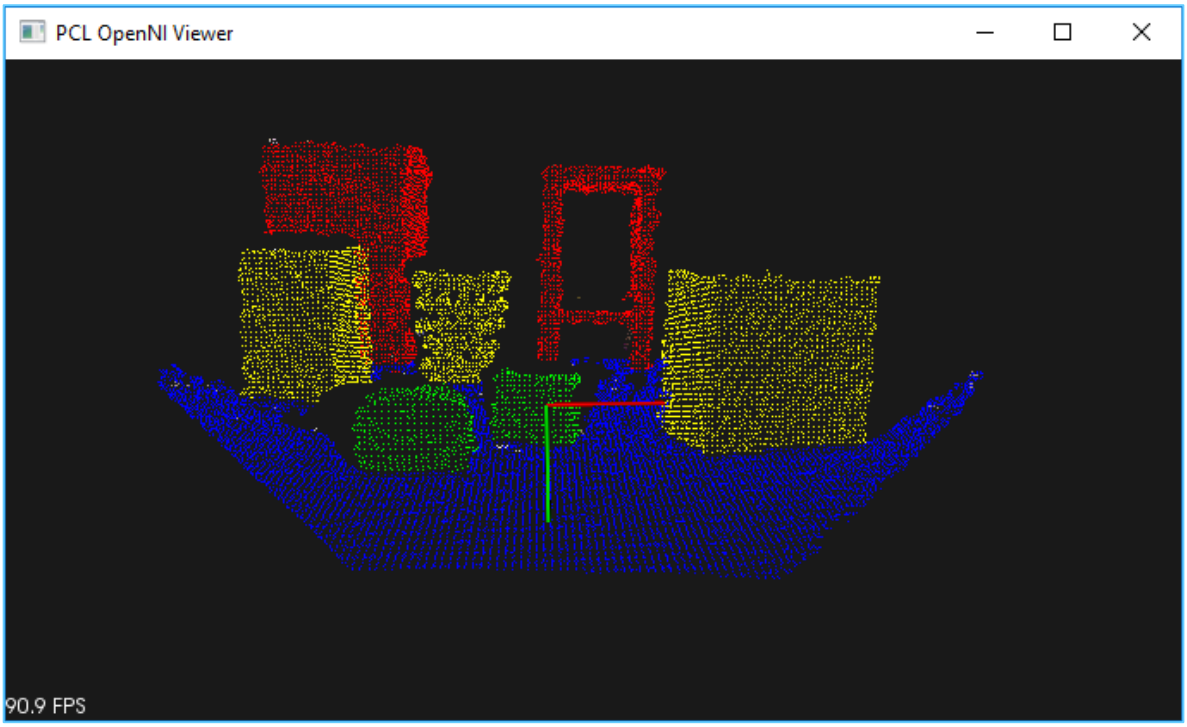
Para a elaboração do primeiro caso de teste, os objetos foram organizados de forma dispersa. As figuras abaixo representam o cenário original e os resultados obtidos para o modelo 1 e para o modelo 2 (adotando ângulos de suavidade 3π rad e 7π rad).

Figura 28 – Cenário original do caso 1



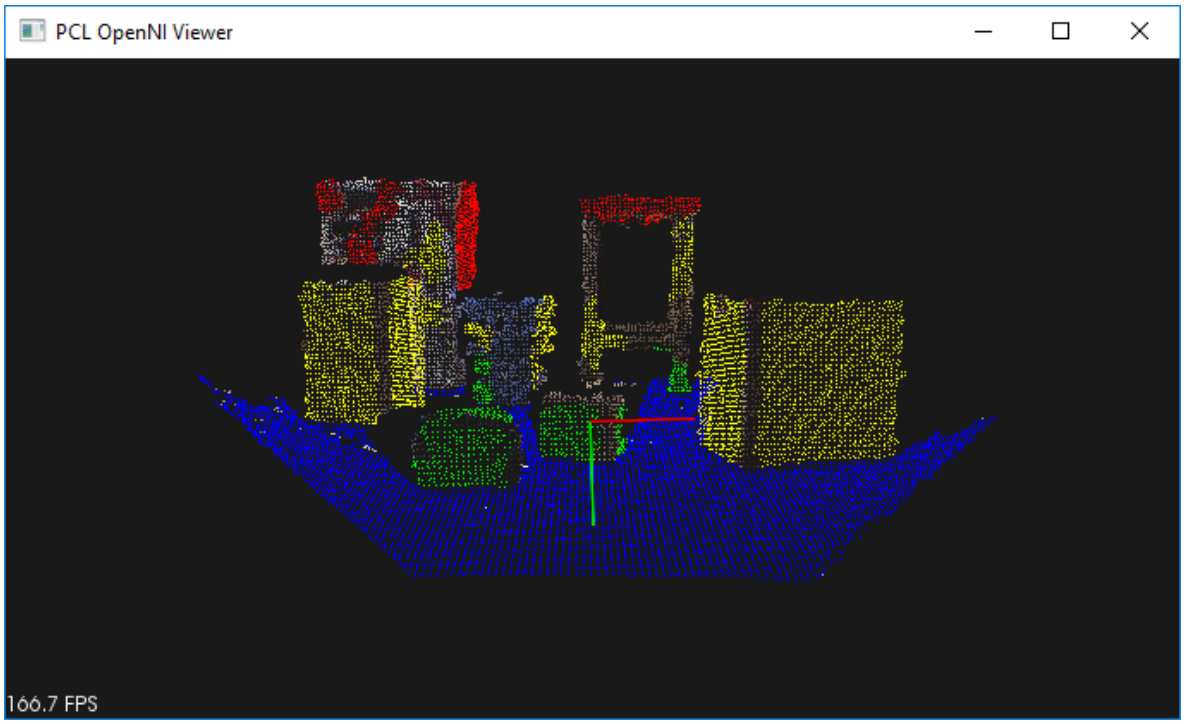
Fonte: elaborado pelo próprio autor

Figura 29 – Detecção de obstáculos dispersos no cenário - *Euclidean Segmentation*



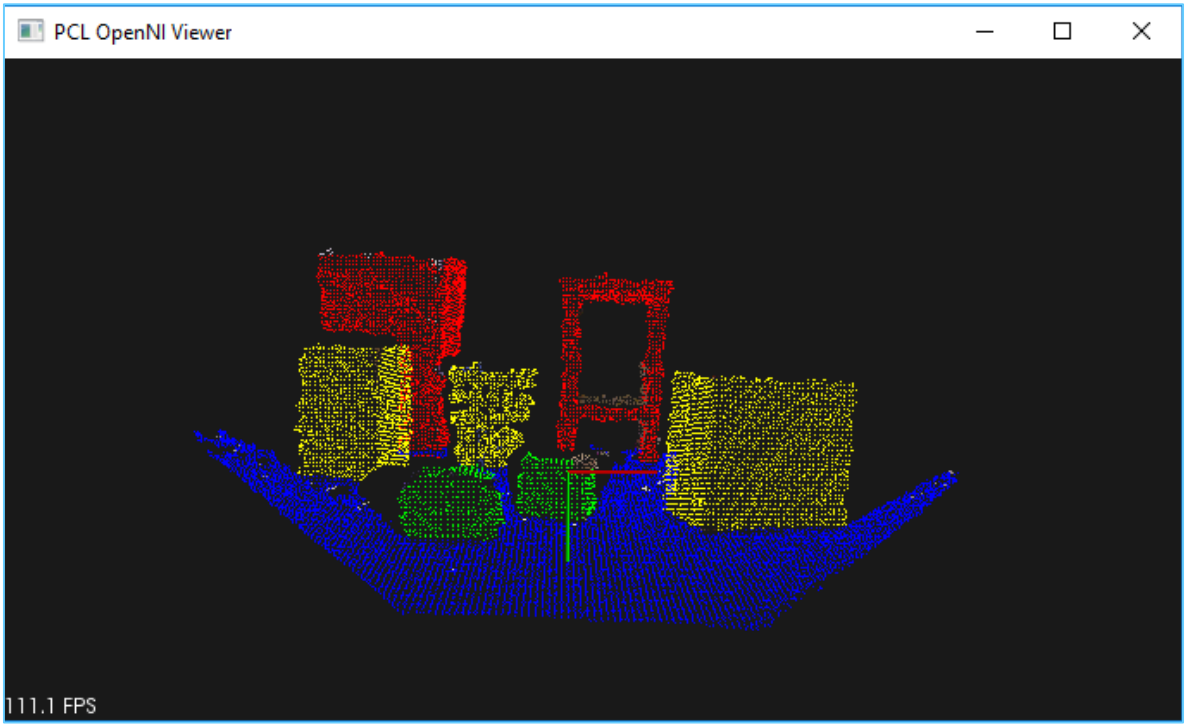
Fonte: elaborado pelo próprio autor

Figura 30 – Detecção de obstáculos dispersos no cenário – *Region Growing* (ângulo de suavidade de 3π rad)



Fonte: elaborado pelo próprio autor

Figura 31 – Detecção de obstáculos dispersos no cenário – *Region Growing* (ângulo de suavidade de 7π rad)



Fonte: elaborado pelo próprio autor

Tabela 6 - Caso 1: Detecção e Classificação de obstáculos dispersos

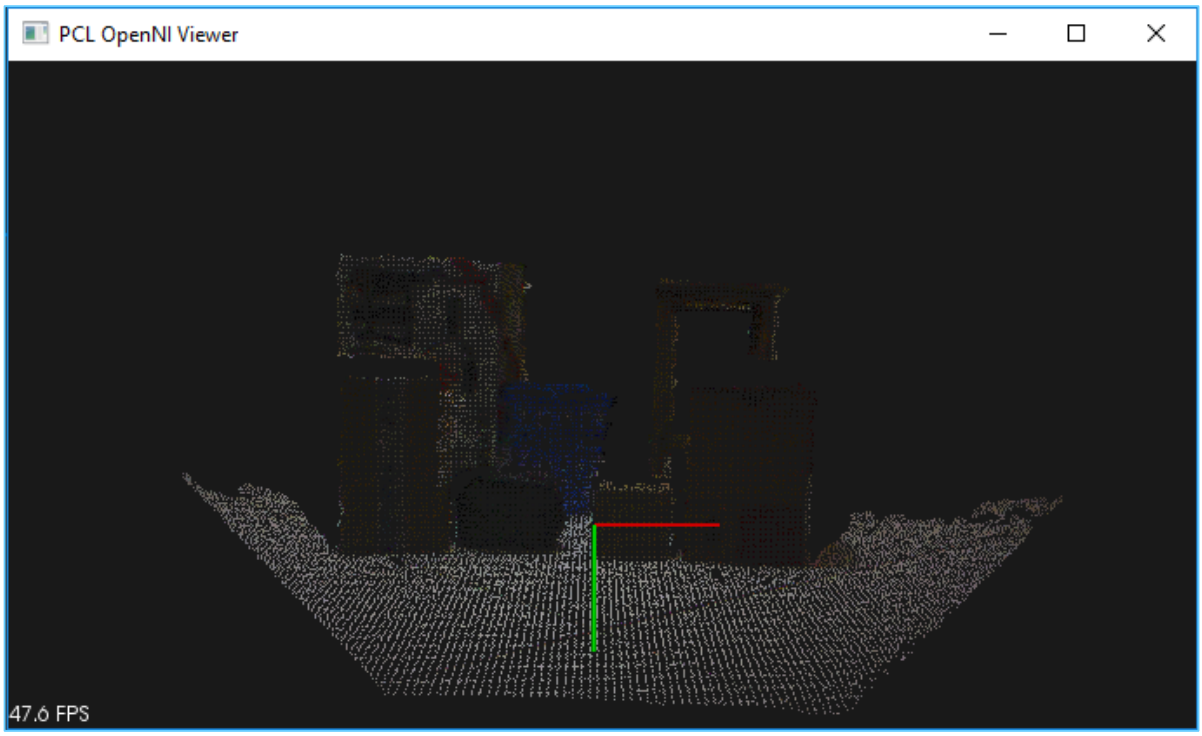
Obstáculo	**Resultado Final Desejado**	***Euclidean Segmentation***	**Region Growing (ângulo de suavidade 3º)**	**Region Growing (ângulo de suavidade 7º)**
Objeto 1	Verde	Verde	Parcialmente verde	Verde
Objeto 2	Verde	Verde	Parcialmente verde	Verde
Objeto 3	Amarelo	Amarelo	Parcialmente amarelo	Amarelo
Objeto 4	Amarelo	Amarelo	Parcialmente amarelo	Amarelo
Objeto 5	Amarelo	Amarelo	Erro	Amarelo
Objeto 6	Vermelho	Vermelho	Erro	Vermelho
Objeto 7	Vermelho	Vermelho	Erro	Vermelho

Fonte: elaborado pelo próprio autor

6.2 Caso 2: Obstáculos distribuídos de forma paralela

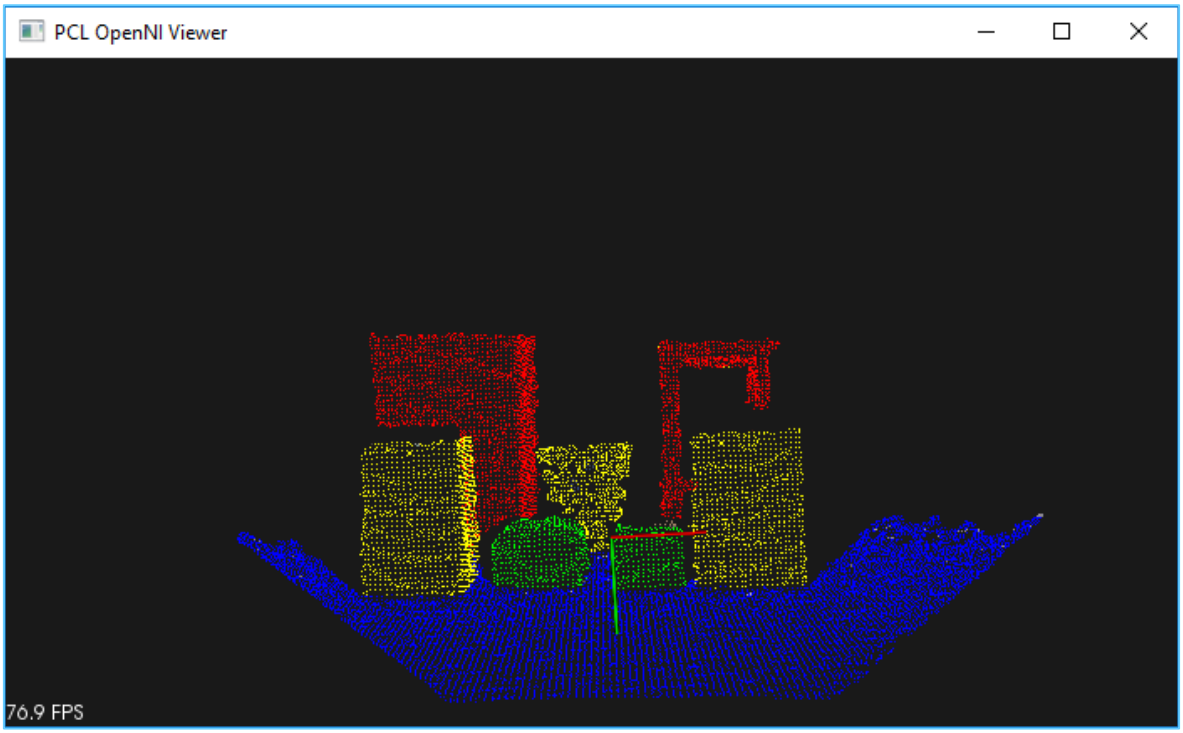
Para a elaboração do segundo caso de teste, os objetos foram organizados de forma paralela. As figuras abaixo representam o cenário original e os resultados obtidos para o modelo 1 e para o modelo 2 (adotando ângulos de suavidade 3π rad e 7π rad).

Figura 32 – Cenário original do caso 2



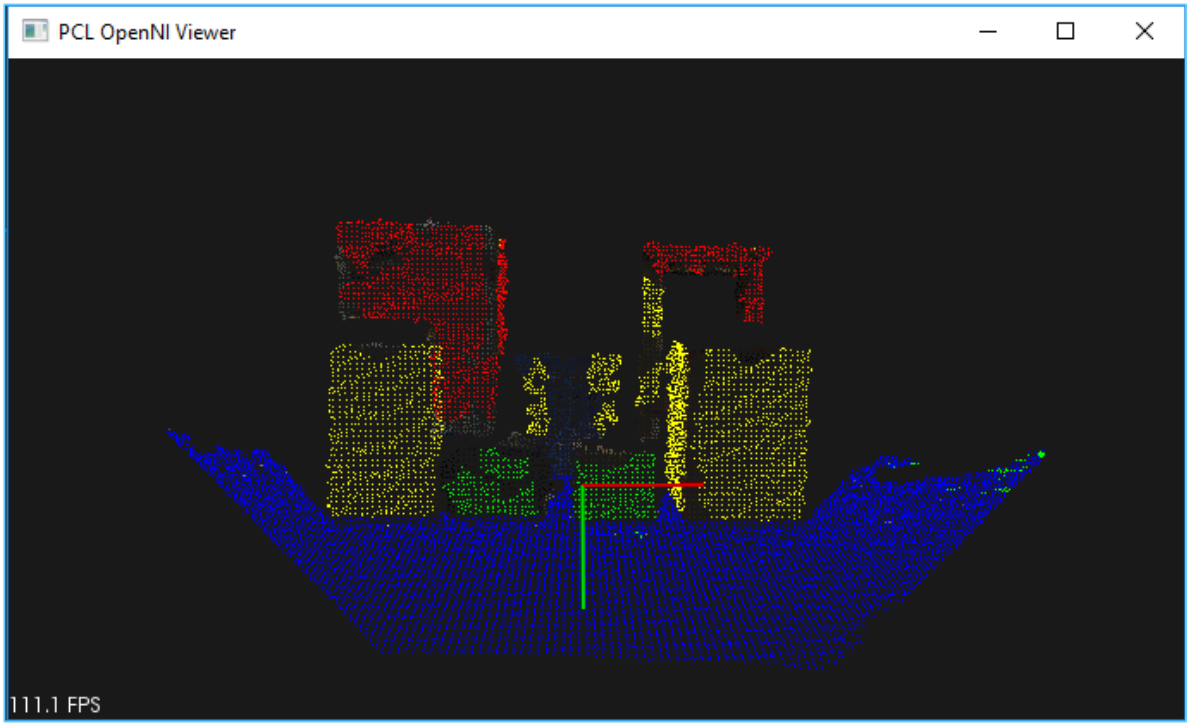
Fonte: elaborado pelo próprio autor

Figura 33 – Detecção de obstáculos paralelos no cenário - *Euclidean Segmentation*



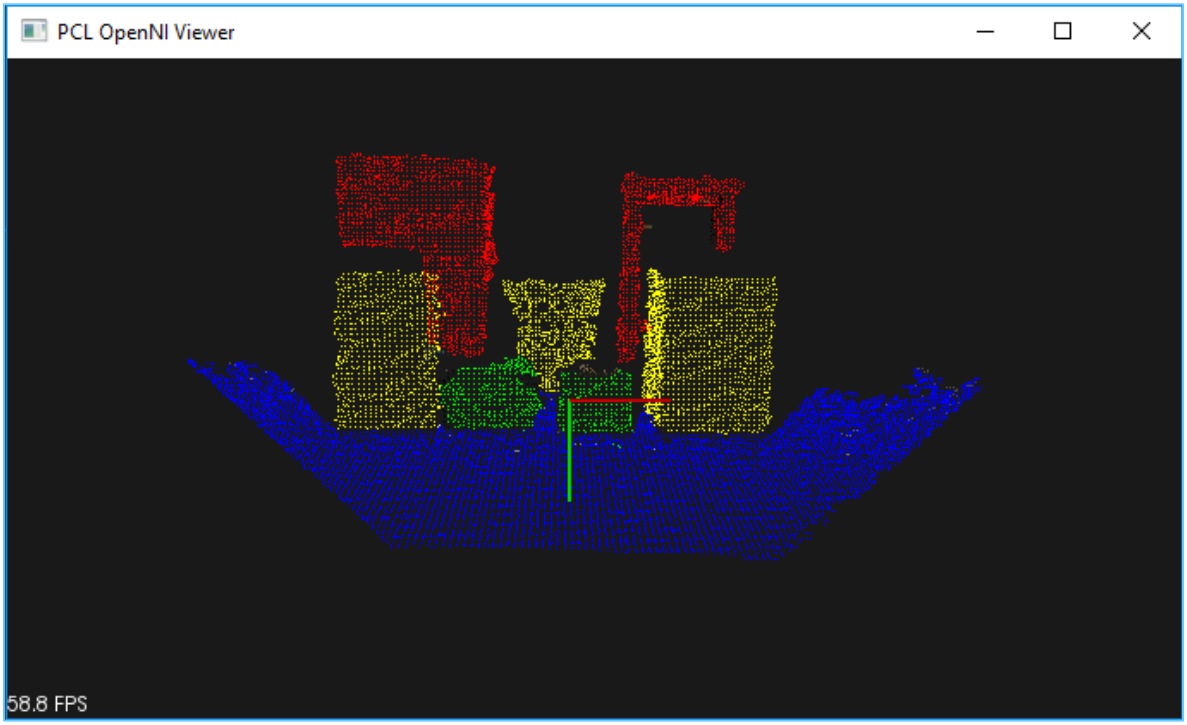
Fonte: elaborado pelo próprio autor

Figura 34 – Detecção de obstáculos dispersos no cenário – *Region Growing* (ângulo de suavidade de 3π rad)



Fonte: elaborado pelo próprio autor

Figura 35 – Detecção de obstáculos dispersos no cenário – *Region Growing* (ângulo de suavidade de 7π rad)



Fonte: elaborado pelo próprio autor

Tabela 7 - Caso 2: Detecção e Classificação de obstáculos paralelos

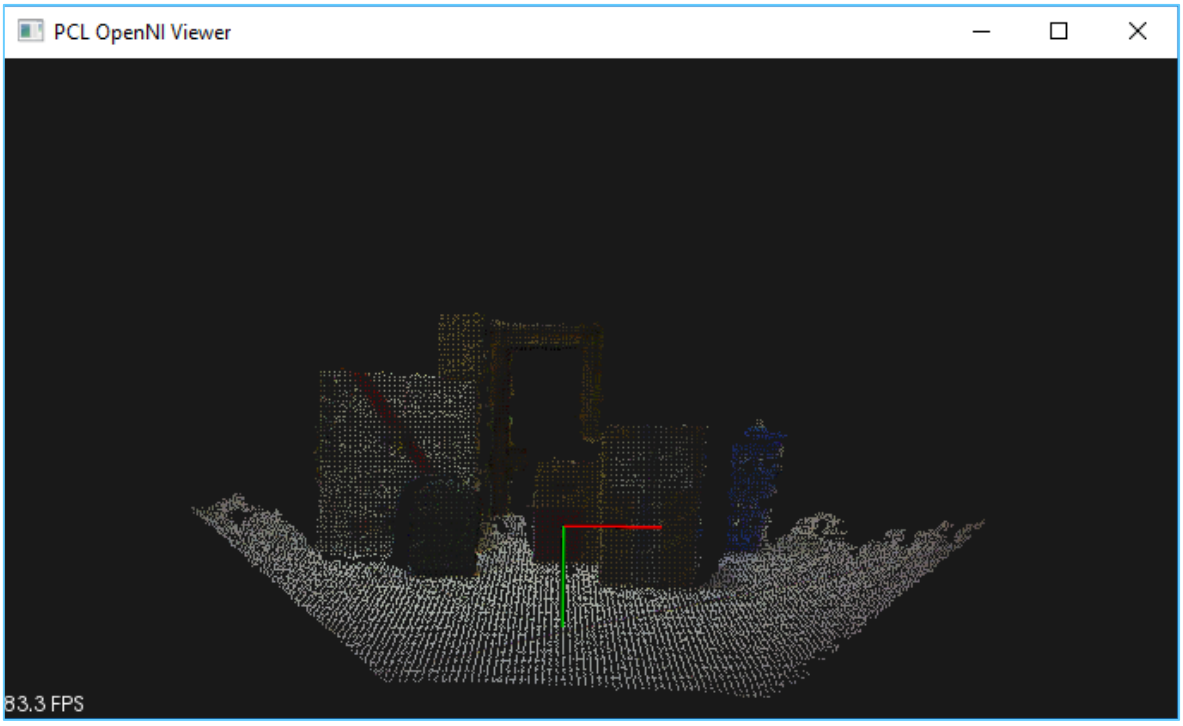
Obstáculo	**Resultado Final Desejado**	***Euclidean Segmentation***	**Region Growing (ângulo de suavidade 3º)**	**Region Growing (ângulo de suavidade 7º)**
Objeto 1	Verde	Verde	Parcialmente verde	Verde
Objeto 2	Verde	Verde	Parcialmente verde	Verde
Objeto 3	Amarelo	Amarelo	Parcialmente amarelo	Amarelo
Objeto 4	Amarelo	Amarelo	Parcialmente amarelo	Amarelo
Objeto 5	Amarelo	Amarelo	Erro	Amarelo
Objeto 6	Vermelho	Vermelho	Parcialmente amarelo	Vermelho
Objeto 7	Vermelho	Vermelho	Erro	Vermelho

Fonte: elaborado pelo próprio autor

6.3 Caso 3: Obstáculos Distribuídos de forma sobreposta

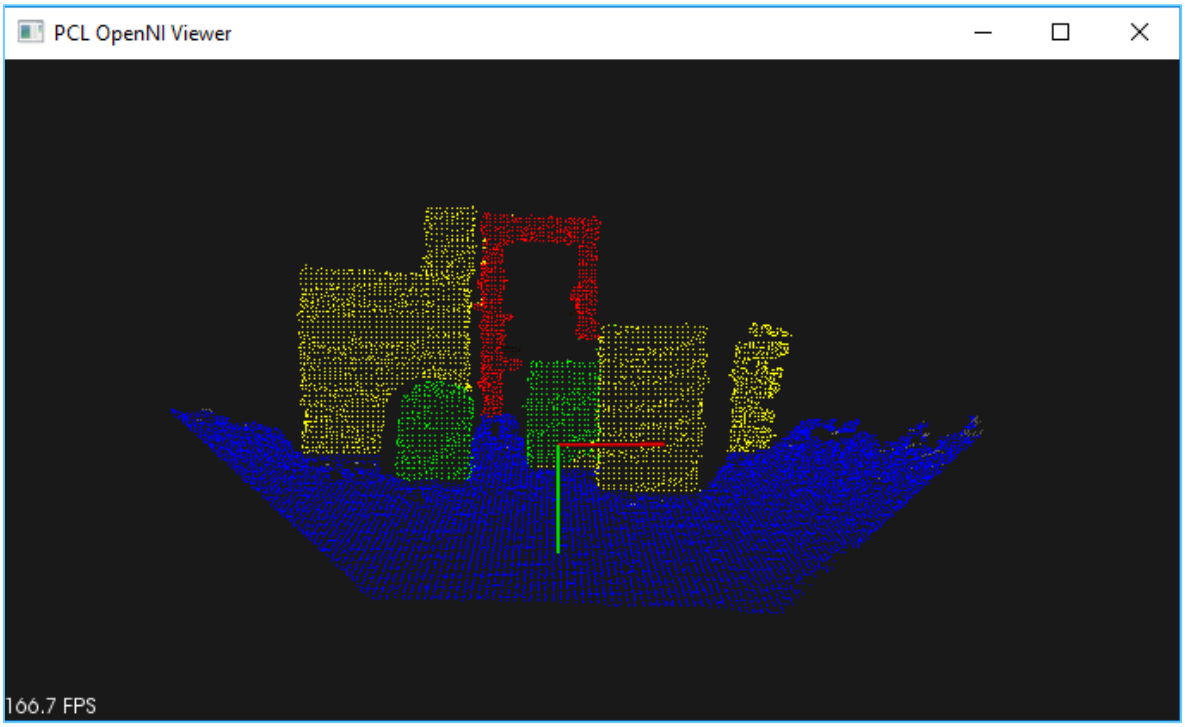
Para a elaboração do segundo caso de teste, determinados objetos foram organizados de forma sobreposta e outros reclinados. As figuras abaixo representam o cenário original e os resultados obtidos para o modelo 1 e para o modelo 2 (adotando ângulos de suavidade 3π rad e 7π rad).

Figura 36 – Cenário original do caso 3



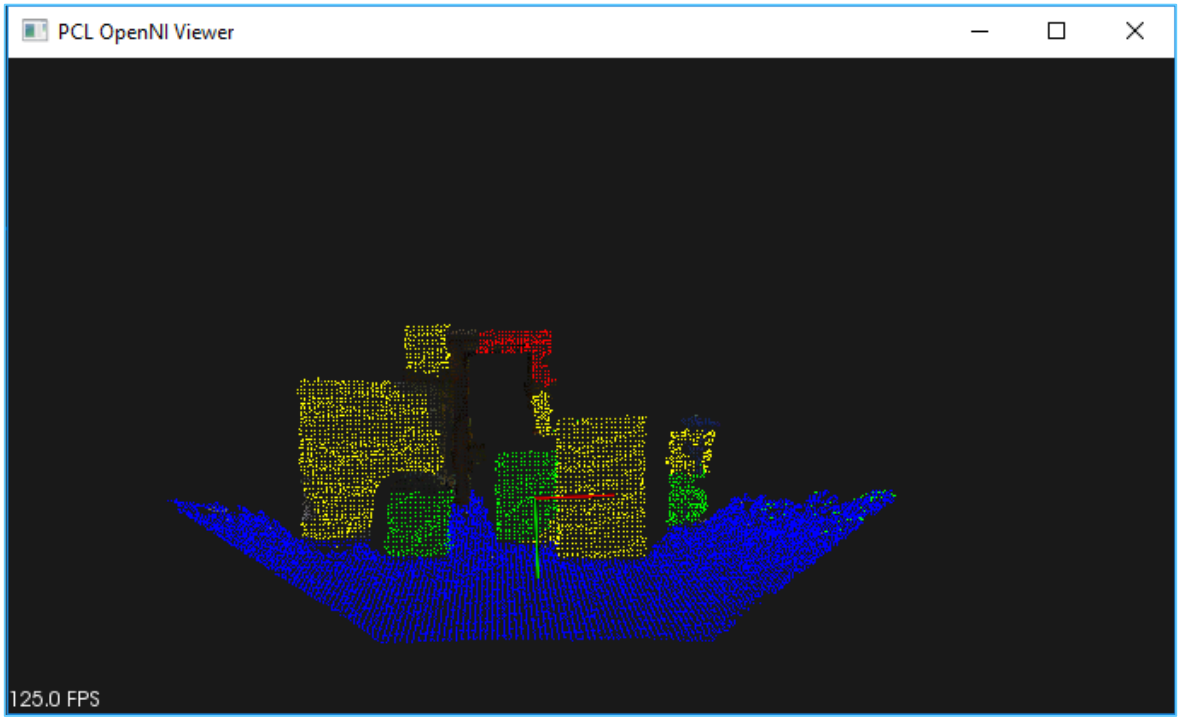
Fonte: elaborado pelo próprio autor

Figura 37 – Detecção de obstáculos paralelos no cenário - *Euclidean Segmentation*



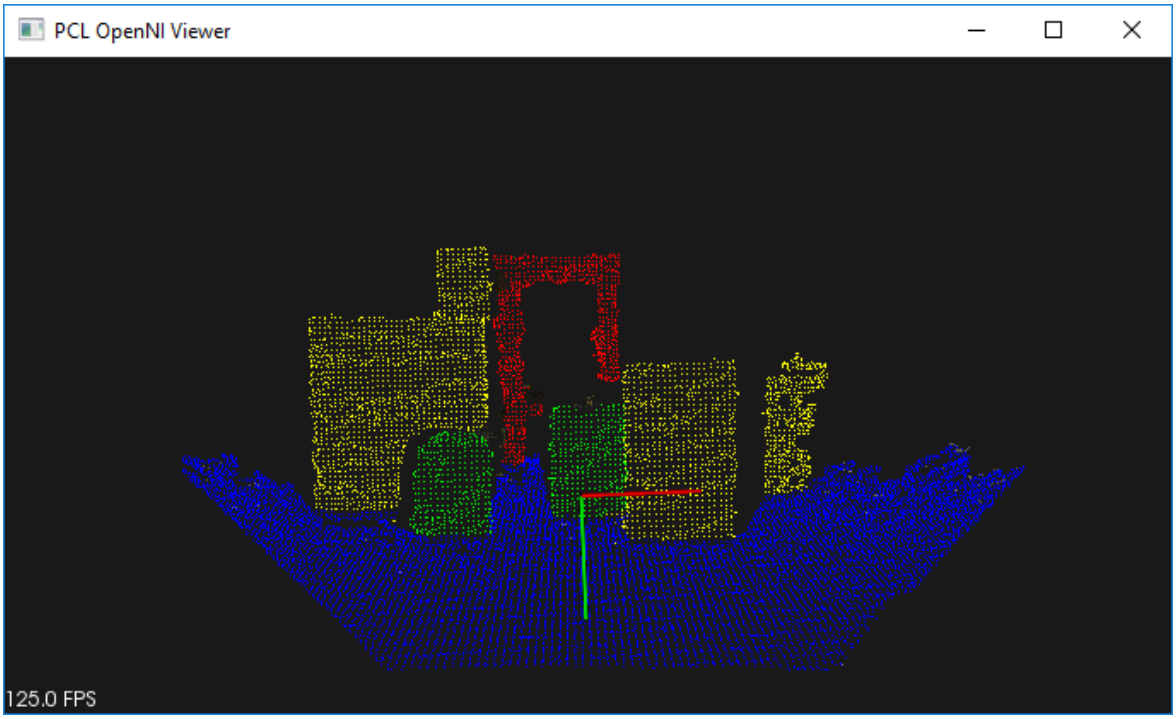
Fonte: elaborado pelo próprio autor

Figura 38 – Detecção de obstáculos dispersos no cenário – *Region Growing* (ângulo de suavidade de 3π rad)



Fonte: elaborado pelo próprio autor

Figura 39 – Detecção de obstáculos dispersos no cenário – *Region Growing* (ângulo de suavidade de 7π rad)



Fonte: elaborado pelo próprio autor

Tabela 8 - Caso 1: Detecção e Classificação de obstáculos sobrepostos

Obstáculo	**Resultado Final Desejado**	***Euclidean Segmentation***	**Region Growing (ângulo de suavidade 3º)**	**Region Growing (ângulo de suavidade 7º)**
Objeto 1	Verde	Verde	Parcialmente verde	Verde
Objeto 2 (*h=13,4cm +36,3cm=49,7cm)*	Amarelo	Amarelo	Parcialmente amarelo	Amarelo
Objeto 3 reclinado (*h=18,9cm*)	Verde	Verde	Verde	Verde
Objeto 4	Amarelo	Amarelo	Parcialmente amarelo	Amarelo
Objeto 5	Amarelo	Amarelo	Erro	Amarelo
Objeto 6 reclinado (*h=36,3 cm)*	Amarelo	Amarelo	Amarelo	Amarelo
Objeto 7	Vermelho	Vermelho	Erro	Vermelho

Fonte: elaborado pelo próprio autor

Os resultados de cada caso de teste permitem perceber um comportamento similar entre o primeiro e segundo modelo se considerarmos o ângulo de suavidade (*Smoothness Threshold*) com o valor de 7π rad, sendo capazes de realizar a classificação dos obstáculos de acordo com os resultados esperados de acordo com os intervalos definidos. Tendo no caso 1, o modelo 1 com relevante desempenho de detecção dos objetos 6 e 7, apresentando menos distorções.

Ao analisar os resultados do modelo 2 ao aplicar ângulo de suavidade de valor 3π rad para os três casos de teste, pode-se observar um considerável número de erros na detecção e classificação de obstáculos.

7 CONCLUSÃO

A partir do estudo bibliográfico realizado a respeito de nuvens de ponto, o trabalho em questão foi desenvolvido com a proposta de utilizar esse recurso para a detecção e classificação de obstáculos. Dessa forma, os métodos de segmentação *Euclidean Segmentation* e *Region Growing* foram avaliados quanto a capacidade classificação. Permitindo assim, que o objetivo geral da dissertação fosse alcançado.

O trabalho foi desenvolvido através da utilização da biblioteca de manipulação de nuvens de ponto, denominada PCL. Por via dessa ferramenta, foi possível desenvolver dois modelos distintos, adotando os métodos de segmentação estabelecidos. Desse modo, foi possível realizar a implementação e análise da detecção e classificação de obstáculos. Deste modo, os objetivos específicos também foram cumpridos.

De forma geral, a utilização de um método de segmentação adequadamente para a classificação de objetos, tem grande valor para a visão computacional, visto que a adoção e especificação de uma técnica adequada permite a extração e manipulação dos dados das imagens capturadas. Sendo assim, as principais contribuições deste trabalho são:

- Análise e avaliação da utilização de nuvens de ponto para a extração de dados de uma imagem em tempo real;
- Avaliação dos recursos disponíveis pela biblioteca PCL para o processamento de nuvens de ponto;
- Análise comparativa dos métodos de segmentação *Euclidean Segmentation* e *Region Growing;*
- Especificação, desenvolvimento e aplicação de dois modelos com técnicas de segmentação distintas para a detecção e classificação de obstáculos de acordo com a altura;

Devido ao fato da PCL ser um recurso recente e pouco explorada, existem diversas aplicações que podem ser exploradas como extensão do presente trabalho em futuros estudos. Dentre essas, podem ser destacados:

- Implementar os modelos desenvolvidos para a definição de rotas;
- Utilizar da PCL associada à ferramenta ROS, para criar aplicativos de definição de rotas robôs;
- Especificar e classificar obstáculos adotando outras variáveis além da altura;

8 REFERÊNCIAS BIBLIOGRÁFICAS

ALDOMA, A. et al. (2012). Tutorial: Point cloud library: Three-dimensional object recognition and 6 dof pose estimation. *IEEE Robotics & Automation Magazine, 19*(3), 80-91.

BALLARD, D. H., & BROWN, C. M. (1982). *Computer Vision.* Englewood Cliffs, NJ: Prenice-Hall.

BORGES, D. (2013). *Reconstrução de Objectos 3D Usando Kinect.*

CORREIA, M. M. (2013). *Reconhecimento de elementos gestuais com kinect.* Universidade do Porto, Faculdade de Engenharia.

DUARTE, A. (2015). *Reconhecimento de Objetos Baseado.* Universidade do Porto, Faculdade de Engenharia.

GOIS, J. (2004). *Reconstrução de superfícies a partir de nuvens de pontos.* Universidade de São Paulo, Tese de Doutorado.

GONZALES, R., & WOODS, R. (2010). *Processamento digital de Imagens* (3 ed.). Pearson.

GRUPO DE ROBÓTICA. (2015). *PCL/OpenNI tutorial*. Acesso em Janeiro de 2017, disponível em Robótica Unileon: http://robotica.unileon.es/index.php/PCL/OpenNI_tutorial_2:_Cloud_processing_(basic)

HUANG, H.-C., HSIEH, C.-T., & YEH, C.-H. (2015). An indoor obstacle detection system using depth information and region growth. *Sensors, 15*(10), 27116-27141.

JOSÉ, M. (2008). *Reconstrução tridimensional de baixo custo a partir de par de imagens estéreo.* Tese de Doutorado, Universidade de São Paulo.

KHOSHELHAM, K., & ELBERINK, S. (2012). Accuracy and Resolution of Kinect Depth Data for Indoor Mapping Applications. *Sensors, 12*(2), 1437-1454.

LIMBERGER, F., & OLIVEIRA, M. (2015). Real-time detection of planar regions in unorganized point clouds. *Pattern Recognition, 48*, pp. 2043-2053.

MACHADO, R. (2013). *PROPOSTA DE SISTEMA BASEADO NA PLATAFORMA KINECT PARA SUPORTE A REABILITAÇÃO DE PACIENTES COM PATOLOGIAS LIGAMENTARES NOS JOELHOS.* Universidade Federal de Santa Catarina - Campus Araranguá.

MARTINS, R. (2014). *Aquisição e processamento de modelos tridimensionais.* Universidade de Brasília, Departamento de Ciência da Computação.

MICROSOFT. (2017). *Componentes do sensor Kinect para Xbox 360*. Acesso em Fevereiro de 2017, disponível em Suporte do Xbox: http://support.xbox.com/pt-BR/xbox-360/accessories/kinect-sensor-components

NGUYEN, V.-D. (2012). *Obstacle avoidance using the Kinect.* Acesso em Janeiro de 2017, disponível em SCRIBD: https://pt.scribd.com/document/80464002/Obstacle-Avoidance-Using-the-Kinect

OZBAY, E., & CINAR, A. (2013). 3D Reconstruction Technique with Kinect and Point Cloud Computing. *AWERProcedia Information Technology & Computer Science. [Online]*, *3*, pp. 1748-1754. Acesso em Fevereiro de 2017, disponível em http://www.world-education-center.org/index.php/P-ITCS

PALNICK, J. (2014). *Plane Detection and Segmentation for DARPA Robotics Challenge.* WORCESTER POLYTECHNIC INSTITUTE.

PCL. (Maio de 2011). *Point Cloud Library*. Acesso em Fevereiro de 2017, disponível em http://pointclouds.org

PEDRINI, H., & SCHWARTZ, W. R. (2008). *Análise de imagens digitais: princípios, algoritmos e aplicações.* Thomson Learning.

QUEIRÓS, H. (2012). *Exploração da Point Cloud Library aplicada à percepção em sistemas autónomos.* Instituto Politécnico do Porto. Instituto Superior de Engenharia do Porto., Tese de Doutorado.

RUSU, R., & COUSINS, S. (2011). 3d is here: Point cloud library (pcl). *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pp. 1-4.

SANTOS, G. (2016). *Novas abordagens para segmentação de nuvens de pontos aplicadas à robótica autônoma e reconstrução 3D.* Universidade Federal de Goiás, Escola de Engenharia Elétrica, Mecânica e de Computação.

SOUSA, A. (2012). *MAC5832: Reconhecimento de objetos com Kinect.* Universidade de São Paulo , Instituto de Matematica e Estatística.

SOUSA, A. (2015). *Superfície mágica: criando superfícies interativas por meio de câmeras RGBD e projetores.* Tese de Doutorado, Universidade de São Paulo., Instituto de Matemática e Estatística.

TEIXEIRA, J. et al. (2011). *Uma avaliação estatística do problema de registro de imagens de profundidade usando o Kinect.* Universidade Federal de Pernambuco, Centro de Informática. TR420, ICS-FORTH.

XU, W. et al. (2014). Nonrigid surface registration and completion from RGBD images. *European conference on computer vision.*, pp. 64-79.