UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO DE JOINVILLE
CURSO DE ENGENHARIA MECATRÔNICA

ELOIZA HEERDT

DETECÇÃO DE OBJETOS UTILIZANDO SENSOR LIDAR

Joinville
2023

ELOIZA HEERDT

DETECÇÃO DE OBJETOS UTILIZANDO SENSOR LIDAR

Trabalho de Conclusão de Curso apresentado como requisito parcial para obtenção do títulode bacharel em Engenharia Mecatrônica no curso de Engenharia Mecatrônica, da Universidade Federal de Santa Catarina, Centro Tecnológico de Joinville.

Orientador: Dr. Gian Ricardo Berkenbrock

Joinville
2023

AGRADECIMENTOS

Agradeço aos meus pais, Vitor e Sandria, que sempre me apoiaram e não mediram esforços em me dar todas as oportunidades que podiam para que eu chegasse até aqui, sempre estando ao meu lado em cada desafio e me incentivando a continuar.

A todos os meus amigos, que deixaram essa jornada mais leve, que compartilharam momentos incríveis a cada ano que passou e estavam lá para me escutar quando precisei. Ao time de futsal feminino da UFSC Joinville e aos times das copinhas, são momentos que ficarão guardados junto as melhores lembranças da graduação. Um agradecimento especial a Jaqueline Bonatti, que esteve ao meu lado do início ao fim dessa caminhada, sempre me apoiando e me fazendo continuar firme, muito obrigada!

Agradeço aos professores que passaram pelo meu caminho ao longo dos anos. E ao meu orientador, Professor Dr. Gian Ricardo Berkenbrock, que além de me acompanhar nesse projeto, ter paciência e compreensão ao me orientar, teve grande influência e impacto na minha vida acadêmica e profissional, desde as aulas de Programação III, que foram no meio de um caos de pandemia, muito obrigada professor.

Agradeço ao laboratório LSE pelos equipamentos disponibilizados para que esse trabalho fosse executado, assim como os membros do laboratório que sempre me ajudaram quando precisei. E agradeço a Universidade Federal de Santa Catarina como um todo, é incrível poder ter a oportunidade e acesso gratuito a esse mundo de conhecimento e educação de qualidade.

Não sou capaz de citar diretamente a todos, mas gostaria de expressar minha profunda gratidão a todas as pessoas que contribuíram de alguma maneira, não só para a realização deste Trabalho de Conclusão de Curso, bem como, no meu caminho até aqui na graduação. Este trabalho representa o fim de uma trajetória de anos de estudo e esforço, e não teria sido possível sem o apoio e orientação de vocês.

RESUMO

Os veículos autônomos vêm se destacando como uma solução promissora para problemas envolvendo a mobilidade urbana. O sistema de direção autônomo é complexo e integra diferentes áreas de conhecimento. As tecnologias empregadas em tais sistemas podem ser utilizadas para diferentes finalidades, mas no geral envolvem sensoriamento, percepção do ambiente, planejamento de rota e controle de diferentes subsistemas. Reconhecer o ambiente à volta do veículo em questão é fundamental para qualquer ação de direção que se queira tomar. Um dos aspectos importantes para o veículo no ambiente são os diferentes obstáculos que possam existir ali. Portanto, nesse trabalho foi implementado um sistema capaz de detectar objetos utilizando um sensor LiDAR 2D. Foram desenvolvidos, um método simples de implementação ingênua, e outro método com maior filtragem utilizando uma biblioteca própria para lidar com nuvem de pontos. Por fim, foram comparados os seus desempenhos. Ambos atingiram o objetivo de detectar objetos, mesmo que com limitações impostas pela capacidade do sensor.

Palavras-chave: Veículos Autônomos. LiDAR. Percepção Ambiental.

ABSTRACT

Autonomous vehicles are emerging as a promising solution to problems involving urban mobility. The autonomous driving system is complex and integrates different areas of knowledge. The technologies employed in such systems can be used for different purposes, but in general involve sensing, environment perception, route planning, and control of different subsystems. Recognizing the environment around the vehicle in question is fundamental to any driving action that may be taken. One of the important aspects for the vehicle in the environment are the different obstacles that may exist there. Therefore, in this work it was implemented a system capable of identifying objects using a 2D LiDAR sensor. A simple method was developed, with a naive implementation, and another method with more filtering, using a specific library for dealing with point clouds. Finally, their performances were compared. Both methods achieved the objective of detecting objects, despite the limitations imposed due to the capacity of the sensor.

Keywords: LiDAR. Autonomous Vehicles. Enviromental Perception.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

A mobilidade urbana é um tema cada vez mais relevante na sociedade, especialmente em grandes cidades, onde o trânsito intenso e congestionamentos são problemas frequentes. Com o passar dos anos, a tecnologia segue avançando e se tornando uma aliada na solução desses problemas, e os sistemas autônomos vêm se destacando como uma alternativa promissora.

Um veículo autônomo pode percorrer trajetos por diferentes ambientes com nenhuma, ou quase nenhuma, interferência humana. A condução autônoma é formada por um sistema complexo que integra diversas tecnologias inovadoras, enfrentando desafios e oportunidades (DINGYI; HAIYAN; KAIMING, 2018). Esses sistemas não se limitam apenas a veículos, e são utilizados para diversos fins, e em ambientes distintos, por exemplo, ambientes industriais, rurais, entre outros.

Apesar dos esforços, conquistas e visibilidade que diferentes pesquisas na área de veículos autônomos tenham acumulado nos últimos anos, tais sistemas continuam sem o resultado de excelência desejado e ainda não atingiram o nível da capacidade humana de direção (PAREKH et al., 2022). Esse fato se dá em função da complexidade do sistema todo, exigindo o somatório de esforços de diferentes áreas.

As quatro principais tarefas desses sistemas são: sensoriamento, percepção, planejamento e controle. O sensoriamento é a aquisição de dados dos sensores; percepção significa extração de características dos dados coletados dos sensores; planejar significa criar uma trajetória factível; e o controle é responsável por executar essa trajetória (HORVÁTH; POZNA; UNGER, 2022). Portanto, a movimentação de um veículo autônomo depende da definição e do planejamento de uma rota baseada na geolocalização e nos obstáculos que possam existir no ambiente à sua volta.

A percepção ambiental é muito importante nesse processo, é a função que permite com que o veículo tenha as informações essenciais para a sua locomoção (PENDLETON et al., 2017). Esses dados podem incluir a localização dos obstáculos ao redor, a existência de irregularidades no solo, entre muitos outros fatores. Uma boa análise, além de fornecer uma boa direção, pode resultar em até mesmo previsões futuras de seus estados.

Detectar obstáculos e a distância desses para o veículo é importante para decisões de como prosseguir o deslocamento. Considerando os requisitos necessários em sistemas como o planejamento de rota da direção autônoma, visa-se a implementação de um sistema de detecção de objetos utilizando um sensor *Light Detection and Ranging* (LiDAR). Para isso, serão implementados dois métodos utilizando um sensor com alcance em duas dimensões. Por fim, é feita a comparação entre o método utilizando uma biblioteca própria para lidar com nuvem de pontos e o

método simplificado proposto.

1.1 OBJETIVOS

1.1.1 Objetivo Geral

Detectar objetos utilizando um sensor LiDAR para aplicações em sistema de navegação veicular.

1.1.2 Objetivos Específicos

- Identificar as diferentes técnicas e tipos de sensores atualmente utilizados para detecção de objetos em veículos terrestres;
- Comparar as técnicas de detecção selecionadas;
- Desenvolver um sistema de detecção de objetos;
- Avaliar o sistema implementado.

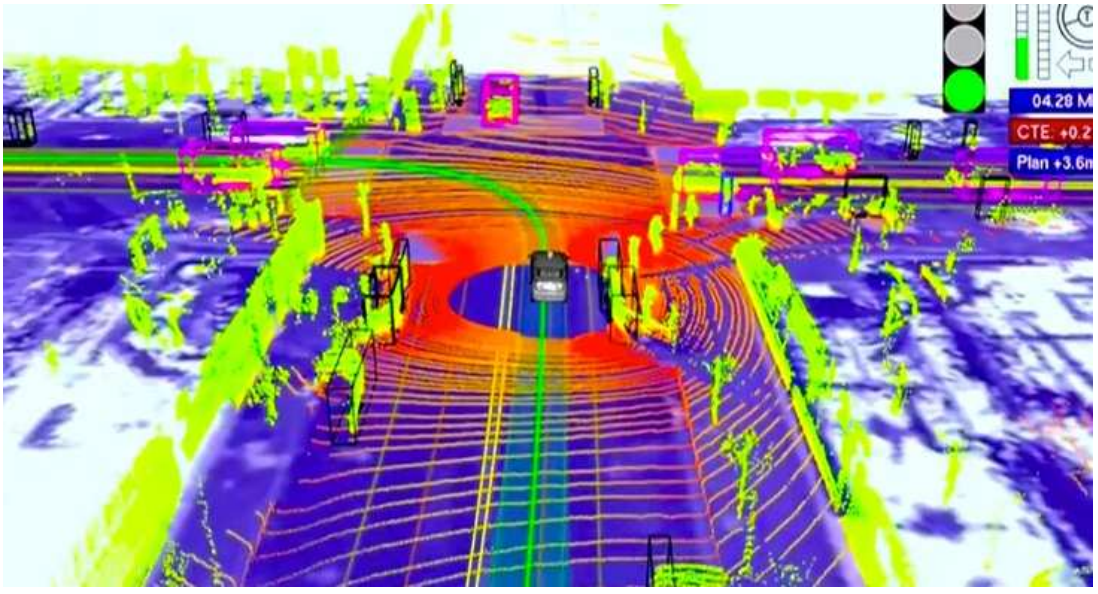
2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo serão abordados os principais tópicos para a fundamentação do tema tratado nesse trabalho e concepção de análise e proposta de solução. Inicialmente trata-se de percepção ambiental utilizando diferentes métodos de sensoriamento. No segundo tópico é abordado o funcionamento do LiDAR e suas aplicações. O tópico seguinte diz respeito às diferentes técnicas utilizadas para a detecção de objetos. O capítulo é finalizado com trabalhos similares.

2.1 PERCEPÇÃO AMBIENTAL

A percepção ambiental desempenha um papel fundamental no contexto da direção autônoma (Figura 1) e navegação de veículos e robôs, influenciando significativamente suas decisões e desempenho. Para realizar essa tarefa, é comum empregar uma variedade de tecnologias, como LiDARs, câmeras e a combinação de ambos, ou ainda, incluir o uso de radares de curto e longo alcance, juntamente com sensores ultrassônicos (PENDLETON et al., 2017).

Figura 1 – "Visão" do computador de bordo de um veículo autônomo da Google.

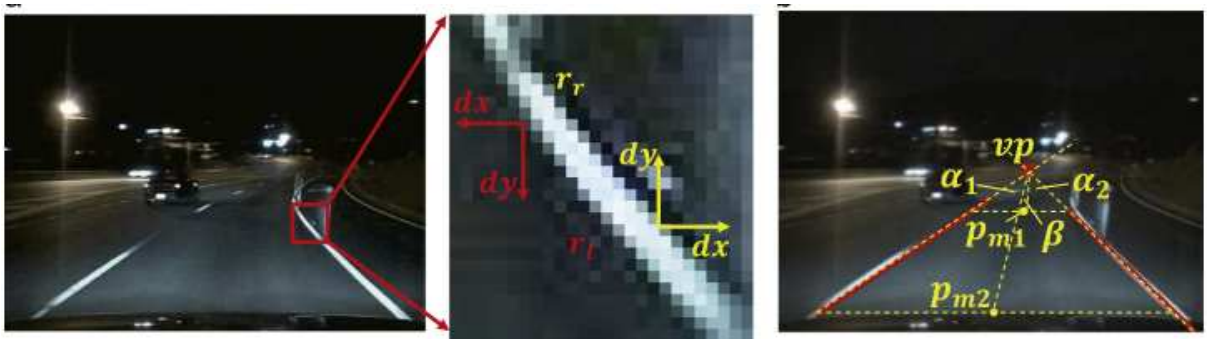


Fonte: IEEE Spectrum (2011)

Wang et al. (2023) abordam a identificação de placas de trânsito, um aspecto de suma importância para as capacidades de percepção necessárias aos veículos autônomos, permitindo-lhes navegar com segurança em vias públicas. O reconhecimento do ambiente pode ter uma variedade de finalidades, incluindo a detecção de obstáculos em geral, a identificação de pedestres e até mesmo o

reconhecimento de faixas de estrada (Figura 2).

Figura 2 – Parte do processo de percepção ambiental baseado em faixas de estrada.



Fonte: Niu et al. (2016, p. 229)

2.1.1 Radar

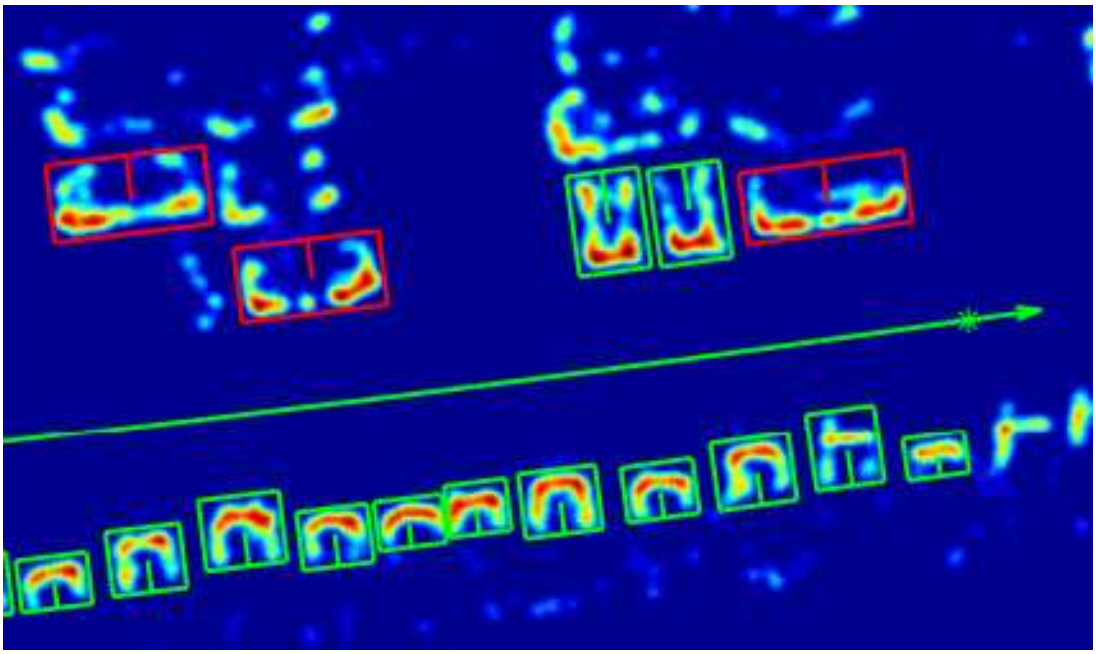
O radar é um sistema de detecção de objetos no qual utiliza do sinal de ondas de rádio para determinar o alcance, a distância ou até mesmo a velocidade de um objeto (ZHU et al., 2017). O sensoriamento por radar funciona muito bem em aplicações com um campo de visão limitado e de forma bastante consistente em diferentes condições climáticas e de iluminação. Porém, as medições são bastante ruidosas, exigindo filtragem e limpeza extensivas (SIVARAMAN, 2013).

Xing, Peng e Pan (2022) promovem a análise do desempenho de um radar de ondas milimétricas para a percepção ambiental aplicada de forma estática e em movimento. O estudo busca os motivos da resolução e precisão insuficientes para o uso seguro em veículos inteligentes e propõe métodos de otimização para melhoria desse desempenho. Os experimentos, nesse caso, foram feitos com o radar posicionado a frente de uma locomotiva.

Embora os dados de um radar não tenham uma densidade comparável a de sensores ópticos, o tempo de assentamento dos filtros, a convergência dos filtros e os parâmetros dinâmicos, como velocidade relativa, serão mais rápidos e robustos (DICKMANN et al., 2016). Dubé et al. (2014) comprovam a possibilidade de utilizar dados de radar para detectar objetos. Nesse caso, implementaram a detecção de carros estacionados utilizando dados de um radar em tempo real (Figura 3).

Ao avaliar o potencial de iniciativas desenvolvidas por corporações de renome, o estudo de Dickmann et al. (2014) destacou de maneira significativa a relevância dos sistemas de radar na progressão das abordagens tecnológicas empregadas em veículos autônomos. Essa pesquisa empreendeu uma análise aprofundada, abordando com destaque os consideráveis desafios e as promissoras perspectivas no caminho rumo a uma condução autônoma segura.

Figura 3 – Detecção de veículos estacionados em um estacionamento.



Fonte: Dubé et al. (2014, p. 1420)

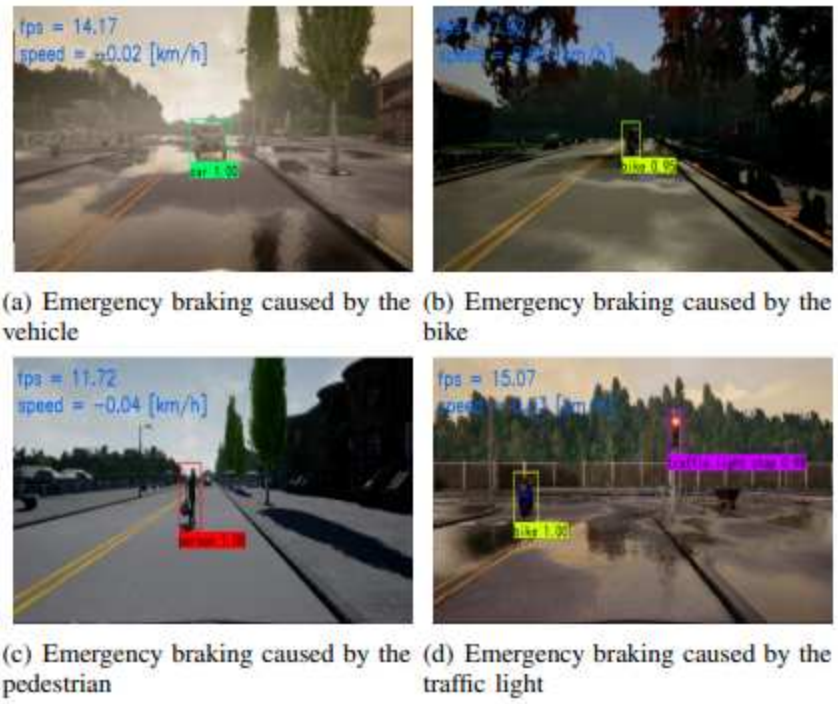
2.1.2 Câmeras

Diferentemente do radar, as câmeras não emitem energia eletromagnética, em vez disso, elas capturam a luz ambiente presente na cena. Isso significa que a detecção de objetos não pode depender de um sinal de referência refletido. Dependendo do objetivo de precisão pretendido em diferentes sistemas de percepção ambiental, a utilização de câmera pode requerer um substancial poder computacional, uma vez que necessita a aplicação de técnicas de visão computacional (SIVARAMAN, 2013).

Embora exija um sistema de processamento robusto em alguns casos, o uso de câmeras também apresenta vantagens, obtendo informações ricas do ambiente (CHEN et al., 2021b). As mesmas habilitam uma ampla gama de aplicações além da simples detecção de objetos, incluindo o reconhecimento e classificação de cada objeto. Isso possibilita, por exemplo, a distinção de comportamentos com base no estado dos semáforos em uma estrada.

Wang, Sun e Zhu (2023) apresentam o desenvolvimento de um sistema modular para direção autônoma, com o módulo responsável pela percepção do ambiente todo baseado em análise de visão computacional. Ou seja, as entradas do sistema são imagens capturadas por uma câmera posicionada a frente do veículo, as quais são utilizadas para identificação de objetos, enviando tais informações para o módulo responsável pela tomada de decisões. Inicialmente algumas imagens foram pré-processadas para treinamento do sistema, e depois experimentos em um ambiente de simulação foram analisados para validação, utilizando imagens de diferentes situações e condições climáticas (Figura 4).

Figura 4 – Diferentes situações em que o freio foi acionado pelo sistema baseado em visão computacional.



(a) Emergency braking caused by the vehicle (b) Emergency braking caused by the bike

(c) Emergency braking caused by the pedestrian (d) Emergency braking caused by the traffic light

Fonte: Wang, Sun e Zhu (2023, p. 13)

2.2 LIDAR

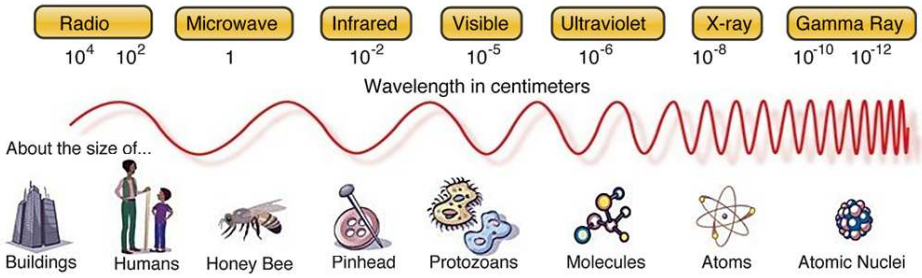
Light Detection and Ranging (LiDAR) é um método de sensoriamento utilizado para medição de distâncias e análises de diferentes ambientes. É semelhante ao radar no sentido de que explora o eletromagnetismo para a detecção e o alcance de objetos espaciais, e é semelhante às formas ópticas de sensoriamento remoto no sentido de que usa a óptica para a refração dessas ondas eletromagnéticas (EM) (LEEUWEN; NIEUWENHUIS, 2010).

O funcionamento de um sensor LiDAR é baseado no envio da luz do laser pulsado para o objeto e medição da luz refletida de volta, sendo possível, assim, analisar a variação no comprimento de onda e o tempo que a luz demorou para retornar, ou seja, a luz precisa atravessar um meio, geralmente a atmosfera, para chegar a um alvo (MCMANAMON, 2019). Essa análise permite obter informações detalhadas sobre a geometria, distância, textura e características do objeto ou ambiente que está sendo escaneado.

As ondas EM utilizadas pelo LiDAR são nos comprimentos de onda óptico e infravermelho (Figura 5). Esses sensores podem ser classificados como sensores ativos, que emitem a própria energia, enviando as ondas EM e recebendo o sinal refletido de volta (MCMANAMON, 2019).

O LiDAR é um ótimo sensor para identificar objetos. Um dos motivos para isso

Figura 5 – Espectro Eletromagnético



Fonte: Administração Nacional da Aeronáutica e Espaço dos Estados Unidos (NASA, 2015)

é a possibilidade de operar em comprimentos de onda semelhantes aos que o olho está acostumado a ver. Além disso, um sensor ativo tem vantagens à noite, podendo iluminar o ambiente com comprimentos de onda menores do que a radiação noturna disponível, proporcionando assim uma melhor resolução (MCMANAMON, 2019).

Existem sensores LiDAR que funcionam com a capacidade de alcance em diferentes dimensões, ou seja, 1D que mede somente o alcance, 2D (Figura 6) que mede a distância horizontal até os alvos e o ângulo em relação ao sensor para obter dados nos eixos X e Y, ou 3D, que pode medir ângulos verticais e horizontais e o alcance (MCMANAMON, 2019). Os sensores LiDAR são divididos em duas formas de detecção, a detecção direta e a coerente.

Figura 6 – LiDAR - Modelo RPlidar A1M8 Slamtec.

Fonte: Adaptado de SLAMTEC (2023)

Na detecção direta, o sensor LiDAR emite pulsos de luz e mede o tempo que leva para esses pulsos retornarem após atingir um objeto. Essa medição do tempo é usada para calcular a distância entre o sensor e o objeto. Essa abordagem é conhecida como *tempo de voo* (time-of-flight) e é bastante comum em sensores LiDAR mais

simples e mais baratos (BHASKARAN, 2018).

Na detecção coerente, o sensor utiliza uma técnica chamada interferometria para medir distâncias com alta precisão, basicamente, o sensor divide o sinal de luz em dois feixes: um feixe de referência e um feixe de retorno do objeto. Esses feixes são combinados novamente e analisados para medir a diferença de fase. A partir dessa diferença, a distância até o objeto pode ser calculada com alta precisão (KADLEC et al., 2019).

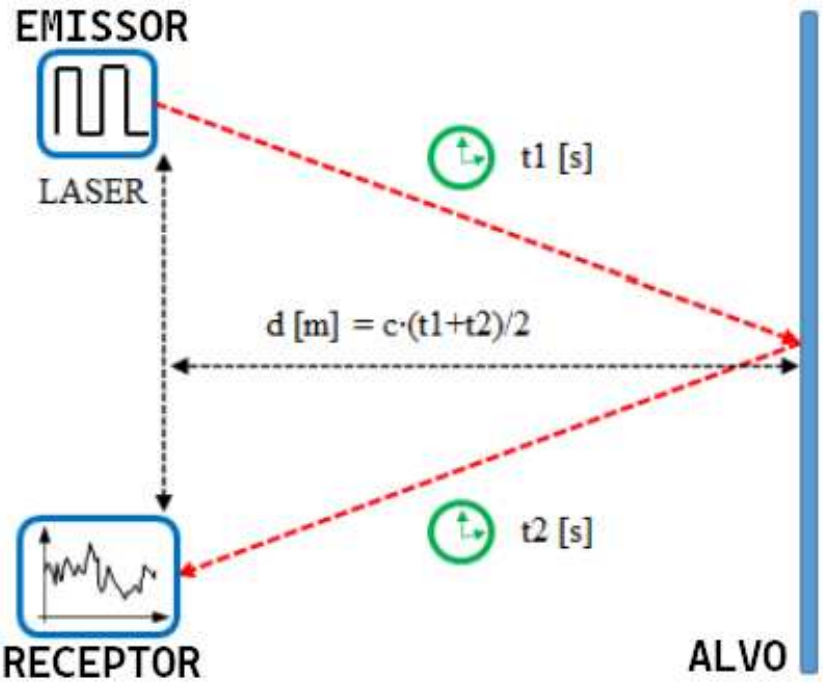
A Figura 7 mostra o princípio de funcionamento de um sistema LiDAR trabalhando com o método time-of-flight. Nesse caso a distância para o alvo pode ser calculada de forma simples (Equação 1) (TUDOR et al., 2021).

$$D = c\frac{\Delta T}{2} \tag{1}$$

Onde:

c - Velocidade da luz no meio de propagação;
ΔT - Tempo que a luz levou para retornar.

Figura 7 – Sistema LiDAR.



Fonte: Adaptado de TUDOR et al. (2021, p. 3)

Um sistema como esse é composto por alguns subsistemas principais, a

unidade de iluminação, que envia os pulsos de laser ao objeto-alvo; o sistema óptico, onde uma lente acumula a luz refletida e projeta-a no sensor; e a unidade de controle, que precisa controlar o sistema todo, mantendo uma calibração na precisão das medições (TUDOR et al., 2021).

O receptor do sistema (Figura 7), tem por objetivo converter os fótons retornados do alvo em informações. O tipo do LiDAR e seu processamento, associados, é o que determina as informações exatas buscadas pelo receptor, entretanto, todos os tipos de receptores LiDAR tem ainda, que lidar com os ruídos (MCMANAMON, 2019).

A quantidade de ruídos e a forma com que serão tratados depende, também, da aplicação a qual o sensor está inserido. No geral, se o comprimento de uma onda EM for significativamente maior do que o tamanho de uma partícula, a onda fluirá ao redor das partículas com pouca ou nenhuma atenuação (MCMANAMON, 2019).

No caso da utilização do sensor LiDAR em carros autônomos, as interferências podem vir de diferentes formas. O tamanho de gotículas de chuva e partículas de neblina comparados ao comprimento de onda emitido pelos sensores LiDAR, por exemplo, fazem com que o sensor tenha dificuldades de cumprir seu comportamento esperado em ambientes onde o clima não está favorável (MCMANAMON, 2019).

Periu C.F. e McLaughlin (2013) demonstram outro importante tópico, que são as vibrações mecânicas transmitidas em um LiDAR aplicado a um veículo agrícola. Houve uma melhoria significativa na acurácia do sensor utilizando o sistema de estabilização desenvolvido pelos autores. Evidentemente, isso depende de diversos fatores como velocidade do veículo, tipo de solo entre outros, porém mostra a importância das análises de diferentes ruídos.

Apesar das diversas interferências em que um sensor LiDAR pode estar vulnerável, as ondas EM desse sensor podem ser transmitidas por meio de diferentes meios, como o vácuo do espaço, ou até mesmo através da água (MCMANAMON, 2019). O potencial desse sensor e a possibilidade que o mesmo traz não só em medições de distância propriamente ditas, mas também em análises completas de diferentes ambientes, o faz ser uma escolha em diversos projetos.

Colin e Faivre (2010) utilizam da tecnologia do LiDAR em conjunto com uma câmera de alta resolução para análise da rugosidade da superfície terrestre, associada a troca de calor em volta de uma típica bacia de rio no noroeste da China. O estudo tem o objetivo de melhorar o entendimento sobre processos hidrológicos e ecológicos relacionados a bacia hidrográfica.

Há também estudos da utilização desse sensor para uma análise complementar dos oceanos. Jamet et al. (2019) demonstram como diferentes tipos de LiDAR podem contribuir para o estudo sobre a morfologia das partículas e composição química que existe no oceano.

2.3 TÉCNICAS DE DETECÇÃO

Há uma variedade extensa de aplicações para os sensores LiDAR, que podem ser empregados de diversas maneiras, seja de forma independente ou em combinação com câmeras e/ou outros sensores. Nessa seção, serão abordadas algumas das técnicas para a detecção de objetos com o uso desse sensor, especialmente o LiDAR 2D.

O conjunto de dados produzidos por um sensor LiDAR (2D ou 3D) são comumente chamados de *nuvem de pontos*, representando o conjunto de todas as medições de distância que o sensor realiza em relação aos objetos dentro de seu alcance. As coordenadas dos pontos dessa nuvem são fornecidas em um sistema de coordenadas local ou global, dependendo do tipo do sistema LiDAR e da área de aplicação (BENEDEK et al., 2021).

Apesar de encontrar muitas abordagens de detecção e identificação de objetos na literatura, não há um conjunto de técnicas rigidamente estabelecidas para o processamento desses dados, devido às grandes diferenças nas características dos dados obtidos por diferentes sensores LiDAR (BENEDEK et al., 2021). Entretanto, uma das técnicas clássicas encontradas é o *clustering*, ou agrupamento, que pode ser subdividido em alguns métodos diferentes, como os métodos com distância Euclidiana e os com supervoxels ou superpontos.

O método de agrupamento com distância Euclidiana primeiro constrói uma kd-tree, a qual é uma árvore de buscas (BLACK, 2020), em toda a nuvem de pontos. Depois, comparando as distâncias Euclidianas dos pontos, agrupa todos os pontos vizinhos em um limite de raio configurado. Intuitivamente, um limite maior possivelmente agrupará objetos próximos todos juntos, mas um limiar menor é mais sensível às partes de objetos que contém poucos pontos (ZHAO; ZHANG; HUANG, 2021). A distância Euclidiana, em um sistema de coordenadas cartesianas, é basicamente o comprimento de um segmento de linha entre dois pontos no espaço (Equação 2) (BLACK, 2004).

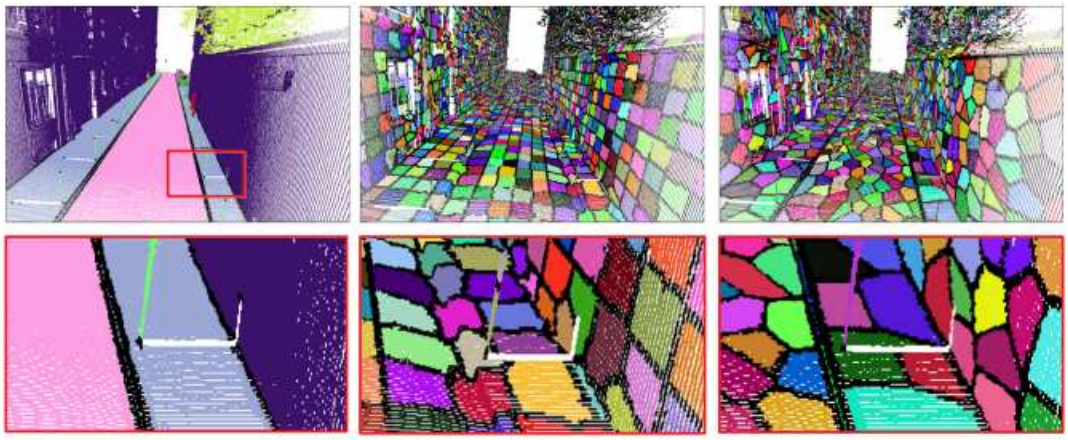
$$d(p,q) = \sqrt{(x_p - x_q)^2 + (y_p - y_q)^2} \tag{2}$$

O método de *clustering* que utiliza superpixels, ou supervoxels, tem o objetivo de reduzir o número de regiões que devem ser consideradas posteriormente por algoritmos que necessitam de maior poder computacional, com uma perda mínima de informações (PAPON et al., 2013). Lin et al. (2018) propuseram um método eficiente e adaptativo para a segmentação de supervoxels em nuvens de pontos tridimensionais, testando o método em três conjuntos de dados públicos de segmentação de nuvens de pontos.

Os resultados experimentais de Lin et al. (2018) mostram que, em comparação com os métodos de segmentação de supervoxels mais avançados, os supervoxels

extraídos usando seu método preservam os limites dos objetos e as estruturas menores de forma mais eficaz. A Figura 8 mostra como os métodos se comportaram em relação à preservação dos aspectos do ambiente, utilizando os limites da calçada, como rótulos da verdade da nuvem de pontos.

Figura 8 – Comparação entre métodos de supervoxel. À esquerda a nuvem de pontos com os rótulos da verdade, ao centro o método de Papon et al. (2013) e à direita o método de Lin et al. (2018).



Fonte: Lin et al. (2018, p. 40)

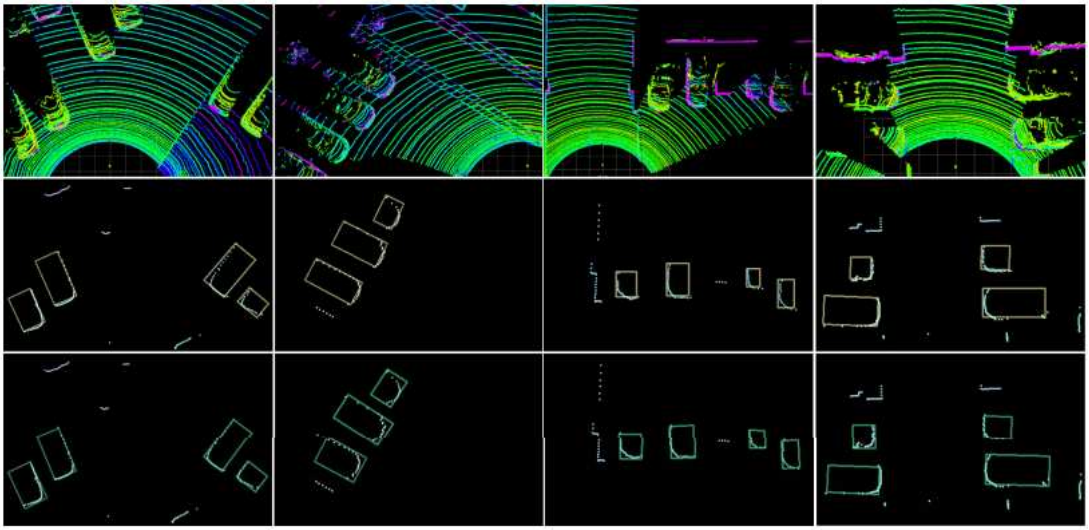
2.4 TRABALHOS RELACIONADOS

Sarker, Qingqing e Westerlund (2020) destacam a limitação inerente à utilização de uma única nuvem de pontos de um sensor LiDAR 2D para a condução de um veículo autônomo. Isso se deve ao fato de que tais dados fornecem uma perspectiva restrita da cena. Em situações onde existem obstáculos irregulares, como uma mesa, por exemplo, a disposição do sensor pode se tornar crucial, já que dependendo da sua posição, o robô ou veículo em questão, poderia não detectar mais do que os pés da mesa, ficando vulnerável as colisões ou encalhamentos na parte superior desse tipo de objeto.

Apesar das limitações, Chen et al. (2021a) implementaram dois métodos diferentes, baseados em *deep learning*, que ofereceram uma alternativa a utilização de câmeras e sensores 3D, para a detecção de veículos, de forma robusta e eficiente, e tendo um baixo custo. Mostraram seus resultados consistentes comparando aos dados do sensor 3D como mostra a Figura 9.

Horváth, Pozna e Unger (2022) demonstram a coleta e processamento de dados de um sensor LiDAR 3D em tempo real. A entrada do experimento são dados do sensor, sem uma câmera ou qualquer outro sensor adicional e a saída, é uma nuvem de pontos voxel 3D da estrada e da calçada (Figura 10), juntamente com uma descrição poligonal 2D da rua, sendo esse resultado útil para outros algoritmos, como

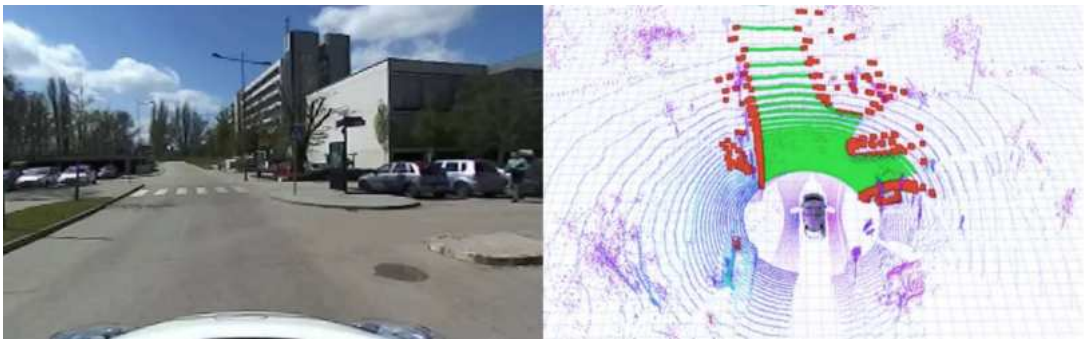
Figura 9 – As imagens superiores mostram os dados do sensor 3D para fins de visualização, as imagens do meio mostram as caixas de referência dos veículos em nuvens de pontos 2D em amarelo e as imagens inferiores mostram as caixas de predição em verde feitas pelo algoritmo implementado.



Fonte: Chen et al. (2021a, p. 7707)

planejadores de caminhos. A solução apresentada inclui três métodos diferentes de detecção de calçadas, um método de detecção de estradas e uma extração de estradas baseada em polígonos 2D.

Figura 10 – Dados coletados de um sensor LiDAR destacando a estrada e as calçadas comparados a imagem real do local no veículo teste utilizado.



Fonte: Horváth, Pozna e Unger (2022, p. 11)

Banerjee et al. (2018) escolheram pelo uso da fusão de informações de uma câmera junto ao sensor LiDAR para detectar as calçadas das vias. Essa abordagem vem crescendo no segmento automobilístico, sendo bastante utilizada devido aos resultados apresentados. A primeira parte do artigo mostra a importância da calibração dos equipamentos para trabalharem em sintonia, agregando valor as informações captadas em conjunto. O estudo é finalizado com os resultados de detecção de objetos utilizando diferentes técnicas na fusão dos dados da câmera junto ao sensor.

Observa-se que diferentes técnicas e objetivos na detecção de objetos num meio podem ser empregados com o uso de um sensor como esse. Ressaltando assim, o papel crucial que sensores LiDAR, e sistemas de detecção no geral, desempenham nos avanços tecnológicos da direção autônoma.

3 MATERIAIS E MÉTODOS

Neste capítulo será apresentado o processo realizado para a concepção do projeto visando a conclusão dos objetivos determinados. Portanto, a primeira etapa trata do planejamento do projeto e levantamento de requisitos, levando a montagem do sistema para a aquisição de dados de um sensor LiDAR, e seguindo com a implementação de diferentes métodos para a detecção de objetos.

Requisitos Funcionais:

- RF001 - O sistema deve detectar objetos utilizando somente os dados do LiDAR;
- RF002 - O sistema deve fornecer a distância média do objeto em relação ao sensor;
- RF003 - O sistema deve fornecer a distância do ponto mais próximo do objeto em relação ao sensor;
- RF004 - O sistema deve ser capaz de detectar objetos dinamicamente;
- RF005 - O sistema deve ser capaz de detectar objetos em movimento;
- RF006 - O sistema deve ter uma interface gráfica para visualização dos objetos detectados;

Requisitos Não-Funcionais:

- RNF001 - A interface gráfica deve mostrar os objetos detectados rotulados por cores e suas respectivas distâncias;
- RNF002 - O sistema de detecção deve ter um tempo de resposta médio abaixo de 100 milissegundos;
- RNF003 - O sensor LiDAR deve fazer leituras em um raio de distância de pelo menos 5 metros em 360°;
- RNF004 - O sistema deve utilizar o sensor RPlidar A1M8 da SLAMTEC.

3.1 SISTEMA

Os materiais utilizados durante o desenvolvimento e avaliação do sistema proposto foram:

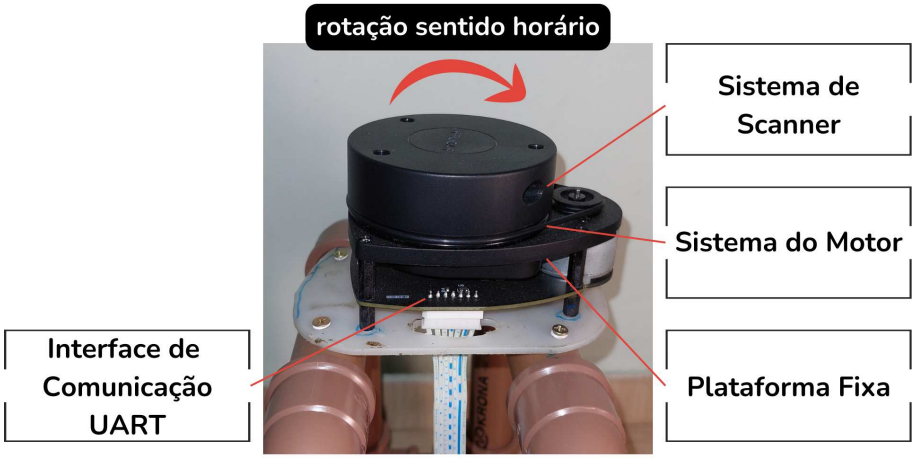
- Notebook Dell Inspiron 15 da série 3000;
- Sensor RPlidar A1M8 SLAMTEC;
- Adaptador UART para USB SLAMTEC;
- Cabo USB para micro USB;
- Tubos e conexões de PVC;
- Trena comum de 5 metros.

O sensor RPlidar A1M8 (Figura 6), é um sensor de baixo custo desenvolvido pela SLAMTEC, empresa chinesa especializada no desenvolvimento e na produção de tecnologia de localização e navegação de robôs autônomos, sensores e dispositivos

de internet das coisas (SLAMTEC, 2023). De acordo com seu *datasheet*, esse sensor faz leituras 360 graus com até 6 metros de distância, com uma precisão de erro menor que 0,5% para distâncias medidas a até um metro e meio do sensor. Esse sensor faz leituras 360 graus com até 6 metros de distância, satisfazendo os requisitos de projeto. A frequência média de operação do sensor é de 5.5Hz para amostrar 360 pontos por rodada, podendo atingir até 10Hz de frequência máxima (SLAMTEC, 2020).

O RPlidar A1 vem com um sistema adaptativo e de detecção de velocidade, ou seja, o sistema pode ajustar a frequência do *scanner* automaticamente conforme a velocidade de rotação do motor (SLAMTEC, 2020). A Figura 11 mostra a composição completa do sistema do sensor.

Figura 11 – Sistema que compõe o RPlidar A1M8.



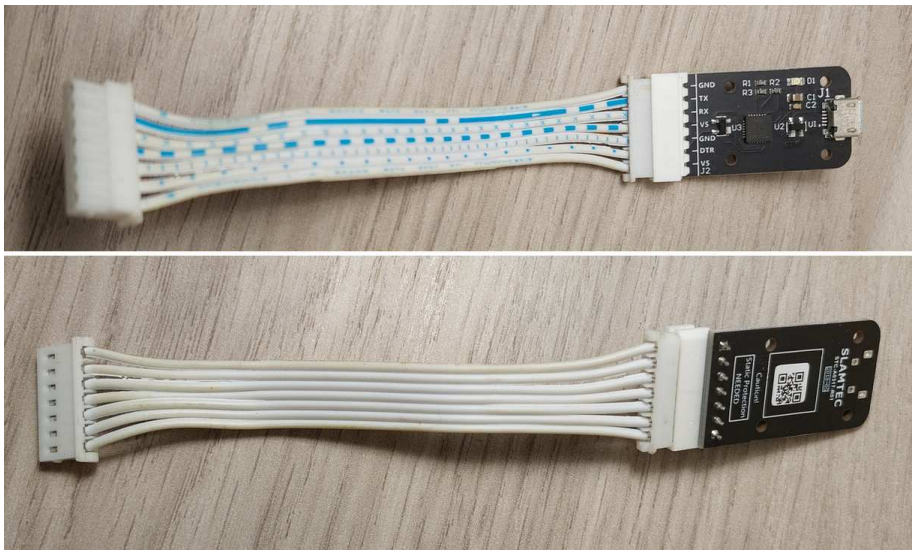
Fonte: Autora (2023).

A interface de comunicação desse sensor é composta por uma porta serial 3.3V-TTL, utilizando o protocolo UART (*Universal Asynchronous Receiver/Transmitter*). Um conversor UART para conexão USB (*Universal Serial Bus*) foi utilizado, também fornecido pela SLAMTEC (Figura 12). O uso desse conversor possibilita a coleta direta de dados do sensor em um notebook.

O ideal para os testes, é que o sensor estivesse a uma altura mediana do solo, se assemelhando assim, a altura da parte frontal de um veículo em que o sistema de detecção de objetos desenvolvido, pudesse ser empregado. Portanto, foram tiradas as medidas da parte frontal do kart elétrico do Laboratório de Sistemas Embarcados (LSE) da Universidade Federal de Santa Catarina (UFSC), do Campus Joinville, que futuramente, pode ser o veículo a utilizar esse sistema (Figura 13).

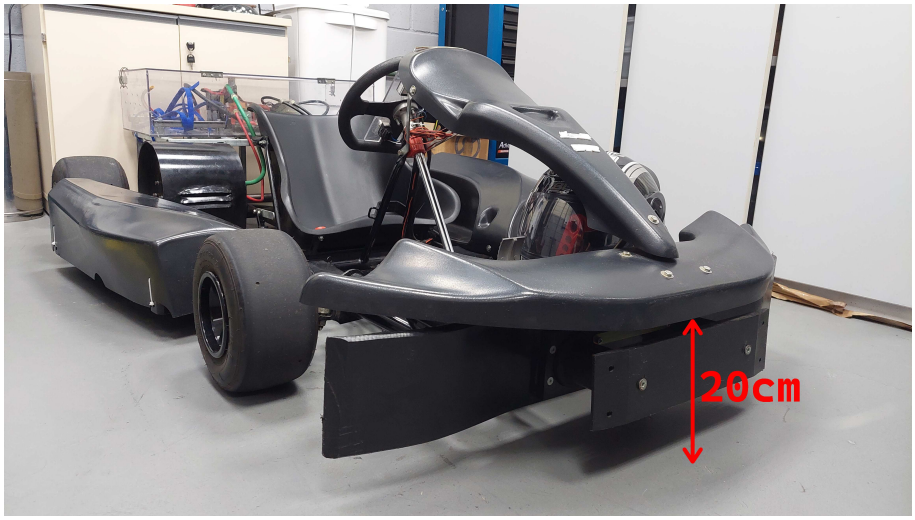
Enfim, para que o sensor ficasse na altura aproximada da parte de cima do para-choque do kart, onde tem espaço para a fixação do sensor, e mantivesse a estabilidade nas leituras, foi confeccionado um suporte feito de tubos PVC de aproximadamente 19

Figura 12 – Conversor UART para USB.



Fonte: Autora (2023)

Figura 13 – Parte frontal do kart elétrico do LSE.



Fonte: Autora (2023)

centímetros de altura e 17 centímetros de largura, como mostra a Figura 14.

3.2 IMPLEMENTAÇÃO

O projeto foi implementado utilizando o Oracle VirtualBox versão 7.0.10, integrado ao sistema operacional Windows 11, permitindo a execução de uma instância virtual do sistema Linux Ubuntu 22.04.2 LTS. O ambiente de desenvolvimento foi estabelecido no Visual Studio Code versão 1.84.0, onde todos os códigos foram implementados em C++.

A SLAMTEC fornece um SDK público de código aberto, implementado em

Figura 14 – Sensor acoplado ao suporte confeccionado.

Fonte: Autora (2023)

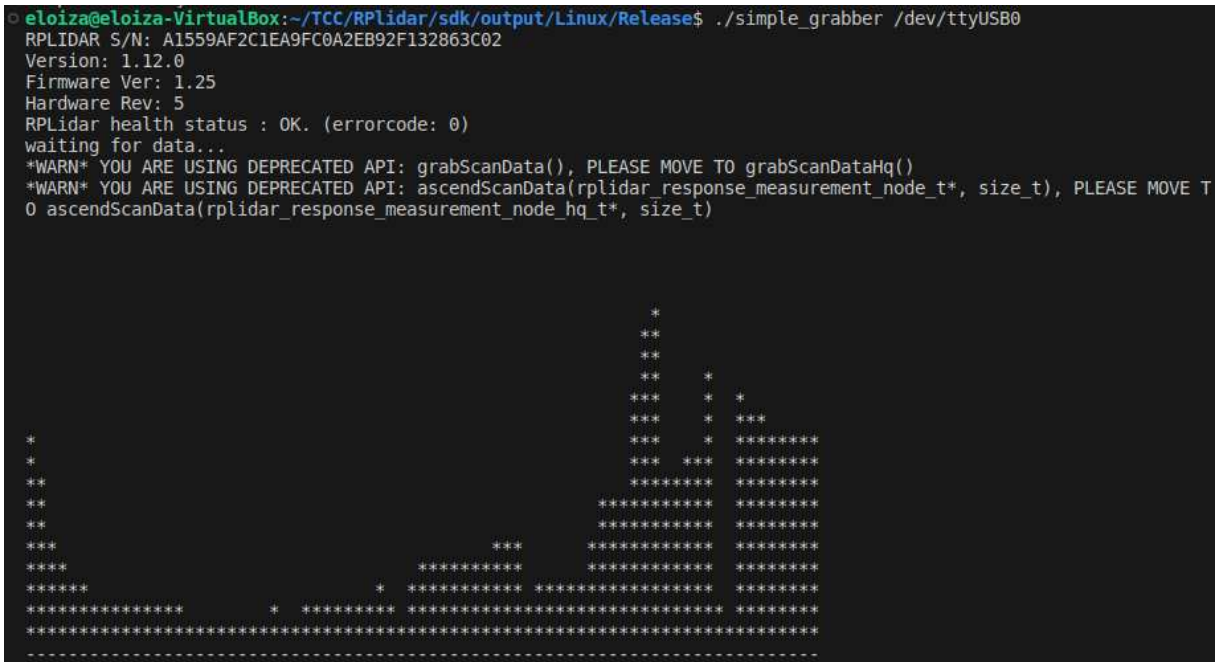
C++[1], para uso dos sensores RPlidar. O SDK está disponível em RPlidar SDK (2019). Para fazer a validação geral de funcionamento do sensor LiDAR, a SLAMTEC fornece junto ao SDK, algumas aplicações prontas para testes. Nesse caso, foi utilizada a aplicação denominada *simple grabber*. Primeiramente, ela tem como saída, o número de série do LiDAR, a versão do *firmware* e o status de integridade do sensor após a conexão com o *notebook*. Em seguida, a aplicação de demonstração obtém uma rodada de dados de varredura e mostra os dados de alcance como histograma (Figura 15). Se o usuário desejar, os ângulos e distâncias da leitura são mostrados também.

Para melhor entender os dados fornecidos pelo sensor, foi implementado um código baseando-se nas aplicações já existentes do SDK e em sua documentação. O objetivo do código era salvar em um arquivo de texto, os dados gerados diretamente do sensor, ou seja, os ângulos em graus, e suas respectivas distâncias em milímetros, dos alvos de uma leitura 360^o do sensor (Apêndice A).

Para que a nuvem de pontos formada pela leitura do sensor ficasse visivelmente mais fácil de ser avaliada, um ambiente parcialmente controlado, ou seja, com suas paredes delimitadas e com menos objetos presentes foi montado (Figura 16). Como é

[1] dialeto de 2011

Figura 15 – Saída do terminal na execução da aplicação *simple grabber*.



Fonte: Autora (2023)

possível observar na imagem, o único objeto presente, além das delimitações gerais do espaço, é a banqueta em que o notebook está apoiado.

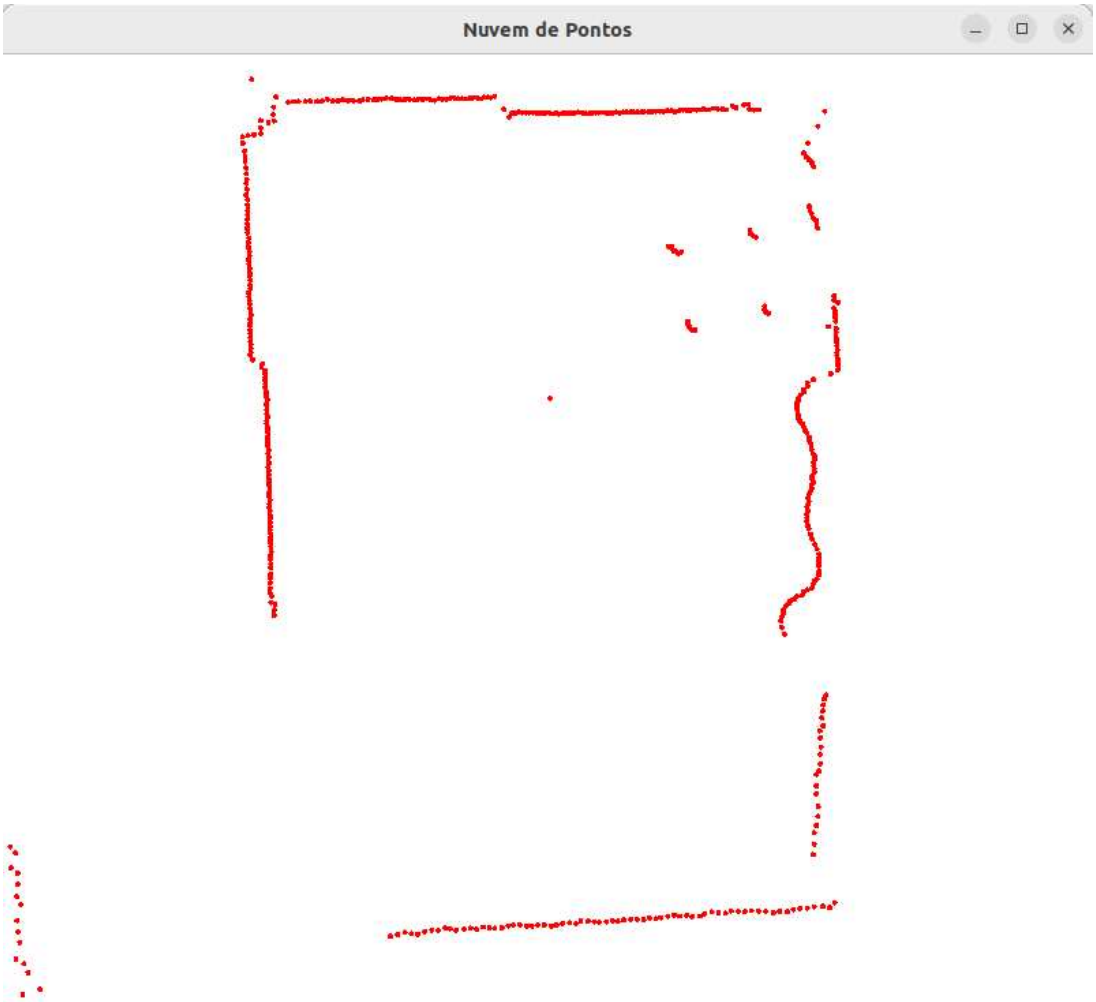
Figura 16 – Ambiente para os primeiros testes de leitura 360^o do sensor.

Fonte: Autora (2023)

Portanto, utilizando o código citado anteriormente, uma leitura desse ambiente

foi feita e salva. Foi então implementado um código para fazer a leitura desse arquivo e formar uma nuvem 2D composta por pontos com coordenadas cartesianas x e y. A nuvem é salva em um arquivo de texto para fins de comparação e avaliação (Apêndice B), e plotada utilizando a biblioteca gráfica SFML (Figura 17). A biblioteca *Simple and Fast Multimedia Library* (SFML) é uma biblioteca de código aberto em C++ que fornece uma variedade de recursos multimídia (SFML, 2023).

Figura 17 – Nuvem de pontos 2D.

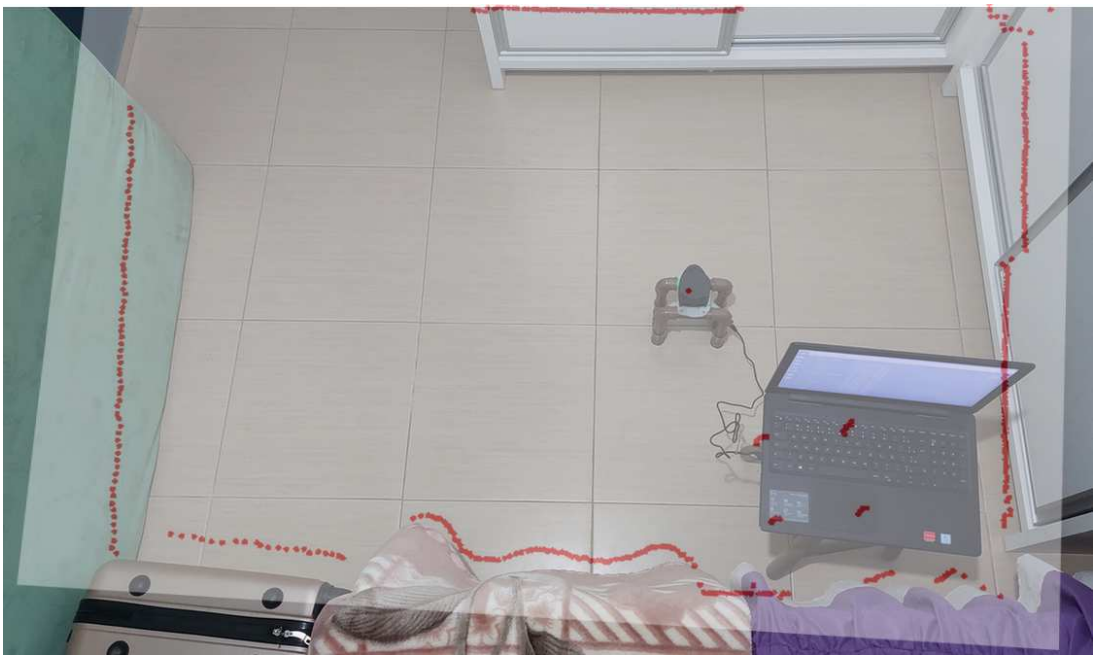


Fonte: Autora (2023)

A Figura 18 mostra que o formato da nuvem de pontos segue exatamente o local em que a leitura foi feita, ou seja, no geral, a leitura funcionou da maneira que deveria. Há algumas divergências na imagem, mas isso acontece devido a diferente angulação em que a leitura é feita pelo sensor, e o ângulo que a foto foi tirada.

A partir da validação dessa nuvem de pontos de maneira superficial, foi iniciada a implementação da detecção dos objetos. Inicialmente o objetivo foi detectar os objetos de forma estática, utilizando essas leituras salvas em arquivo de texto, e posteriormente,

Figura 18 – Nuvem de pontos 2D sobreposta no ambiente real em que a leitura foi feita.



Fonte: Autora (2023)

foi incorporada à leitura dinâmica em tempo real do sensor.

Nesse trabalho foram propostos dois métodos diferentes para detecção de objetos, um método simples utilizando diretamente os dados que o sensor dispõe, e um método com maior tratamento dos dados utilizando uma biblioteca própria para lidar com nuvem de pontos. A seguir serão detalhadas as implementações de cada método baseando-se na aplicação deles de forma estática para fins de análise.

As técnicas implementadas para detecção de objetos são dinâmicas, e elas implementam as mesmas técnicas utilizadas de modo estático. A aplicação de ambos os métodos de forma dinâmica, tem como principal diferença comparada a forma estática, a obtenção dos dados da nuvem de pontos. Dinamicamente, a cada ciclo da janela de visualização da biblioteca SFML, uma nova leitura 360^o do sensor é feita, e o método de detecção de objetos em questão é aplicado. Ou seja, a cada 100 ms aproximadamente, uma nova análise no ambiente é feita.

3.2.1 Método de Detecção Proposto

O método implementado pela autora foi pensado baseado na utilização direta dos dados advindos do sensor LiDAR, o qual fornece distância e ângulo de cada ponto em relação ao sensor. Portanto, a nuvem de pontos, nesse caso, não segue com coordenadas cartesianas e sim, com coordenadas de ângulos em graus e distâncias em milímetros. Pensando na estrutura desejada para um sistema de navegação veicular, e que o ideal é ter algumas informações sobre a localização do objeto. Foram implementadas duas classes, a classe *Object*, para manter a estrutura dos objetos que

forem detectados, e a classe *Ponto2D* que representa os pontos que constituem os objetos (Figura 19).

Figura 19 – Classes dos objetos e pontos 2D.

```
6   class Object
7   {
8   public:
9       Object();
10      Ponto2D getPontoMedio();
11      Ponto2D getClosestPoint();
12      void addPoint(double distance, double angle);
13      Ponto2D getLastPoint();
14
15      std::vector<Ponto2D> points;
16  };
```

(a) Classe de objetos.

```
4   class Ponto2D
5   {
6   public:
7       Ponto2D(double coord_x, double coord_y, double distancia, double angulo);
8       double x;
9       double y;
10      double distance;
11      double angle;
12  };
```

(b) Classe de pontos 2D.

Fonte: Autora (2023)

Para obter mais facilmente as informações dos pontos de cada objeto, é necessário que o ponto seja constituído não só da sua distância e ângulo, informações as quais são fornecidas diretamente pela nuvem de pontos desse método, como também, as coordenadas cartesianas em que os pontos estão. Para satisfazer essa necessidade, a função *addPoint* trata os dados, fazendo o cálculo das duas informações faltantes, coordenadas x e y, antes de criar o ponto de fato (Figura 20).

Figura 20 – Função que cria um ponto 2D e o adiciona ao objeto.

```
36  void Object::addPoint(double angle, double distance)
37  {
38      double anguloRadianos = rad_angle(angle);
39
40      double x = distance * std::cos(anguloRadianos);
41      double y = distance * std::sin(anguloRadianos);
42      Ponto2D point(x, y, angle, distance);
43      points.push_back(point);
44  }
```

Fonte: Autora (2023)

O principal destaque no código desse método, é a função *pontoPertenceAoObjeto* (Figura 21), que é onde de fato, o método foi implementado. A função recebe como parâmetros um ponto da nuvem e um objeto. Então, para verificar

se esse ponto da nuvem pertence ao objeto em questão, são definidas duas variáveis que representam o limite máximo que o ponto pode ter, para mais ou para menos, em ângulo e distância, do último ponto do objeto. Se o ponto comparado satisfaz tais requisitos, a função retorna *true* confirmando que o mesmo faz parte do objeto, se não, retorna *false*.

Figura 21 – Função que define se um ponto pertence a determinado objeto.

```
49  // Funcao para verificar se um ponto pertence a um objeto
50  bool pontoPertenceAoObjeto(const std::pair<double, double> &ponto, Object objeto)
51  {
52      // limite para angulo e distancia
53      double limiteAngulo = 5.0;      // graus
54      double limiteDistancia = 100.0; // mm
55
56      // pegar o ultimo ponto da lista so ja que os angulos sao em ordem
57      Ponto2D ultimoPonto = objeto.getLastPoint();
58      double diferencaAngulo = std::abs(ponto.first - ultimoPonto.angle);
59      double diferencaDistancia = std::abs(ponto.second - ultimoPonto.distance);
60
61      // o ponto pertence ao objeto se seu angulo e +-limiteAngulo do que o ultimo ponto da lista do objeto
62      // e sua distancia é +-limiteDistancia do que a distancia do ultimo ponto da lista do objeto
63      if (diferencaAngulo <= limiteAngulo && diferencaDistancia <= limiteDistancia)
64      {
65          return true; // O ponto pertence a esta lista
66      }
67
68      return false; // O ponto não pertence a esta lista
69  }
```

Fonte: Autora (2023)

Ressalta-se que nesse caso, somente o último ponto da lista do objeto é comparado, e isso é devido à nuvem gerada sempre estar com os ângulos em ordem crescente, ou seja, em ordem de leitura. Consequentemente, os pontos são adicionados aos objetos, também em ordem crescente. Importante também, observar as unidades de medida adotadas. A informação gerada pelo SDK do sensor, e salva pela nuvem de pontos desse método, contém o ângulo em graus e a distância em milímetros. Portanto, os limites estabelecidos para comparação, seguem as mesmas unidades.

A função *detectObjects* (Figura 22) utiliza a função mostrada anteriormente para verificar ponto a ponto da nuvem. Se o ponto pertence a determinado objeto, a função *addPoint* é chamada, para adicioná-lo ao objeto em questão. Nessa mesma função, *detectObjects*, é aplicado o único filtro contra possíveis ruídos do sensor utilizado nesse método, que é uma verificação na qual se a distância do ponto for igual a zero, o ponto é ignorado e a verificação se ele pertence a algum objeto não é feita, passando assim, para o próximo ponto da nuvem. Essa verificação foi implementada por conta da observação durante os testes, de alguns ruídos presentes nas leituras do sensor, onde a distância fica com valor zero, como é possível observar no Apêndice A.

Por fim, iterando objeto a objeto identificado, são criados os pontos para visualização da nuvem utilizando a biblioteca SFML. Os primeiros testes foram feitos

Figura 22 – Função que verifica todos os pontos da nuvem e retorna um vetor com os objetos detectados.

```
71  // funcao para que retorna lista de objetos detectados na nuvem
72  std::vector<Object> detectObjects(const std::vector<std::pair<double, double>> &cloud)
73  {
74      std::vector<Object> objects;
75      for (const auto &ponto : cloud)
76      {
77          bool pontoPertence = false;
78
79          if (ponto.second == 0)
80          {
81              continue;
82          }
83
84          // Iterar sobre a lista de objetos
85          for (auto &objeto : objects)
86          {
87              if (pontoPertenceAoObjeto(ponto, objeto))
88              {
89                  pontoPertence = true;
90                  objeto.addPoint(ponto.first, ponto.second); // Adicione o ponto à lista
91                  break;
92              }
93          }
94
95          if (!pontoPertence)
96          {
97              Object new_object;
98              new_object.addPoint(ponto.first, ponto.second);
99              objects.push_back(new_object);
100         }
101     }
102
103     return objects;
104 }
```

Fonte: Autora (2023)

com uma plotagem simples, identificando os objetos detectados somente com uma *label* enumerando cada um deles, Figura 23(a), não existindo ainda a estrutura de objetos com as informações de fácil acesso como apresentada na classe *Objects*. Após melhor estruturação do código, facilitando o acesso às informações de cada objeto, foi obtido o ponto médio de cada objeto detectado, Figura 23(b).

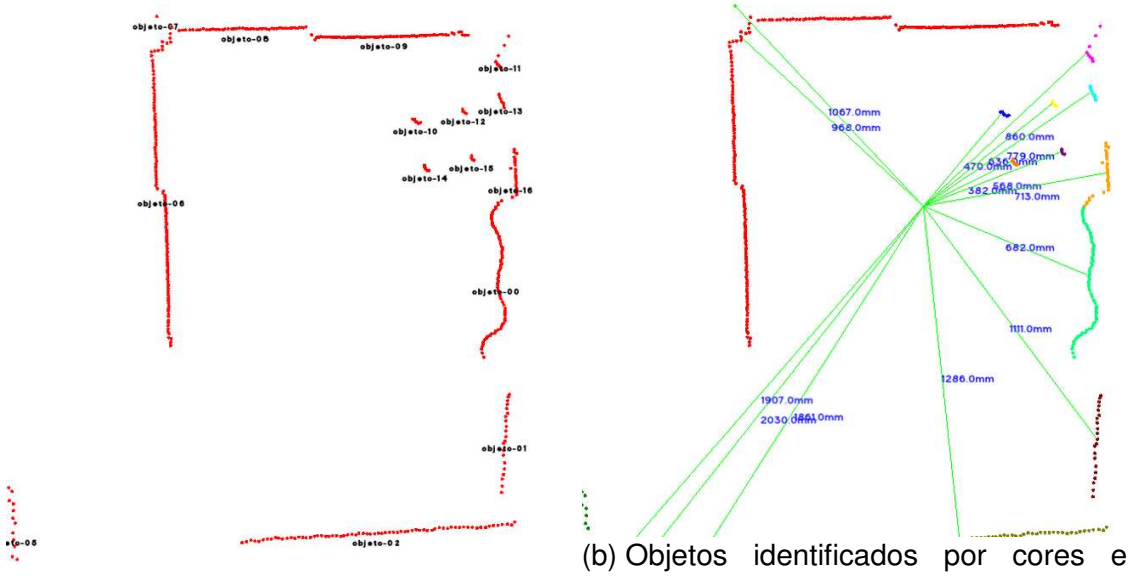
3.2.2 Método de Detecção Utilizando Biblioteca PCL

Point Cloud Library (PCL), é uma biblioteca em C++, *open source*, para processamento de imagens 2D/3D e nuvens de pontos. A biblioteca contém algoritmos de última geração para filtragem, estimativa de recursos, reconstrução de superfícies, ajustes e segmentação. E é apoiada por uma comunidade internacional de pesquisadores de robótica e de percepção (RUSU; COUSINS, 2011).

À diferença do método anterior, a classe da nuvem de pontos PCL, é formada em coordenadas cartesianas. Portanto, fazendo a leitura do arquivo de texto com ângulos e distâncias, se faz necessária a conversão de ponto a ponto para coordenadas com x, y e z (Figura 24). Como o sensor gera somente pontos em duas dimensões e a biblioteca PCL foi implementada para suportar nuvens de três dimensões, a coordenada do eixo z, foi considerada uma constante de valor zero.

Para reutilizar a estrutura de objetos e pontos criada para o método anterior e

Figura 23 – Evolução nas saídas do código de detecção de objetos do método proposto.



(a) Objetos identificados por texto.

(b) Objetos identificados por cores e distância média.

Fonte: Autora (2023)

Figura 24 – Criação da nuvem PCL baseada no arquivo de texto gerado com leitura 360°.

```
41    // Cria um objeto PCL PointCloud
42    pcl::PointCloud<pcl::PointXYZ>::Ptr nuvemPCL(new pcl::PointCloud<pcl::PointXYZ>);
43
44    std::string linha;
45    // Le o arquivo linha por linha
46    while (std::getline(arquivo, linha))
47    {
48        std::istringstream iss(linha);
49
50        // Tenta extrair os valores de angulo e distancia de cada linha
51        if (!(iss >> angulo >> distancia))
52        {
53            std::cerr << "Erro ao ler linha do arquivo." << std::endl;
54            continue;
55        }
56
57        // Converte o angulo para radianos
58        double anguloRadianos = angulo * M_PI / 180.0;
59
60        // Calcula os valores de x, y e z com base na distancia, angulo e z constante
61        double x = distancia * cos(anguloRadianos);
62        double y = distancia * sin(anguloRadianos);
63        double z = 0.0; // Valor constante de z
64
65        pcl::PointXYZ pontoPCL;
66        pontoPCL.x = x;
67        pontoPCL.y = y;
68        pontoPCL.z = z;
69        nuvemPCL->points.push_back(pontoPCL);
70    }
```

Fonte: Autora (2023)

unificar a forma com que as informações dos objetos são armazenadas, foi feita uma adaptação no código. A função *addPoint* foi separada em duas funções diferentes, a primeira mantendo a implementação necessária para o método proposto e tendo somente seu nome modificado para *addPointWithDistance*, e a segunda, sendo uma

nova função chamada *addPointWithCoordinates* (Figura 25). A nova função tem o mesmo objetivo da primeira função implementada, ou seja, fazer o cálculo das informações faltantes, sendo nesse caso a distância e o ângulo, para que então seja criado o ponto e adicionado ao objeto.

Figura 25 – Função para adicionar pontos ao objeto utilizando coordenadas cartesianas.

```
46  void Object::addPointWithCoordinates(double coord_x, double coord_y)
47  {
48      double angle = std::atan2(coord_y, coord_x) * 180.0 / M_PI;
49      double distance = sqrt(std::pow(coord_x, 2) + std::pow(coord_y, 2));
50
51      Ponto2D point(coord_x, coord_y, angle, distance);
52      points.push_back(point);
53  }
```

Fonte: Autora (2023)

A biblioteca PCL oferece vários recursos prontos de fácil utilização, portanto, em casos onde a nuvem de pontos é mais densa, em três dimensões e em ambientes com muita informação, ela pode oferecer vantagens de processamento. Para os casos de uso desse projeto, não se é exigido tantos recursos, visto que a nuvem de pontos 2D não é tão densa e não contém tantos ruídos, então, foram utilizados dois recursos de filtragem da PCL.

O primeiro filtro foi implementado utilizando o tipo *PointIndices*, destinado a armazenar índices de pontos de uma nuvem. Nesse caso, foi usado para armazenar os índices de pontos que estejam a mais de seis metros do sensor. Após a obtenção desses índices, foi criado um objeto do tipo *ExtractIndices*, utilizado para extração de pontos de uma determinada nuvem. Esse objeto foi configurado para ter como entrada a nuvem em questão e os índices armazenados anteriormente, tendo como saída, a nuvem sem os pontos desses índices (Figura 26). Esse filtro foi aplicado visto que segundo as especificações do sensor, ele é capaz de fazer leituras de até seis metros (SLAMTEC, 2020), portanto, pontos podem ser considerados ruídos.

Figura 26 – Extraindo pontos da nuvem que estejam a mais de 6 metros do sensor.

```
84      // Armazena indices de pontos de nuvem PCL
85      pcl::PointIndices::Ptr inliers(new pcl::PointIndices);
86
87      int count = 0;
88      for (int i = 0; i < nuvemPCL->size(); i++)
89      {
90          if (abs(nuvemPCL->points[i].x) > 6000 || abs(nuvemPCL->points[i].y) > 6000)
91          {
92              inliers->indices.push_back(i); // pega indices de pontos que estejam a mais de 6 metros do sensor
93              count++;
94          }
95      }
96      pcl::ExtractIndices<pcl::PointXYZ> extract; // cria um objeto do tipo extract indices, ele eh usado pra tirar pontos de uma nuvem pcl
97      extract.setInputCloud(nuvemPCL->makeShared());
98      extract.setIndices(inliers);
99      extract.setNegative(true);
100     extract.filter(*nuvemPCL);
```

Fonte: Autora (2023)

O segundo filtro foi utilizando os recursos de voxels. A biblioteca PCL dispõe do tipo *VoxelGrid*, um filtro que transforma a nuvem de pontos, agrupando-a em uma grade de voxels visando reduzir sua densidade (Figura 27). Após a filtragem, a nuvem é transformada novamente para uma nuvem comum da biblioteca PCL, a qual será usada para a detecção de objetos, e é salva em um arquivo do tipo *Point Cloud Data* (PCD), que compõe todas as informações necessárias a respeito da nuvem e seus pontos (Apêndice C).

Figura 27 – Diminuindo a densidade da nuvem utilizando um filtro voxel.

```
102
103     pcl::PCLPointCloud2::Ptr cloud2(new pcl::PCLPointCloud2()); // tipo necessário para processamento do filtro voxel
104     pcl::toPCLPointCloud2(*nuvemPCL, *cloud2);                  // converte a nuvem recebida em um tipo PCLPointCLoud2
105     pcl::VoxelGrid<pcl::PCLPointCloud2> filtro;                 // esse filtro tem o objetivo de agrupar a nuvem em voxels
106     filtro.setInputCloud(cloud2);
107     filtro.setLeafSize(0.05f, 0.05f, 0.05f);
108     filtro.filter(*cloud2);
109     pcl::fromPCLPointCloud2(*cloud2, *nuvemPCL);
110
111     // Save cloud to pcd file -  salvando para fins de analise e comparacao
112     pcl::io::savePCDFileASCII("nuvemPCL", *nuvemPCL);
113 }
```

Fonte: Autora (2023)

Tendo a nuvem filtrada, um vetor de objetos é criado, e a função *detectObjects* é chamada para preencher esse vetor com os objetos detectados (Figura 28). Como a nuvem da biblioteca PCL é toda em coordenadas cartesianas, o método para detectar objetos, ou *clusters*, é baseado em distância Euclidiana. Então, um objeto do tipo *EuclideanClusterExtraction* é criado e configurado conforme desejado. A configuração consiste na tolerância que se deseja em relação à distância máxima de um ponto para outro para serem considerados do mesmo objeto e a quantidade mínima e máxima de pontos que um objeto pode ter para ser criado.

A biblioteca PCL fornece a estrutura de dados de *kd-tree*, fornecida pela FLANN, uma biblioteca também escrita em linguagem C++, utilizada para realizar pesquisas rápidas e aproximadas do vizinho mais próximo em espaços de alta dimensão (MUJA; LOWE, 2009). O objeto *EuclideanClusterExtraction* permite que seja configurado o método de buscas dos *clusters*, portanto, essa estrutura de *kd-tree* foi utilizada.

A nuvem filtrada anteriormente é a entrada do método, e um vetor *PointIndices* contendo *clusters* com os índices de seus respectivos pontos, é a saída. Tendo os índices dos pontos separados em objetos, a informação das coordenadas desses pontos é retirada da nuvem e os objetos do tipo *Object* são criados.

Figura 28 – Função utilizada para detectar os objetos usando PCL.

```
112 void detectObjects(pcl::PointCloud<pcl::PointXYZ>::Ptr &nuvemPCL, std::vector<Object> &objects)
113 {
114
115     // estrutura de dados usada para pesquisas eficientes em nuvens de pontos
116     pcl::search::KdTree<pcl::PointXYZ>::Ptr tree(new pcl::search::KdTree<pcl::PointXYZ>);
117
118     pcl::EuclideanClusterExtraction<pcl::PointXYZ> ext_clusters; // objeto usado pra extrair grupos de pontos próximos na nuvem
119     ext_clusters.setClusterTolerance(100);                       // configura a dist máx que os pontos podem ter um do outro
120     ext_clusters.setMinClusterSize(2);                           // cluster tem que ter no mínimo 2 pontos
121     ext_clusters.setMaxClusterSize(500);                         // clusters tem que ter no maximo 500 pontos
122
123     // Seta a nuvem lida e filtrada como entrada da KD Tree
124     tree->setInputCloud(nuvemPCL->makeShared());
125
126     // Extrai clusters
127     std::vector<pcl::PointIndices> cluster_indices;
128     ext_clusters.setSearchMethod(tree);
129     ext_clusters.setInputCloud(nuvemPCL->makeShared());
130     ext_clusters.extract(cluster_indices);
131
132     for (auto cluster_indice : cluster_indices)
133     {
134         Object new_object;
135         // Vai passando indice por indice dos pontos de cada cluster detectado e formando os objetos
136         for (const auto &indice : cluster_indice.indices)
137         {
138             pcl::PointXYZ point;
139             point.x = (*nuvemPCL)[indice].x;
140             point.y = (*nuvemPCL)[indice].y;
141             point.z = (*nuvemPCL)[indice].z;
142
143             new_object.addPointWithCoordinates(point.x, point.y);
144         }
145         objects.push_back(new_object);
146     }
147 }
```

Fonte: Autora (2023)

4 RESULTADOS E DISCUSSÕES

Neste capítulo serão analisados os resultados apresentados por dois cenários de testes em que foram utilizados os métodos de detecção desenvolvidos nesse projeto. Todos os resultados foram obtidos das medições feitas dinamicamente. Serão detalhadas comparações de algumas medições e discutidos os resultados. Por fim, serão avaliadas as limitações do sistema.

4.1 CENÁRIO 1

O primeiro cenário de teste teve como principal objetivo a comparação das medições de distância dos objetos. O ambiente foi formado utilizando dois objetos de formatos simples e fácil detecção, em um espaço fechado de aproximadamente 2,40 metros por 1,70 metros (Figura 29).

Figura 29 – Cenário de teste 1.

Fonte: Autora (2023)

No caso do método proposto, foram configurados como 10 centímetros e 5 graus, os limites de distância e ângulo, respectivamente, que o ponto pode ter do último ponto da lista do objeto para ser considerado parte do objeto em questão. Para o

método utilizando a PCL, o limite imposto foi de que um ponto pode ter no máximo 10 centímetros de distância Euclidiana um do outro para serem considerados parte de um mesmo objeto. Experimentos empíricos foram realizados a fim de chegar nos valores de limites estabelecidos.

Como é possível observar na Figura 30, nesse caso, a principal diferença dos dois métodos é a divisão de um dos lados do ambiente em duas partes de parede, detectando um objeto a mais. Fato esse, que não teria impactos quando se tratando de um sistema de navegação veicular. Isso é o resultado da configuração dos parâmetros de limitação dos pontos a serem considerados de um mesmo objeto que cada método traz. Esses parâmetros podem ser alterados conforme necessário.

Para as comparações de medições de distâncias, foi utilizada a distância média dos objetos para a plotagem, visto que fazendo as medições com uma trena, é mais fácil medir a partir do centro do objeto em questão (Tabela 1).

Tabela 1 – Distâncias médias dos objetos em relação ao sensor no cenário 1

Objeto	Método Proposto	Método PCL	Trena
Caixa	1048 mm	1047 mm	1041 mm
Garrafa	776 mm	775 mm	777 mm

Fonte: Autora (2023).

A partir da medida das distâncias da Tabela 1, é possível fazer algumas comparações visando entender se as medições fazem sentido com as especificações do *datasheet* do sensor. Portanto, foram calculados os erros percentuais da medição dos dois métodos em relação a medição da trena, utilizada como valor nominal, em cada um dos objetos, como mostra a Tabela 2.

Tabela 2 – Erro percentual das medidas de distância do cenário 1

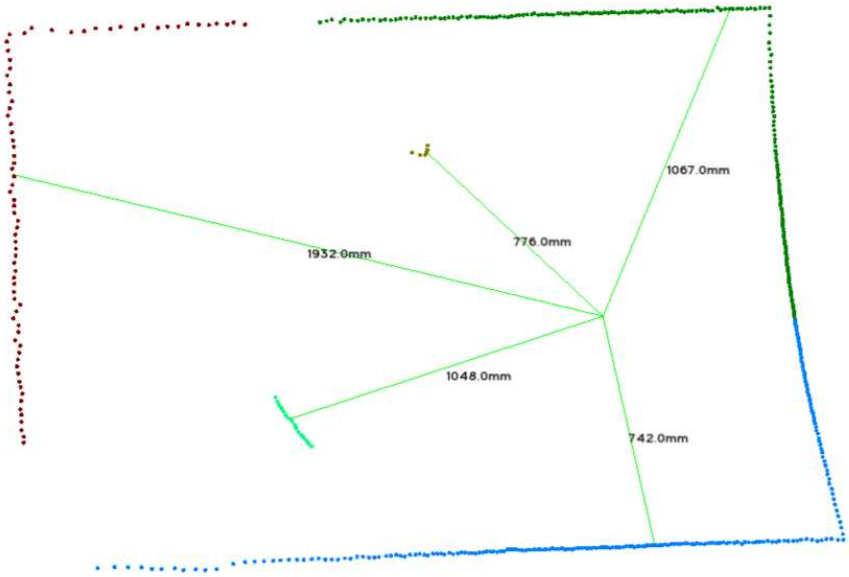
Objeto	Método Proposto	Método PCL
Caixa	0,67%	0,58%
Garrafa	0,13%	0,26%

Fonte: Autora (2023).

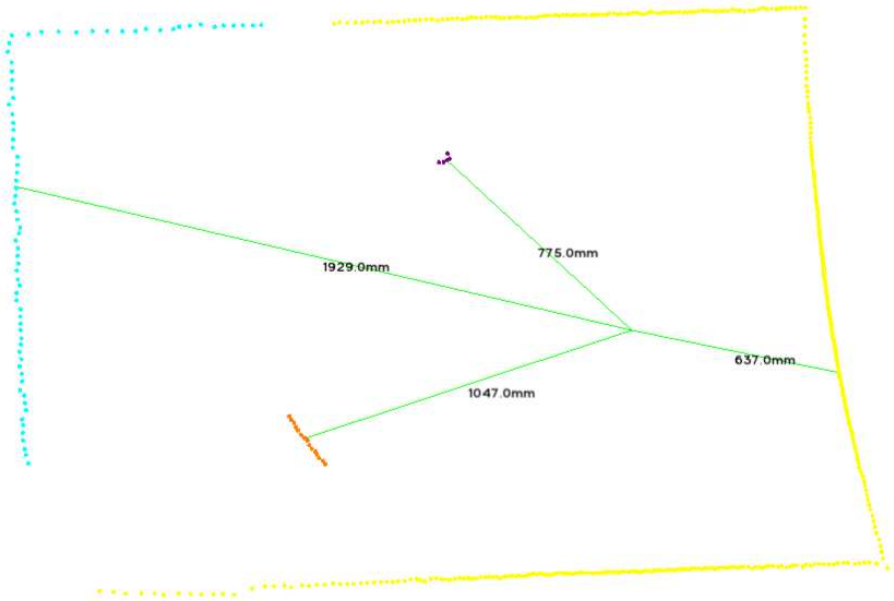
O *datasheet* do sensor especifica o desempenho das medições de distância de duas formas. Uma métrica para medições abaixo de um metro e meio de distância, sendo essa de um erro de menos de 0,5% da distância medida, e uma métrica para objetos a mais de um metro e meio de distância do sensor, com um erro de menos de 1% da distância medida (SLAMTEC, 2020).

Observa-se que os erros calculados, apesar de não estarem de acordo com as especificações do sensor, estão bem próximos. Levando em consideração que a

Figura 30 – Objetos detectados no cenário 1.



(a) Objetos detectados método proposto.



(b) Objetos detectados utilizando PCL.

Fonte: Autora (2023)

trena não passou por uma aferição relevante, e que o ponto médio dos objetos pelos métodos propostos fora retirado a partir do ponto mediano do *array* de pontos do objeto, necessitam-se estudos futuros para uma conclusão sobre a precisão.

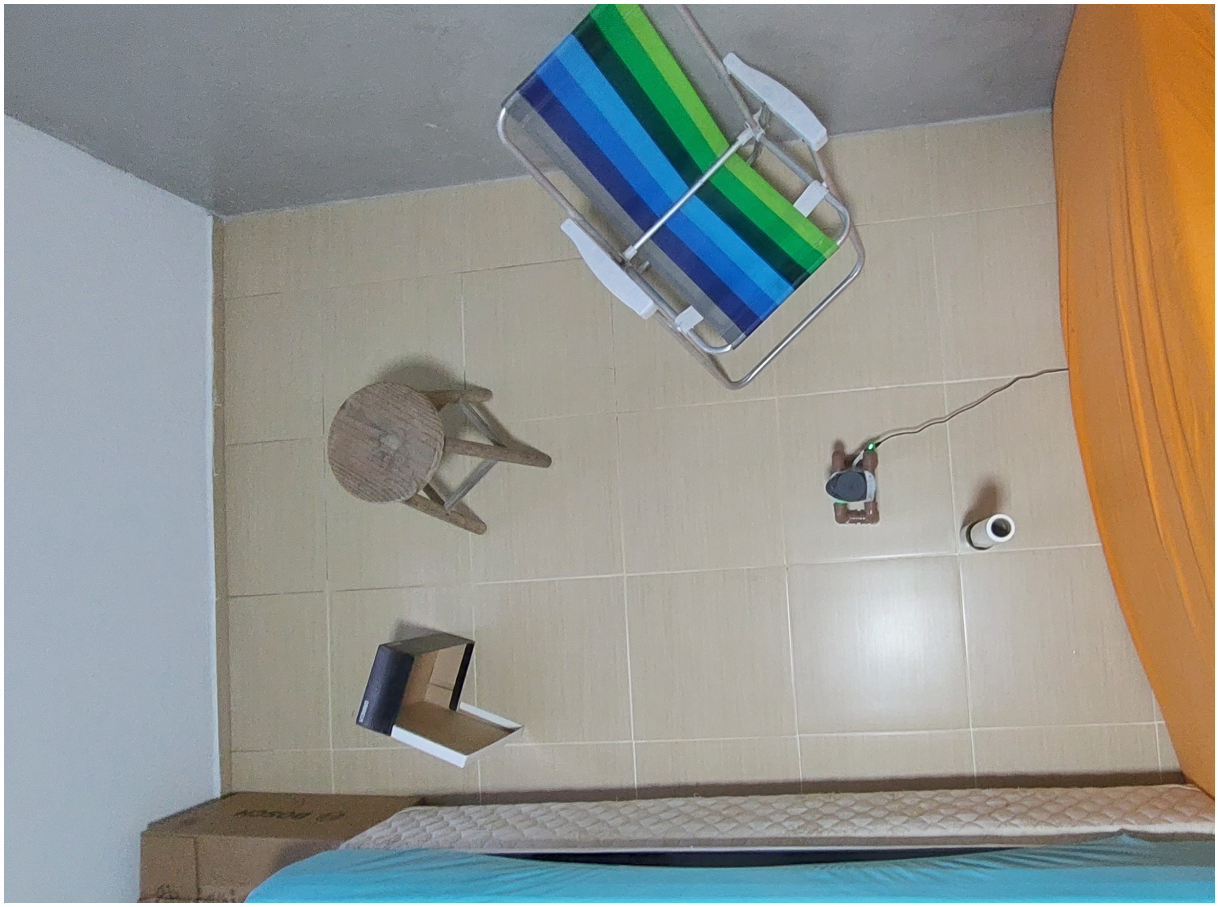
Os sistemas de navegação veicular precisam também, ter um tempo de resposta adequado para as tomadas de decisões. Pensando nisso, foram medidos os tempos que cada método demora para fazer a leitura da nuvem, processá-la e

detectar os objetos (Apêndice D). Utilizando os dados de tempo coletados e calculando sua média, o método proposto apresenta um tempo de resposta médio de 98,075 milissegundos, enquanto o método utilizando PCL apresentou um tempo de resposta médio de 97,75 milissegundos.

4.2 CENÁRIO 2

Para o segundo cenário, o principal enfoque foi testar o desempenho de detecção de objetos dos métodos implementados. O ambiente utilizado foi o mesmo, porém, incluindo uma maior quantidade de objetos de diferentes formatos e que possam desafiar as limitações do sistema proposto (Figura 31).

Figura 31 – Cenário de teste 2.



Fonte: Autora (2023)

As configurações de filtragem e a respeito dos limites de angulação e distâncias para os pontos serem agrupados como objetos foram mantidas as mesmas, para ambos os métodos. Como é possível observar na Figura 32, nesse cenário ocorreram diferenças visíveis na detecção dos objetos. Essas diferenças estão principalmente nos objetos menores, como os apoios da cadeira de praia, e os apoios da banqueta. O

método proposto detectou todos eles, enquanto o método usando PCL não conseguiu detectar alguns deles, principalmente os da cadeira, que são ainda mais finos.

Para as comparações de medições de distâncias, assim como no cenário anterior, foi utilizada a distância média dos objetos (Tabela 3).

Tabela 3 – Distâncias médias dos objetos em relação ao sensor no cenário 2

Objeto	Método Proposto	Método PCL	Trena
Caixa	1261 mm	1258 mm	1280 mm
Garrafa	297 mm	296 mm	295 mm
Cadeira Apoio 1	482 mm	483 mm	480 mm
Cadeira Apoio 2	652 mm	650 mm	645 mm
Cadeira Apoio 3	888 mm	-	895 mm
Cadeira Apoio 4	769 mm	-	775 mm
Banqueta Apoio 1	885 mm	888 mm	885 mm
Banqueta Apoio 2	1053 mm	1050 mm	1055 mm
Banqueta Apoio 3	1174 mm	-	1180 mm
Banqueta Apoio 4	1026 mm	1027 mm	1035 mm

Fonte: Autora (2023).

A partir da medição das distâncias, foram calculados os erros percentuais dos métodos, para cada um dos objetos do ambiente em questão, como mostra a Tabela 4.

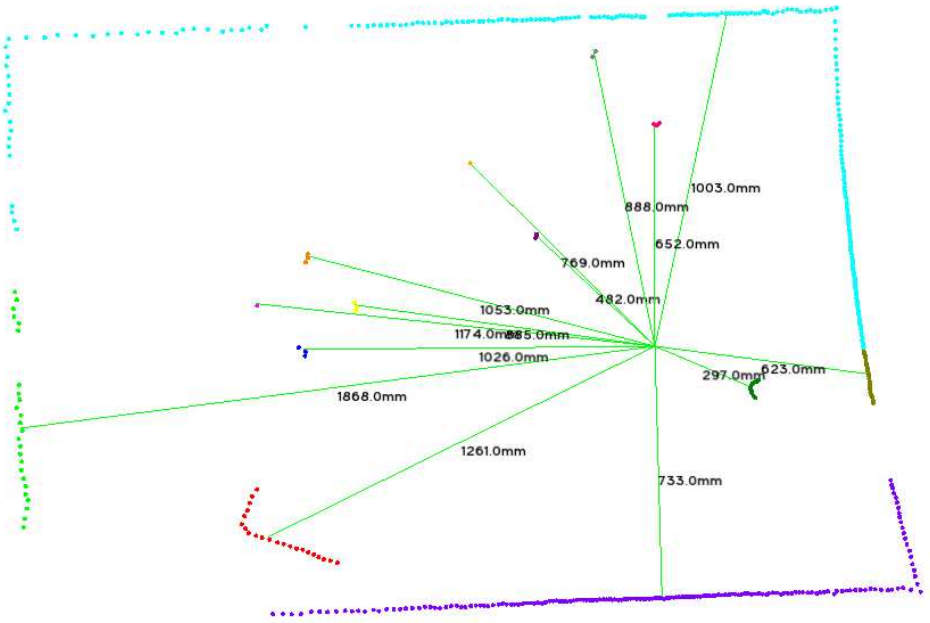
Tabela 4 – Erro percentual das medidas de distância do cenário 2

Objeto	Método Proposto	Método PCL
Caixa	1,48%	1,72%
Garrafa	0,68%	0,34%
Cadeira Apoio 1	0,42%	0,63%
Cadeira Apoio 2	1,09%	0,78%
Cadeira Apoio 3	0,78%	-
Cadeira Apoio 4	0,77%	-
Banqueta Apoio 1	0%	0,34%
Banqueta Apoio 2	0,19%	0,47%
Banqueta Apoio 3	0,51%	-
Banqueta Apoio 4	0,87%	0,77%

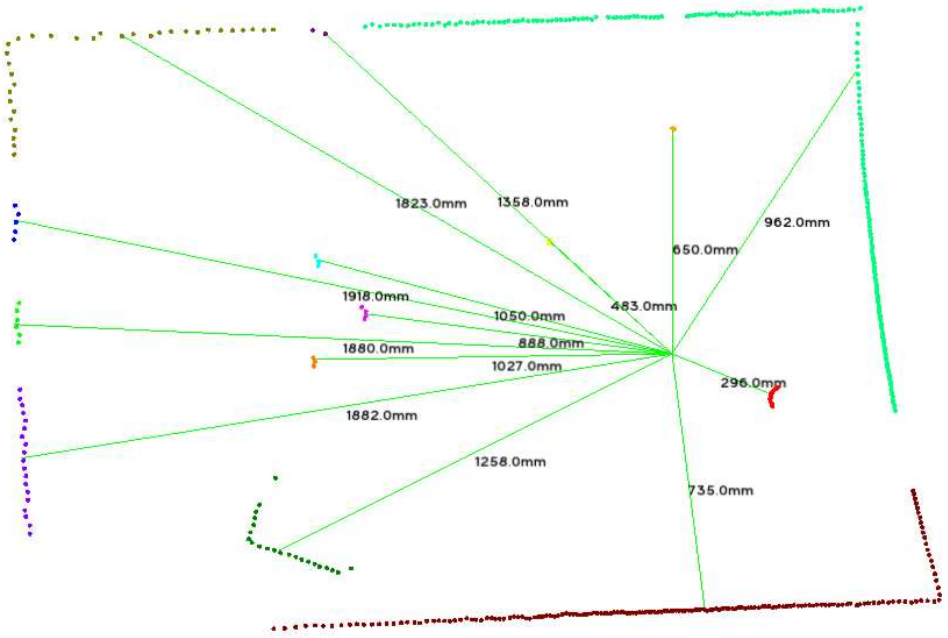
Fonte: Autora (2023).

Analisando os valores apresentados, é notável que os erros de medida da caixa, ficaram muito discrepantes em relação aos outros. Porém, esse erro pode ser justificado, pois a medição como ponto médio do objeto com a trena, foi feita no ponto de canto onde começa a tampa da caixa, que está aberta. Se analisar a nuvem de pontos da Figura 32, o ponto médio considerado pelo sistema não é, na verdade, naquele ponto, constatando um erro na distância nominal utilizada nesse caso.

Figura 32 – Objetos detectados no cenário 2



(a) Objetos detectados método proposto.



(b) Objetos detectados utilizando PCL.

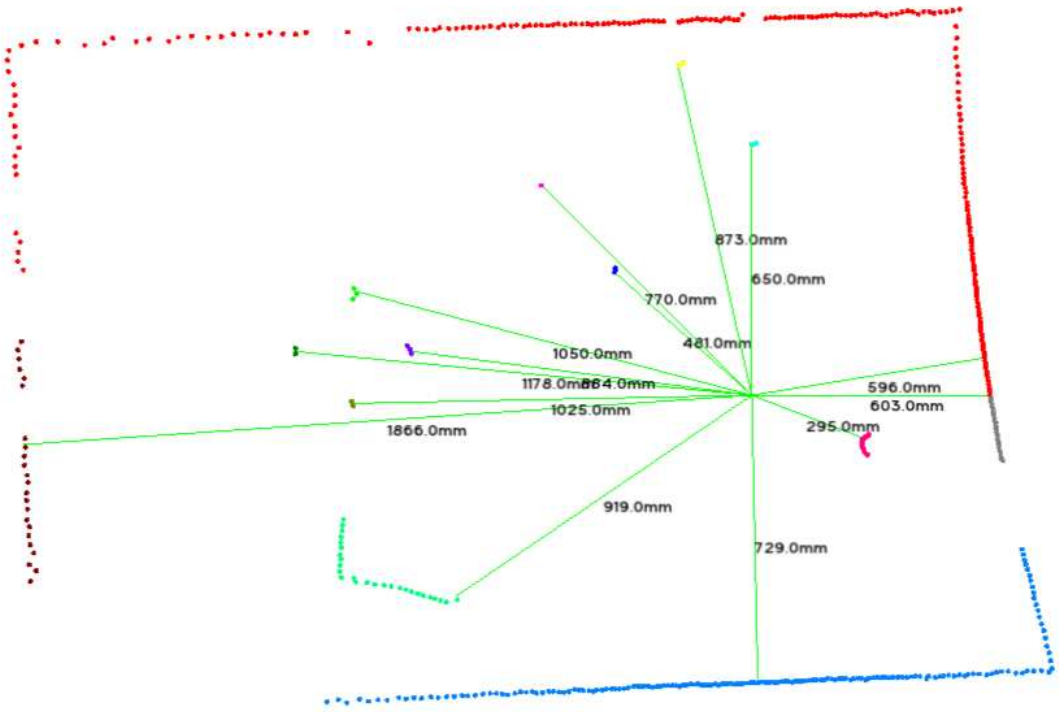
Fonte: Autora (2023)

4.3 DISCUSSÕES

Dados os resultados experimentais, é possível fazer comparações mais claras em relação aos métodos apresentados. Em ambos cenários de testes, pode-se comprovar que as distâncias medidas pelo sensor são confiáveis e apresentam um erro pequeno.

Importante ressaltar, que as medidas de distâncias utilizadas para as comparações, foram as distâncias médias dos objetos, apenas pela facilidade em medir com a trena o ponto médio para fins de análise. Entretanto, o importante para o sistema proposto, tendo como enfoque um sistema de navegação veicular, é a captação do ponto com menor distância do objeto em relação ao sensor, para assim, evitar colisões e afins (Figura 33).

Figura 33 – Plotagem com a distância do ponto mais próximo do objeto em relação ao sensor utilizando o método proposto.



Fonte: Autora (2023)

Em relação aos tempos de resposta, o método utilizando a biblioteca PCL apresentou um tempo minimamente abaixo do método proposto, porém, ambos satisfizeram o requisito ficando abaixo de 100 ms.

Nos aspectos da detecção dos objetos de fato, o método PCL apresentou maior dificuldades em detectar objetos pequenos e irregulares. Isso pode estar relacionado

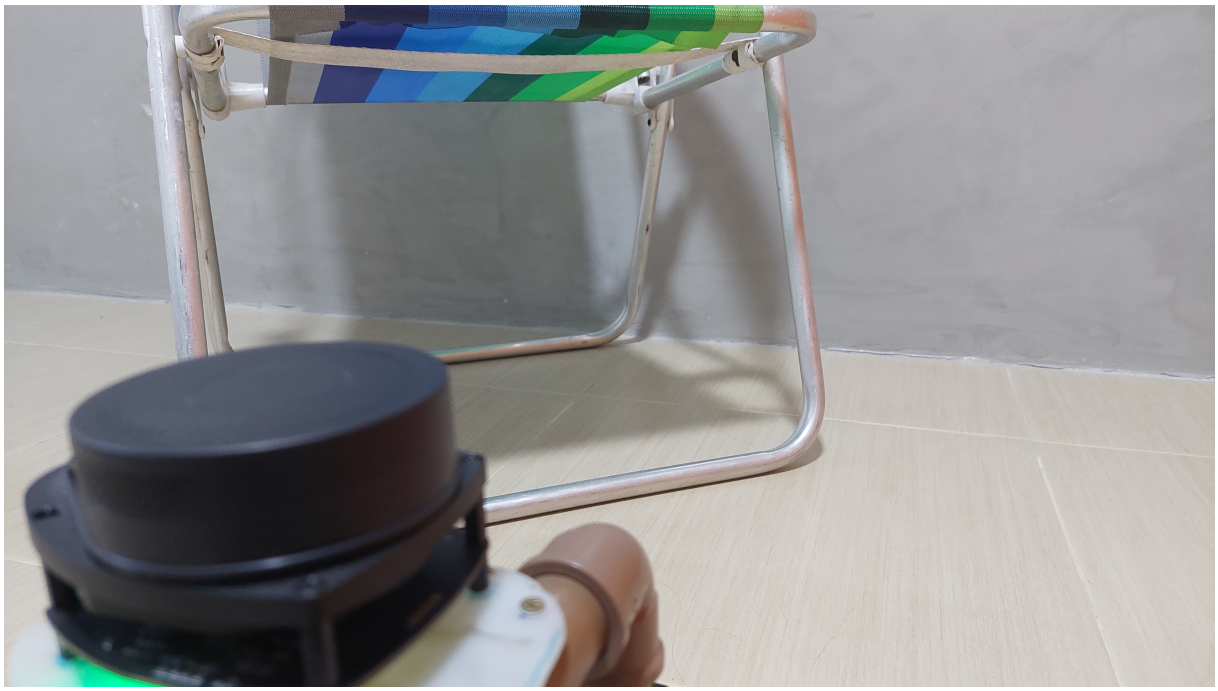
ao uso de uma maior filtragem no método com a PCL, enquanto no método proposto a nuvem é quase a leitura original feita pelo sensor, no método com a PCL, a nuvem é transformada em um agrupamento de voxels para diminuir a sua densidade, o que apesar de diminuir o tempo de processamento, pode causar uma perda de informações.

4.4 LIMITAÇÕES

O segundo cenário de teste traz uma perspectiva sobre a principal limitação do sistema proposto. Como destacado por Sarker, Qingqing e Westerlund (2020) e apresentado na seção de fundamentação teórica desse trabalho (Capítulo 2), a nuvem de pontos em duas dimensões tem suas limitações.

A cadeira de praia do teste mostra que em casos como esse, onde o objeto é irregular, o veículo utilizando um sistema de detecção de objetos como o desenvolvido nesse projeto teria problemas, e provavelmente, ocorreria uma colisão. A Figura 34 mostra que apesar do sensor só detectar as quatro hastes de apoio da cadeira que estão em seu campo de alcance, ainda há os apoios horizontais paralelos ao chão, e a parte de cima dela. Desse modo, exemplificando a principal limitação desse trabalho.

Figura 34 – "Visão" do sensor LiDAR em relação à cadeira de praia.



Fonte: Autora (2023)

5 CONCLUSÕES

O objetivo desse trabalho é a detecção de objetos utilizando um sensor LiDAR, com enfoque da aplicação em sistemas de navegação veicular, visto a importância da percepção ambiental para tais sistemas. Para a fundamentação teórica do projeto, foram revisitadas as diversas possibilidades de sensoriamento utilizados em veículos fazendo o papel de percepção. Foram citados também, os diferentes tipos de sensores LiDAR e as diferentes técnicas aplicadas a eles, que se mostram uma tecnologia valiosa e crescente nas pesquisas dessa área de estudo.

O sistema foi desenvolvido com a utilização de um sensor 2D. Tal fato, trouxe mais desafios ao projeto, visto que os recursos para lidar com nuvem de pontos 2D são mais limitados. Mesmo com as limitações, o objetivo do projeto foi concluído, sendo implementados dois métodos diferentes para a detecção de objetos.

Um método foi proposto, mantendo uma certa simplicidade e utilizando diretamente os dados fornecidos pelo sensor para a detecção dos objetos. E o outro método, foi construído com a utilização dos recursos fornecidos pela biblioteca PCL, que oferece diversas possibilidades de tratamento de nuvem de pontos. Ambos os métodos apresentaram bons resultados em ambientes controlados, mesmo com objetos em movimento. A validação foi feita por meio de comparações de medições manuais da distância média dos objetos em relação ao sensor, com as medições apresentadas pelo sistema.

Por fim, a principal limitação apresentada, está relacionada aos resultados da detecção de objetos com formas irregulares. No contexto da percepção ambiental, o alcance vertical limitado do sensor gera desafios significativos nesse aspecto. Há inúmeros caminhos a serem seguidos para contornar essa limitação. Portanto, o sistema apresentado possui oportunidades para melhorias que podem ser exploradas em futuros trabalhos, como, por exemplo:

- Buscar uma solução para a detecção dos objetos com formatos irregulares verticalmente no uso do sensor 2D;
- A migração dos métodos propostos para o processamento em uma Raspberry Pi e a aplicação em um sistema de navegação veicular;
- A comparação de desempenho em um sistema de navegação veicular, dos métodos propostos com um método utilizando um sensor LiDAR 3D;
- A fusão dos métodos propostos com o uso de visão computacional utilizando uma câmera, possibilitando também a identificação dos objetos.

REFERÊNCIAS

BANERJEE, K. et al. Online camera lidar fusion and object detection on hybrid data for autonomous driving. In: **2018 IEEE Intelligent Vehicles Symposium (IV)**. Changshu, China: IEEE, 2018. p. 1632–1638. Disponível em: https://ieeexplore.ieee.org/document/8500699. Acesso em: 11 jun. 2023.

BENEDEK, C. et al. Positioning and perception in lidar point clouds. **Digital Signal Processing**, v. 119, p. 103193, 2021.

BHASKARAN, S. **Direct detection time of flight lidar sensor system design and a vortex tracking algorithm for a doppler lidar**. 2018. Dissertation (PhD Thesis in Mechanical Engineering) — Arizona State University, 2018.

BLACK, P. E. **Euclidean distance**. Dictionary of Algorithms and Data Structures, 2004. Disponível em: https://www.nist.gov/dads/HTML/euclidndstnc.html. Acesso em: 25 out. 2023.

BLACK, P. E. **k-d tree**. Dictionary of Algorithms and Data Structures, 2020. Disponível em: https://xlinux.nist.gov/dads/HTML/kdtree.html. Acesso em: 25 out. 2023.

CHEN, G. et al. Pseudo-image and sparse points: Vehicle detection with 2d lidar revisited by deep learning-based methods. **IEEE Transactions on Intelligent Transportation Systems**, v. 22, n. 12, p. 7699–7711, 2021.

CHEN, Q. et al. Sensing system of environmental perception technologies for driverless vehicle: A review of state of the art and challenges. **Sensors and Actuators A: Physical**, v. 319, p. 112566, 2021.

COLIN, J.; FAIVRE, R. Aerodynamic roughness length estimation from very high-resolution imaging lidar observations over the heihe basin in china. **Hydrology and Earth System Sciences**, v. 14, n. 12, p. 2661–2669, 2010.

DICKMANN, J. et al. Radar contribution to highly automated driving. In: **Proceedings of 2014 11th European Radar Conference**. IEEE, 2014. p. 412–415. Disponível em: https://ieeexplore.ieee.org/document/6991295. Acesso em: 16 set. 2023.

DICKMANN, J. et al. "automotive radar the key technology for autonomous driving: From detection and ranging to environmental understanding". In: **2016 IEEE Radar Conference (RadarConf)**. IEEE, 2016. p. 1–6. Disponível em: https://ieeexplore.ieee.org/document/7485214. Acesso em: 24 out. 2023.

DINGYI, Y.; HAIYAN, W.; KAIMING, Y. State-of-the-art and trends of autonomous driving technology. In: **Proceedings of the 2018 IEEE INTERNATIONAL SYMPOSIUM ON INNOVATION AND ENTREPRENEURSHIP - TEMS-ISIE**. IEEE, 2018. p. 1–8. Disponível em: https://ieeexplore.ieee.org/document/8478449. Acesso em: 10 jun. 2023.

DUBé, R. et al. Detection of parked vehicles from a radar based occupancy grid. In: **2014 IEEE Intelligent Vehicles Symposium Proceedings**. IEEE, 2014. p. 1415–1420. Disponível em: https://ieeexplore.ieee.org/document/6856568. Acesso em: 24 out. 2023.

ESPECTRO ELETROMAGNÉTICO. **NASA**. 2015. Disponível em: https://www.nasa.gov/audience/forstudents/k-4/dictionary/Electromagnetic_Spectrum.html. Acesso em: 08 jun. 2023.

HORVÁTH, E.; POZNA, C.; UNGER, M. Real-time lidar-based urban road and sidewalk detection for autonomous vehicles. **Sensors**, v. 20, n. 194, p. 1–17, 2022.

IEEE SPECTRUM. 2011. Disponível em: https://spectrum.ieee.org/how-google-self-driving-car-works. Acesso em: 24 out. 2023.

JAMET, C. et al. Going beyond standard ocean color observations: Lidar and polarimetry. **Frontiers in Marine Science**, v. 6, n. 251, p. 1–24, 2019.

KADLEC, E. A. et al. Coherent lidar for autonomous vehicle applications. In: **2019 24th OptoElectronics and 2019 International Conference on Photonics in Switching and Computing (PSC)**. Fukuoka, Japan: IEEE, 2019. p. 1–3. Disponível em: https://ieeexplore.ieee.org/document/8478449. Acesso em: 10 jun. 2023.

LEEUWEN, M. van; NIEUWENHUIS, M. Retrieval of forest structural parameters using lidar remote sensing. **European Journal of Forest Research**, v. 1, n. 129, p. 749–770, may 2010.

LIN, Y. et al. Toward better boundary preserved supervoxel segmentation for 3d point clouds. **ISPRS Journal of Photogrammetry and Remote Sensing**, v. 143, p. 39–47, 2018.

MCMANAMON, P. F. **LiDAR Technologies and Systems**. Bellingham, Washington USA: SPIE, 2019.

MUJA, M.; LOWE, D. G. Fast approximate nearest neighbors with automatic algorithm configuration. In: **International Conference on Computer Vision Theory and Application VISSAPP'09)**. INSTICC Press, 2009. p. 331–340. Disponível em: https://www.cs.ubc.ca/research/flann/uploads/FLANN/flann_visapp09.pdf. Acesso em: 02 nov. 2023.

NIU, J. et al. Robust lane detection using two-stage feature extraction with curve fitting. **Pattern Recognition**, v. 59, p. 225–233, 2016.

PAPON, J. et al. Voxel cloud connectivity segmentation - supervoxels for point clouds. In: **2013 IEEE Conference on Computer Vision and Pattern Recognition**. IEEE, 2013. p. 2027–2034. Disponível em: https://ieeexplore.ieee.org/document/6619108. Acesso em: 25 out. 2023.

PAREKH, D. et al. A review on autonomous vehicles: Progress, methods and challenges. **Electronics**, v. 11, n. 14: 2162, p. 1–18, 2022.

PENDLETON, S. D. et al. Perception, planning, control, and coordination for autonomous vehicles. **Machines**, v. 5, n. 1:6, p. 1–54, 2017.

PERIU C.F., A. M. C. L.; MCLAUGHLIN, N. Isolation of vibrations transmitted to a lidar sensor mounted on an agricultural vehicle to improve obstacle detection. **Canadian Biosystems Engineering**, v. 55, n. 1, p. 2.33–2.42, dec 2013.

RPLIDAR SDK. **SLAMTEC**. 2019. Disponível em: https://github.com/Slamtec/rplidar_sdk. Acesso em: 18 out. 2023.

RUSU, R. B.; COUSINS, S. 3d is here: Point cloud library (pcl). In: **2011 IEEE International Conference on Robotics and Automation**. IEEE, 2011. p. 1–4. Disponível em: https://ieeexplore.ieee.org/document/5980567. Acesso em: 19 out. 2023.

SARKER, V. K.; QINGQING, L.; WESTERLUND, T. 3d perception with low-cost 2d lidar and edge computing for enhanced obstacle detection. In: **2020 IEEE Conference on Industrial Cyberphysical Systems (ICPS)**. IEEE, 2020. p. 49–54. Disponível em: https://ieeexplore.ieee.org/document/9274771. Acesso em: 25 set. 2023.

SFML. **SFML**. 2023. Disponível em: https://github.com/SFML/SFML. Acesso em: 18 out. 2023.

SIVARAMAN, S. **Learning, modeling, and understanding vehicle surround using multi-modal sensing**. 2013. Dissertation (PhD Thesis in Electrical Engineering) — Uinversity of California, 2013.

SLAMTEC. **Datasheet RPlidar A1M8**. Shanghai, China, 2020. Disponível em: https://bucket-download.slamtec.com/d1e428e7efbdcd65a8ea111061794fb8d4ccd3a0/LD108_SLAMTEC_rplidar_datasheet_A1M8_v3.0_en.pdf. Acesso em: 18 out. 2023.

SLAMTEC. **SLAMTEC**. 2023. Disponível em: https://www.slamtec.ai/home/about/. Acesso em: 18 out. 2023.

TUDOR, E. et al. Lidar sensors used for improving safety of electronic-controlled vehicles. In: **2021 12th International Symposium on Advanced Topics in Electrical Engineering (ATEE)**. Bucharest, Romania: IEEE, 2021. p. 1–5. Disponível em: https://ieeexplore.ieee.org/document/9425123. Acesso em: 10 jun. 2023.

WANG, J.; SUN, H.; ZHU, C. Vision-based autonomous driving: A hierarchical reinforcement learning approach. **IEEE Transactions on Vehicular Technology**, p. 1–15, 2023.

WANG, Y. et al. Traffic sign attack via pinpoint region probability estimation network. **Pattern Recognition**, p. 110035, 2023.

XING, L.; PENG, F.; PAN, R. Research on the method of environmental perception based on millimeter wave radar. In: **Proceedings of the 2022 3rd China International SAR Symposium (CISS)**. IEEE, 2022. p. 1–4. Disponível em: https://ieeexplore.ieee.org/document/9971289. Acesso em: 16 set. 2023.

ZHAO, Y.; ZHANG, X.; HUANG, X. A technical survey and evaluation of traditional point cloud clustering methods for lidar panoptic segmentation. **2021 IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)**, p. 2464–2473, 2021.

ZHU, H. et al. Overview of environment perception for intelligent vehicles. **IEEE Transactions on Intelligent Transportation Systems**, v. 18, n. 10, p. 2584–2601, 2017.

APÊNDICE A - ARQUIVO DE TEXTO GERADO A PARTIR DA LEITURA DO SENSOR NO FORMATO "ÂNGULO DISTÂNCIA"

```
0.015625 607
0.5625 606
1.21875 603
2.01562 604
2.54688 605
3.32812 606
4 608
4.65625 609
5.32812 614
5.85938 621
6.51562 625
7.1875 629
7.73438 634
8.51562 640
9.17188 642
9.70312 645
10.375 649
11.1562 651
11.8281 656
12.4844 663
13.0156 663
13.6875 662
14.3438 665
15.0156 668
15.6719 671
16.4688 668
17.125 670
17.6719 675
18.3438 673
19 676
19.6875 675
20.3438 674
21.0156 676
21.6719 680
22.2188 682
22.8906 682
23.5469 688
```

```
24.3438 689
24.8906 692
25.5625 696
26.2188 701
26.7656 708
27.5625 718
28.2188 727
28.8906 738
29.4375 749
30.1094 760
30.7656 764
31.3125 770
31.9844 773
32.7656 781
33.3125 786
34.1094 788
34.6562 785
35.3125 787
35.9844 790
36.6562 784
37.3281 779
37.9844 781
38.6719 773
39.4375 773
40.1094 768
40.6562 769
41.3125 768
42.1094 769
42.6562 775
43.3125 778
43.9844 784
44.6562 796
45.1875 812
46.2031 0
46.875 0
47.0625 990
47.6094 994
48.2812 1003
48.9375 1014
49.6094 1025
50.1562 1042
```

```
50.9375 1046
51.4844 1060
52.1562 1079
52.8125 1092
53.4844 1111
54.1562 1123
54.6875 1126
55.4844 1149
56.0312 1165
56.6875 1193
57.3594 1213
58.0312 1230
58.6875 1242
59.3594 1266
60.0156 1287
60.5625 1415
61.2188 1420
61.7656 1409
62.5469 1400
63.2188 1395
63.875 1388
64.5469 1383
65.2031 1382
65.875 1373
66.5312 1370
67.2031 1359
67.8594 1352
68.5312 1345
69.2031 1341
69.8594 1335
70.5312 1328
71.1875 1329
71.8594 1322
72.5156 1315
73.3125 1315
73.8438 1317
74.6406 1313
75.1719 1306
75.9688 1304
76.5 1298
77.1719 1298
```

```
77.9531 1297
78.5 1297
79.1719 1292
79.8281 1294
80.5 1290
81.1719 1288
81.9688 1287
82.625 1288
83.1719 1285
83.8438 1286
84.6406 1286
85.1719 1283
85.8438 1281
86.6406 1279
87.1875 1285
87.8438 1282
88.6406 1283
89.3125 1287
89.8594 1289
90.6406 1287
91.3125 1287
91.9844 1290
92.5312 1290
93.3125 1288
93.9844 1290
94.5312 1293
95.2031 1299
95.9844 1302
96.5312 1302
97.2031 1302
97.875 1310
98.5312 1309
99.2031 1314
100 1319
100.672 1318
101.203 1319
101.875 1328
102.531 1331
103.203 1338
103.859 1348
104.531 1351
```

```
105.188 1352
105.859 1362
106.531 1371
106.938 1657
107.609 1657
108.391 1660
109.062 1666
109.719 1658
110.266 1696
110.938 1706
111.594 1701
112.266 1722
112.922 1709
113.594 1712
114.25 1743
114.922 1746
115.578 1758
116.25 1763
116.922 1783
117.578 1785
118.25 1801
118.906 1804
119.578 1811
120.234 1823
120.906 1847
121.578 1860
122.234 1861
122.906 1883
123.453 1910
124.125 1922
124.906 1934
125.453 1966
126.125 1975
126.797 1973
127.469 2025
128.125 2030
128.797 2049
129.469 2015
130.141 1970
130.797 1907
131.484 1945
```

```
132.266 1896
132.938 1887
133.609 1892
134.266 1856
134.938 1841
135.609 1823
136.281 1789
136.938 1783
137.609 1761
138.281 1743
138.953 1747
139.625 1714
140.281 1714
141.828 858
142.5 850
143.172 839
143.844 845
144.625 834
145.297 833
145.828 827
146.625 819
147.297 812
147.828 806
148.75 798
149.297 793
150.078 788
150.625 784
151.281 777
152.078 772
152.75 768
153.406 764
153.953 759
154.75 757
155.406 752
156.094 748
156.734 744
157.281 741
158.078 738
158.734 736
159.406 732
160.078 728
```

```
160.859 727
161.531 723
162.062 720
162.859 719
163.531 715
164.062 715
164.734 712
165.531 710
166.188 709
166.859 706
167.406 705
168.312 702
168.859 702
169.531 699
170.203 699
170.859 699
171.531 698
172.203 696
172.984 694
173.531 694
174.203 693
174.875 695
175.531 694
176.328 690
176.875 693
177.656 693
178.203 692
178.875 694
179.672 691
180.203 693
180.875 693
181.422 694
182.203 696
182.875 696
183.672 698
184.219 697
185 699
185.547 700
186.219 709
186.891 707
187.422 733
```

```
188.219 740
188.766 740
189.438 741
190.219 743
190.641 745
191.438 747
192.234 749
192.781 751
193.562 754
194.109 755
194.781 758
195.469 758
195.984 761
196.656 765
197.453 769
198 772
198.672 776
199.328 780
200.125 780
200.672 784
201.469 787
202.125 793
202.672 794
203.344 800
204.016 805
204.688 811
205.344 813
206.016 817
206.562 824
207.234 830
208.031 833
208.688 842
209.234 848
209.906 853
210.578 859
211.25 864
211.906 870
212.578 877
213.25 883
213.797 893
214.594 900
```

```
215.25 909
215.797 914
216.469 923
217.141 932
217.812 943
218.359 950
219.016 960
219.688 976
220.359 986
221.031 978
221.703 969
222.484 958
223.031 968
223.828 980
224.375 963
225.172 956
225.703 969
226.5 981
226.922 1067
227.719 995
228.516 966
229.188 958
229.719 951
230.516 940
231.062 933
231.859 922
232.516 915
233.312 911
233.859 899
234.531 892
235.203 888
235.859 879
236.531 872
237.328 863
237.875 861
238.656 855
239.453 847
240 843
240.672 839
241.344 835
242.125 829
```

```
242.797 821
243.344 816
244.016 812
244.797 808
245.469 801
246.141 799
246.812 793
247.5 792
248.141 786
248.812 781
249.609 781
250.156 777
250.828 774
251.609 772
252.281 771
252.828 766
253.625 763
254.281 760
254.953 758
255.5 758
256.297 756
256.953 751
257.625 753
258.297 751
258.844 749
259.625 750
260.203 0
260.969 715
261.766 695
262.422 704
263.094 702
263.766 703
264.438 701
265.219 700
265.781 700
266.562 699
267.234 698
267.766 698
268.562 697
269.234 697
269.781 698
```

```
270.438 695
271.234 697
271.781 695
272.453 697
273.109 697
273.766 699
274.438 698
275.094 701
275.625 702
276.547 702
277.203 701
277.875 703
278.406 704
279.062 707
279.609 708
280.391 710
281.047 710
281.594 712
282.25 713
283.047 717
283.703 717
284.359 721
285.031 724
285.688 727
286.219 729
286.891 731
287.672 734
288.219 737
289 741
289.531 743
290.203 747
290.984 751
291.516 753
292.188 759
292.844 762
293.516 764
294.172 770
294.719 776
295.516 779
296.062 784
296.859 790
```

```
297.391 794
297.938 800
298.734 804
299.406 813
300.078 817
300.734 823
301.406 826
301.953 842
302.625 843
303.422 859
303.969 866
304.625 858
305.297 863
305.859 870
308.016 470
308.688 475
309.594 471
309.891 470
310.562 472
311.484 473
311.547 0
311.906 481
312.203 0
312.875 0
313.547 0
313.75 971
314.547 933
315.328 886
316 862
316.656 860
317.328 860
318 861
318.656 857
319.953 638
320.5 634
321.281 636
321.953 639
322.203 0
322.875 0
323.453 788
324.109 784
```

```
324.781 782
325.578 779
326.25 782
326.781 779
327.562 775
328.203 0
330.781 386
331.312 383
332.234 382
332.906 382
333.438 383
333.547 0
334.203 0
334.359 386
334.766 392
334.875 0
335.547 0
336.734 571
337.516 567
338.312 568
338.734 573
338.875 0
339.547 0
340.234 738
341.031 734
341.578 742
342.375 727
343.031 729
343.578 726
344.25 725
344.922 723
345.578 702
346.25 720
346.922 717
347.594 715
348.391 715
349.047 714
349.594 713
350.391 712
351.062 710
351.594 709
```

```
352.391 710
352.938 708
353.547 708
354.344 706
355.016 688
355.922 645
356.578 630
357.25 630
358.031 620
358.594 620
359.234 612
```

APÊNDICE B - ARQUIVO DE TEXTO GERADO COM A NUVEM DE PONTOS COM COORDENADAS CARTESIANAS

```
x y z
607 0.165534 0
605.971 5.9493 0
602.864 12.8256 0
603.626 21.2439 0
604.402 26.8843 0
604.978 35.1807 0
606.519 42.4119 0
606.99 49.4371 0
611.347 57.0156 0
617.756 63.3962 0
620.963 70.9213 0
624.057 78.6985 0
628.232 85.3242 0
632.944 94.7706 0
633.792 102.333 0
635.773 108.71 0
638.389 116.878 0
638.698 125.958 0
642.071 134.464 0
647.323 143.323 0
645.967 149.318 0
643.2 156.647 0
644.27 164.747 0
645.191 173.067 0
646.055 181.256 0
640.595 189.373 0
640.295 197.286 0
643.147 204.907 0
638.802 211.805 0
639.171 220.084 0
635.542 227.401 0
631.958 234.318 0
631.034 242.429 0
631.933 251.118 0
631.359 257.895 0
628.292 265.279 0
```

```
630.713 274.856 0
627.74 284.013 0
627.722 291.254 0
627.872 300.321 0
628.877 309.702 0
632.142 318.842 0
636.512 332.23 0
640.595 343.755 0
646.151 356.556 0
652.298 368.114 0
657.453 381.256 0
656.48 390.807 0
657.846 400.173 0
655.653 409.449 0
656.736 422.68 0
656.85 431.675 0
652.439 441.891 0
645.725 446.391 0
642.201 454.914 0
639.25 464.176 0
628.95 468.057 0
619.442 472.369 0
615.567 480.664 0
603.51 483.017 0
597.002 491.038 0
587.378 494.783 0
583.388 501.018 0
576.86 507.007 0
570.495 515.652 0
569.96 525.138 0
566.091 533.69 0
564.111 544.459 0
566.224 559.469 0
572.289 576.047 0
0 0 0
0 0 0
674.388 724.776 0
670.136 734.135 0
667.472 748.659 0
666.078 764.549 0
664.195 780.686 0
```

```
667.606 800.041 0
659.155 812.176 0
660.091 829.385 0
661.978 852.071 0
660.032 869.955 0
661.091 892.905 0
657.604 910.322 0
650.868 918.829 0
651.059 946.744 0
650.934 966.183 0
655.202 996.975 0
654.253 1021.43 0
651.233 1043.45 0
645.474 1061.1 0
645.218 1089.24 0
643.197 1114.75 0
695.436 1232.31 0
683.682 1244.58 0
666.569 1241.36 0
645.431 1242.34 0
628.566 1245.36 0
611.179 1246.2 0
594.375 1248.76 0
579.615 1254.58 0
561.185 1253.08 0
545.602 1256.67 0
526.566 1252.84 0
509.543 1252.31 0
492.263 1251.68 0
476.131 1253.63 0
459.674 1253.37 0
442.614 1252.07 0
428.566 1258 0
411.605 1256.29 0
395.087 1254.25 0
377.604 1259.62 0
366.464 1264.99 0
347.778 1266.1 0
334.231 1262.51 0
316.155 1265.09 0
303.012 1262.14 0
```

```
288.191 1265.6 0
270.7 1268.44 0
258.58 1270.96 0
242.719 1269 0
228.523 1273.66 0
212.911 1272.31 0
197.67 1272.74 0
179.81 1274.38 0
165.331 1277.34 0
152.775 1275.89 0
137.91 1278.58 0
120.116 1280.38 0
107.986 1278.45 0
92.8415 1277.63 0
74.9481 1276.8 0
63.052 1283.45 0
48.2338 1281.09 0
30.4376 1282.64 0
15.4425 1286.91 0
3.16312 1289 0
-14.3891 1286.92 0
-29.4793 1286.66 0
-44.6693 1289.23 0
-56.9708 1288.74 0
-74.423 1285.85 0
-89.6355 1286.88 0
-102.15 1288.96 0
-117.802 1293.65 0
-135.743 1294.9 0
-148.095 1293.55 0
-163.254 1291.72 0
-179.486 1297.65 0
-194.187 1294.52 0
-210.154 1297.09 0
-229.042 1298.96 0
-244.076 1295.2 0
-256.263 1293.87 0
-273.272 1299.58 0
-288.784 1299.29 0
-305.602 1302.63 0
-322.891 1308.76 0
```

```
-338.971 1307.78 0
-354.207 1304.78 0
-372.195 1310.16 0
-390.096 1314.33 0
-482.745 1585.12 0
-501.275 1579.36 0
-523.73 1575.22 0
-544.101 1574.65 0
-559.422 1560.77 0
-587.459 1591.01 0
-609.652 1593.35 0
-626.014 1581.62 0
-652.478 1593.6 0
-665.617 1574.05 0
-685.233 1568.88 0
-715.883 1589.2 0
-735.737 1583.42 0
-758.998 1585.71 0
-779.755 1581.19 0
-807.302 1589.77 0
-826.376 1582.19 0
-852.449 1586.48 0
-872.007 1579.25 0
-893.924 1575 0
-917.94 1575.03 0
-948.677 1584.75 0
-974.005 1584.59 0
-992.617 1574.18 0
-1022.96 1580.9 0
-1052.89 1593.59 0
-1078.24 1591.06 0
-1106.7 1586.06 0
-1140.35 1601.49 0
-1164.36 1595.27 0
-1181.79 1579.9 0
-1231.87 1607.21 0
-1253.28 1596.93 0
-1283.83 1596.93 0
-1280.86 1555.52 0
-1270 1505.99 0
-1246 1443.65 0
```

```
-1288.39 1457.08 0
-1275.2 1403.1 0
-1285.44 1381.46 0
-1304.98 1369.93 0
-1295.47 1329.09 0
-1300.37 1303.19 0
-1302.68 1275.28 0
-1292.98 1236.42 0
-1302.69 1217.41 0
-1300.61 1187.24 0
-1301.01 1159.93 0
-1317.54 1147.22 0
-1305.76 1110.31 0
-1318.39 1095.29 0
-674.524 530.265 0
-674.35 517.447 0
-671.568 502.909 0
-682.265 498.538 0
-680.027 482.824 0
-684.821 474.246 0
-684.223 464.509 0
-683.937 450.545 0
-683.284 438.711 0
-682.242 429.165 0
-682.22 413.981 0
-681.842 404.896 0
-682.964 393.071 0
-683.199 384.57 0
-681.419 373.36 0
-682.128 361.504 0
-682.765 351.647 0
-683.17 342.016 0
-681.912 333.283 0
-684.673 322.913 0
-683.778 312.972 0
-683.83 303.118 0
-683.499 293.88 0
-683.506 286.183 0
-684.637 275.528 0
-685.883 266.946 0
-685.223 257.476 0
```

```
-684.435 248.059 0
-686.807 238.379 0
-685.762 229.04 0
-685.001 221.751 0
-687.064 211.907 0
-685.666 202.7 0
-687.515 196.337 0
-686.876 187.47 0
-687.481 177.398 0
-688.499 169.264 0
-687.512 160.508 0
-688.037 153.719 0
-687.444 142.213 0
-688.771 135.643 0
-687.364 127.011 0
-688.806 118.94 0
-690.123 111.046 0
-690.389 102.797 0
-689.565 94.4219 0
-688.803 84.7697 0
-689.581 78.1899 0
-689.456 69.9959 0
-692.222 62.0836 0
-691.89 54.0763 0
-688.583 44.1908 0
-691.969 37.7785 0
-692.42 28.3431 0
-691.66 21.7 0
-693.866 13.6258 0
-690.989 3.95573 0
-692.996 -2.45531 0
-692.919 -10.5828 0
-693.786 -17.2223 0
-695.486 -26.7543 0
-695.124 -34.9094 0
-696.567 -44.7031 0
-695.111 -51.2775 0
-696.34 -60.9219 0
-696.722 -67.6636 0
-704.828 -76.8053 0
-701.893 -84.8265 0
```

```
-726.859 -94.6863 0
-732.399 -105.788 0
-731.356 -112.776 0
-730.97 -121.509 0
-731.214 -131.816 0
-732.189 -137.568 0
-732.164 -148.136 0
-731.99 -158.717 0
-732.392 -166.14 0
-732.976 -176.811 0
-732.224 -184.044 0
-732.916 -193.385 0
-730.541 -202.171 0
-731.579 -209.556 0
-732.903 -219.268 0
-733.598 -230.641 0
-734.216 -238.561 0
-735.157 -248.436 0
-736.039 -258.161 0
-732.376 -268.374 0
-733.523 -276.766 0
-732.395 -288.04 0
-734.607 -298.666 0
-732.645 -306.051 0
-734.514 -317.001 0
-735.313 -327.628 0
-736.871 -338.736 0
-734.752 -348.006 0
-734.215 -358.354 0
-737.028 -368.465 0
-737.99 -379.829 0
-735.284 -391.468 0
-738.642 -404.193 0
-739.992 -414.144 0
-739.418 -425.287 0
-739.545 -436.983 0
-738.644 -448.22 0
-738.557 -459.819 0
-739.012 -472.218 0
-738.441 -484.143 0
-742.095 -496.733 0
```

```
-740.876 -510.982 0
-742.327 -524.625 0
-741.34 -534.612 0
-742.257 -548.62 0
-742.946 -562.722 0
-744.995 -578.127 0
-744.931 -589.557 0
-745.891 -604.356 0
-751.065 -623.28 0
-751.334 -638.509 0
-737.759 -642.025 0
-723.459 -644.646 0
-706.492 -647.018 0
-707.593 -660.557 0
-706.993 -678.646 0
-688.331 -673.476 0
-673.962 -678.02 0
-676.728 -693.542 0
-675.276 -711.592 0
-728.754 -779.363 0
-669.403 -736.155 0
-639.889 -723.67 0
-626.129 -725.07 0
-614.857 -725.502 0
-597.711 -725.494 0
-586.371 -725.712 0
-569.426 -725.147 0
-556.814 -726.074 0
-544.284 -730.532 0
-530.207 -726.004 0
-517.594 -726.471 0
-506.755 -729.207 0
-493.322 -727.512 0
-480.896 -727.409 0
-465.872 -726.452 0
-457.852 -729.172 0
-444.75 -730.221 0
-430.484 -729.447 0
-421.5 -730.059 0
-410.949 -731.465 0
-400.424 -732.725 0
```

```
-387.594 -732.811 0
-375.316 -730.191 0
-366.084 -729.272 0
-355.754 -729.92 0
-344.068 -731.082 0
-332.564 -728.699 0
-323.185 -730.72 0
-312.243 -728.94 0
-303.085 -731.713 0
-292.646 -729.489 0
-282.276 -728.204 0
-272.12 -732.06 0
-263.761 -730.862 0
-254.186 -731.072 0
-243.566 -732.571 0
-234.653 -734.424 0
-226.155 -731.854 0
-215.107 -732.05 0
-205.899 -731.577 0
-196.785 -732.011 0
-189.788 -733.856 0
-179.088 -734.482 0
-169.538 -731.613 0
-161.375 -735.505 0
-152.332 -735.388 0
-144.917 -734.847 0
-135.067 -737.738 0
-0 -0 0
-112.233 -706.137 0
-99.5353 -687.836 0
-92.8405 -697.851 0
-84.409 -696.907 0
-76.3383 -698.843 0
-67.9429 -697.7 0
-58.3432 -697.564 0
-51.4982 -698.103 0
-41.9179 -697.742 0
-33.6834 -697.187 0
-27.2086 -697.469 0
-17.4914 -696.78 0
-9.31807 -696.938 0
```

```
-2.66794 -697.995 0
5.31291 -694.98 0
15.0104 -696.838 0
21.6001 -694.664 0
29.8315 -696.361 0
37.8023 -695.974 0
45.9116 -697.491 0
54.0114 -695.907 0
62.2418 -698.231 0
68.808 -698.62 0
80.0408 -697.422 0
87.895 -695.468 0
96.3197 -696.37 0
102.915 -696.437 0
111.355 -698.176 0
118.182 -698.067 0
128.059 -698.356 0
136.046 -696.844 0
143.094 -697.473 0
151.283 -696.766 0
161.863 -698.491 0
169.849 -696.592 0
178.806 -698.477 0
187.763 -699.229 0
196.58 -699.918 0
203.617 -699.987 0
212.393 -699.464 0
222.819 -699.363 0
230.423 -700.053 0
241.246 -700.629 0
248.397 -700.248 0
257.974 -701.041 0
268.939 -701.194 0
276.171 -700.527 0
286.634 -702.796 0
295.826 -702.233 0
304.84 -700.549 0
315.297 -702.487 0
324.499 -704.895 0
335.564 -703.02 0
344.445 -704.282 0
```

```
356.919 -704.776 0
365.288 -704.983 0
374.813 -706.764 0
386.518 -704.996 0
399.179 -708.255 0
409.463 -706.986 0
420.597 -707.409 0
430.428 -704.988 0
445.606 -714.422 0
454.494 -709.989 0
473.138 -716.953 0
483.873 -718.208 0
487.518 -706.038 0
498.654 -704.353 0
509.64 -705.101 0
289.464 -370.284 0
296.913 -370.767 0
300.189 -362.943 0
301.425 -360.615 0
306.928 -358.58 0
313.32 -354.344 0
0 -0 0
321.265 -357.98 0
0 -0 0
0 -0 0
0 -0 0
671.459 -701.415 0
654.494 -664.926 0
630.073 -622.9 0
620.071 -598.796 0
625.431 -590.284 0
632.311 -582.908 0
639.848 -576.121 0
643.399 -566.116 0
488.4 -410.499 0
489.21 -403.274 0
496.222 -397.819 0
503.216 -393.821 0
0 -0 0
0 -0 0
633.054 -469.24 0
```

```
635.145 -459.616 0
638.858 -450.982 0
642.594 -440.356 0
650.209 -434.456 0
651.698 -426.768 0
654.079 -415.7 0
0 -0 0
336.885 -188.426 0
335.985 -183.855 0
338.016 -177.959 0
340.08 -173.983 0
342.575 -171.265 0
0 -0 0
0 -0 0
347.988 -167.034 0
354.593 -167.116 0
0 -0 0
0 -0 0
524.567 -225.545 0
523.9 -216.835 0
527.791 -209.906 0
533.982 -207.826 0
0 -0 0
0 -0 0
694.518 -249.576 0
694.14 -238.591 0
703.976 -234.482 0
692.874 -220.125 0
697.261 -212.762 0
696.383 -205.247 0
697.78 -196.794 0
698.109 -188.077 0
679.878 -174.841 0
699.366 -171.134 0
698.403 -162.241 0
698.305 -153.609 0
700.374 -143.881 0
700.993 -135.663 0
701.273 -128.784 0
702.011 -118.85 0
701.378 -110.31 0
```

```
701.383 -103.646 0
703.748 -94.0126 0
702.629 -87.0439 0
703.514 -79.5708 0
702.563 -69.5802 0
685.399 -59.7718 0
643.367 -45.8688 0
628.877 -37.6045 0
629.274 -30.2262 0
619.634 -21.3024 0
619.813 -15.2129 0
611.945 -8.18172 0
```

APÊNDICE C - ARQUIVO NO FORMATO PCD GERADO A PARTIR DA NUVEM DE PONTOS FILTRADA UTILIZANDO A BIBLIOTECA PCL

```
# .PCD v0.7 - Point Cloud Data file format
VERSION 0.7
FIELDS x y z
SIZE 4 4 4
TYPE F F F
COUNT 1 1 1
WIDTH 525
HEIGHT 1
VIEWPOINT 0 0 0 1 0 0 0
POINTS 525
DATA ascii
-728.75391 -779.36304 0
-135.06747 -737.73761 0
-669.40338 -736.15497 0
-161.37479 -735.5047 0
-152.33176 -735.38837 0
-144.91725 -734.84692 0
-179.08809 -734.48175 0
-234.65305 -734.42426 0
-189.78804 -733.8559 0
-387.59412 -732.81091 0
-400.42404 -732.72479 0
-243.56599 -732.57056 0
-272.11978 -732.06 0
-215.10713 -732.05048 0
-196.78537 -732.01062 0
-226.15474 -731.85382 0
-303.08527 -731.71259 0
-169.53844 -731.6131 0
-205.89896 -731.57751 0
-410.9494 -731.46539 0
-344.06796 -731.08221 0
-254.18555 -731.07159 0
-263.76071 -730.862 0
-323.18533 -730.72034 0
-544.28351 -730.53162 0
-444.74976 -730.22095 0
```

```
-375.31564 -730.19116 0
-421.5 -730.05939 0
-355.75354 -729.92017 0
-292.64645 -729.48889 0
-430.48352 -729.44702 0
-366.08438 -729.2724 0
-506.75546 -729.20703 0
-457.85239 -729.17224 0
-312.24329 -728.93976 0
-332.5636 -728.69916 0
-282.27628 -728.20404 0
-493.32239 -727.51221 0
-480.89557 -727.40875 0
-517.59406 -726.47119 0
-465.87244 -726.45154 0
-556.81396 -726.07385 0
-530.20715 -726.00366 0
-586.37097 -725.7121 0
-614.85651 -725.50153 0
-597.71094 -725.49402 0
-569.42615 -725.14679 0
-626.12878 -725.07013 0
-639.88892 -723.66992 0
483.87253 -718.20844 0
473.13828 -716.9527 0
445.60614 -714.42224 0
-675.27582 -711.59229 0
454.49359 -709.98914 0
399.17892 -708.255 0
420.59668 -707.40894 0
409.46283 -706.98596 0
374.81268 -706.7641 0
-112.23271 -706.13654 0
487.51804 -706.03833 0
509.63953 -705.10107 0
365.2879 -704.98279 0
386.51813 -704.99628 0
430.4278 -704.98792 0
324.49863 -704.89478 0
356.91919 -704.77563 0
498.65424 -704.35284 0
```

```
344.44528 -704.28223 0
335.56448 -703.02026 0
286.63397 -702.79584 0
315.29749 -702.48663 0
295.82623 -702.23273 0
671.45917 -701.41541 0
268.93854 -701.19403 0
257.97446 -701.04077 0
241.246 -700.62927 0
276.17105 -700.52734 0
304.83994 -700.54877 0
248.3974 -700.24835 0
230.42299 -700.05304 0
203.61665 -699.98663 0
196.57994 -699.91809 0
212.39343 -699.46411 0
222.81853 -699.36249 0
187.76334 -699.22882 0
-76.338264 -698.84296 0
68.808029 -698.61969 0
161.86293 -698.49078 0
178.80563 -698.47656 0
128.0589 -698.3559 0
62.241783 -698.23132 0
111.35473 -698.1756 0
-51.498241 -698.10309 0
118.18192 -698.06665 0
-2.6679387 -697.99487 0
-92.840553 -697.85144 0
-41.917923 -697.742 0
-67.942894 -697.69965 0
-58.343174 -697.56439 0
-27.208582 -697.46948 0
45.911564 -697.4906 0
143.09444 -697.4726 0
80.040779 -697.422 0
-33.683426 -697.18677 0
-84.409042 -696.9068 0
-9.3180704 -696.93768 0
15.010382 -696.83838 0
136.04605 -696.84393 0
```

```
-17.491354 -696.78052 0
151.28268 -696.76581 0
169.84943 -696.5918 0
102.91538 -696.43695 0
29.831495 -696.36133 0
96.319679 -696.37024 0
37.802258 -695.97412 0
54.011436 -695.90717 0
87.895012 -695.46783 0
5.3129048 -694.97968 0
21.600119 -694.66425 0
-676.72809 -693.54169 0
-99.535294 -687.83551 0
-706.99347 -678.64587 0
-673.96173 -678.02032 0
-688.33112 -673.47552 0
654.49402 -664.92603 0
-707.59308 -660.55737 0
-706.49237 -647.01813 0
-723.45862 -644.64612 0
-737.75873 -642.02496 0
-751.33386 -638.50873 0
-751.06451 -623.28009 0
630.07281 -622.89984 0
-745.89136 -604.3559 0
620.07092 -598.79553 0
625.43146 -590.28424 0
-744.93085 -589.5575 0
632.31146 -582.90839 0
-744.99512 -578.12738 0
639.84772 -576.12146 0
643.3988 -566.11566 0
-742.94574 -562.72162 0
-742.25684 -548.61993 0
-741.34033 -534.61249 0
-742.32715 -524.625 0
-740.87622 -510.98178 0
-742.09515 -496.73312 0
-738.44067 -484.14291 0
-739.01215 -472.21826 0
633.0545 -469.23981 0
```

```
-738.55719 -459.8187 0
635.14484 -459.61615 0
638.85779 -450.98196 0
-738.64386 -448.22009 0
642.59436 -440.35608 0
-739.54523 -436.98264 0
650.20923 -434.45593 0
651.69794 -426.76788 0
-739.4184 -425.28748 0
654.07861 -415.69965 0
-739.99231 -414.1442 0
488.39978 -410.49927 0
-738.64172 -404.19348 0
489.20999 -403.27359 0
496.22183 -397.81891 0
503.21597 -393.82062 0
-735.28363 -391.46768 0
-737.9903 -379.82928 0
296.91263 -370.76663 0
289.46429 -370.28424 0
-737.02765 -368.46475 0
300.18869 -362.94318 0
301.42468 -360.61496 0
306.92767 -358.57971 0
-734.21472 -358.35428 0
321.26495 -357.98022 0
313.32034 -354.34357 0
-734.75208 -348.00629 0
-736.87109 -338.73587 0
-735.31262 -327.62836 0
-734.51385 -317.00055 0
-732.6449 -306.05139 0
-734.60693 -298.66641 0
-732.39459 -288.04025 0
-733.52344 -276.76584 0
-732.37646 -268.37415 0
-736.0387 -258.16095 0
694.51825 -249.57649 0
-735.15668 -248.43645 0
-734.21564 -238.56113 0
694.13983 -238.59149 0
```

```
703.97601 -234.4819 0
-733.59778 -230.64107 0
524.56683 -225.54524 0
692.87366 -220.12526 0
-732.90283 -219.26804 0
523.90027 -216.83522 0
697.26141 -212.76175 0
527.79126 -209.90562 0
-731.57867 -209.55574 0
533.98248 -207.82613 0
696.38318 -205.2473 0
-730.54138 -202.17146 0
697.78003 -196.79433 0
-732.91632 -193.38486 0
336.88547 -188.42555 0
698.10901 -188.07672 0
-732.22449 -184.04436 0
335.9855 -183.85524 0
338.01559 -177.95915 0
-732.97601 -176.81107 0
679.8783 -174.84137 0
340.0795 -173.98254 0
342.57474 -171.26456 0
699.36627 -171.13385 0
354.59311 -167.11594 0
347.98792 -167.03415 0
-732.39227 -166.14006 0
698.40314 -162.24083 0
-731.99048 -158.71667 0
698.30463 -153.60889 0
-732.16449 -148.13565 0
700.37372 -143.88074 0
-732.1886 -137.56773 0
700.99335 -135.66264 0
-731.21368 -131.81645 0
701.27295 -128.78358 0
-730.96954 -121.50936 0
702.0105 -118.84962 0
-731.35608 -112.77554 0
701.37848 -110.30957 0
-732.39935 -105.78829 0
```

```
701.38324 -103.64632 0
-726.85864 -94.686272 0
703.74829 -94.012581 0
702.62891 -87.043861 0
-701.89276 -84.826492 0
703.5144 -79.570808 0
-704.82758 -76.805275 0
702.56287 -69.580231 0
-696.72205 -67.663574 0
-696.34009 -60.921864 0
685.39868 -59.771755 0
-695.11121 -51.277534 0
643.36694 -45.86882 0
-696.56702 -44.703148 0
628.87671 -37.604488 0
-695.12396 -34.909386 0
629.27448 -30.226221 0
-695.4856 -26.754332 0
619.63391 -21.302437 0
-693.78625 -17.222328 0
619.81335 -15.212857 0
-692.91919 -10.582829 0
611.94531 -8.1817198 0
-692.99567 -2.4553065 0
0 0 0
607 0.16553357 0
-690.98865 3.9557323 0
605.97083 5.9492955 0
602.86359 12.825566 0
-693.86621 13.625783 0
603.62628 21.243858 0
-691.65967 21.700029 0
604.4024 26.884266 0
-692.42017 28.343081 0
604.97797 35.180721 0
-691.96948 37.778549 0
606.51892 42.411938 0
-688.58344 44.190792 0
606.99011 49.43708 0
-691.89001 54.076271 0
611.34705 57.015587 0
```

```
-692.2215 62.08358 0
617.75555 63.396221 0
-689.45599 69.995911 0
620.96307 70.921295 0
-689.5813 78.189941 0
624.05731 78.698456 0
-688.80341 84.769676 0
628.23224 85.324226 0
-689.56543 94.421936 0
632.94434 94.770576 0
633.79181 102.33268 0
-690.38879 102.79745 0
635.77283 108.71027 0
-690.12299 111.04636 0
638.38892 116.87839 0
-688.80634 118.94038 0
638.6983 125.95834 0
-687.36401 127.01075 0
642.07117 134.46432 0
-688.77057 135.64333 0
-687.44421 142.21271 0
647.3233 143.32323 0
645.96674 149.31844 0
-688.03741 153.71893 0
643.19971 156.64653 0
-687.51233 160.50784 0
644.26971 164.7469 0
-688.49878 169.26442 0
645.19135 173.0668 0
-687.4809 177.39787 0
646.05511 181.25607 0
-686.87622 187.47005 0
640.59479 189.37344 0
-687.51495 196.33687 0
640.29529 197.28642 0
-685.66589 202.70003 0
643.14709 204.90692 0
638.80164 211.80531 0
-687.06372 211.90671 0
639.17053 220.08408 0
-685.00104 221.75113 0
```

```
635.54224 227.40065 0
-685.76202 229.04027 0
631.95819 234.31779 0
-686.80743 238.37894 0
631.03436 242.42856 0
-684.43457 248.05913 0
631.93341 251.1179 0
-685.22253 257.47632 0
631.35913 257.89459 0
628.29199 265.27945 0
-685.8833 266.94595 0
630.71259 274.85571 0
-684.63739 275.52789 0
627.73993 284.01337 0
-683.50586 286.18304 0
627.72223 291.25381 0
-683.4986 293.88031 0
627.87213 300.3208 0
-683.8302 303.11752 0
628.87653 309.70197 0
-683.77832 312.97156 0
632.14233 318.84183 0
-684.67255 322.91254 0
636.51178 332.23004 0
-681.9115 333.28317 0
-683.16968 342.01642 0
640.59485 343.75461 0
-682.76514 351.64716 0
646.15131 356.5564 0
-682.1283 361.50375 0
652.29834 368.11392 0
-681.41882 373.35965 0
657.45251 381.25604 0
-683.19952 384.57047 0
656.4801 390.80667 0
-682.96375 393.07062 0
657.84601 400.17325 0
-681.84167 404.89624 0
655.65271 409.4491 0
-682.21967 413.98105 0
656.73639 422.6799 0
```

```
-682.24152 429.16492 0
656.8504 431.67526 0
-683.28381 438.71091 0
652.43903 441.89056 0
645.72449 446.39093 0
-683.93707 450.54535 0
642.20105 454.91406 0
639.24982 464.17633 0
-684.22272 464.50864 0
628.95007 468.05743 0
619.44226 472.3688 0
-684.82117 474.2457 0
615.56732 480.66403 0
-680.02734 482.82382 0
603.50964 483.01666 0
597.00177 491.03754 0
587.37848 494.78333 0
-682.26453 498.53799 0
583.38849 501.01785 0
-671.56793 502.90906 0
576.86023 507.00714 0
570.49481 515.65167 0
-674.35034 517.4472 0
569.96045 525.13818 0
-674.52441 530.26483 0
566.09076 533.69019 0
564.11066 544.45862 0
566.22424 559.46948 0
572.2887 576.04657 0
674.38818 724.77625 0
670.13617 734.13458 0
667.47174 748.65912 0
666.07825 764.54938 0
664.19482 780.68573 0
667.60608 800.04132 0
659.15546 812.17615 0
660.09137 829.38495 0
661.97827 852.07147 0
660.03247 869.95471 0
661.09125 892.90503 0
657.60358 910.3222 0
```

```
650.86816 918.82898 0
651.05853 946.74377 0
650.93372 966.18335 0
655.20172 996.97528 0
654.25293 1021.4314 0
651.2326 1043.454 0
645.47424 1061.097 0
645.21844 1089.2424 0
-1318.3877 1095.2853 0
-1305.7612 1110.3079 0
643.19653 1114.7499 0
-1317.537 1147.2163 0
-1301.0057 1159.928 0
-1300.6063 1187.2422 0
-1302.687 1217.4135 0
695.43549 1232.3126 0
-1292.9783 1236.4175 0
666.56946 1241.3566 0
645.43134 1242.3439 0
683.68188 1244.5798 0
628.56555 1245.3635 0
611.17938 1246.1957 0
594.37488 1248.7625 0
492.26263 1251.6799 0
442.61377 1252.0691 0
509.54272 1252.3059 0
526.56592 1252.8405 0
561.18451 1253.0765 0
459.67395 1253.3654 0
476.13062 1253.627 0
395.08664 1254.2454 0
579.61493 1254.5798 0
411.60455 1256.2904 0
545.60199 1256.6696 0
428.56558 1258.0034 0
377.60428 1259.619 0
303.01208 1262.1361 0
334.23138 1262.5076 0
366.46436 1264.9873 0
316.15509 1265.0936 0
288.19067 1265.6027 0
```

```
347.77811 1266.104 0
270.69983 1268.4363 0
242.71906 1268.9962 0
258.5802 1270.9624 0
212.91141 1272.3085 0
197.66998 1272.7413 0
228.52303 1273.6614 0
179.80975 1274.3773 0
-1302.6841 1275.2816 0
152.77486 1275.8859 0
74.948128 1276.8021 0
165.33139 1277.3447 0
92.841469 1277.6312 0
107.98569 1278.4475 0
137.90979 1278.584 0
120.11604 1280.3782 0
48.233849 1281.0923 0
30.437609 1282.6389 0
63.05196 1283.4521 0
-74.422997 1285.848 0
-29.479305 1286.6624 0
-89.635475 1286.8821 0
-14.389106 1286.9196 0
15.442521 1286.9073 0
-56.970795 1288.7413 0
-102.14952 1288.9586 0
3.1631165 1288.9961 0
-44.669334 1289.2263 0
-163.25375 1291.7245 0
-148.095 1293.55 0
-117.80171 1293.6475 0
-256.26285 1293.8665 0
-194.18747 1294.5162 0
-135.7435 1294.9045 0
-244.07567 1295.2031 0
-210.15405 1297.0857 0
-179.48616 1297.6459 0
-229.04195 1298.9614 0
-288.78415 1299.2939 0
-273.27213 1299.5793 0
-305.60168 1302.6326 0
```

```
-1300.3741 1303.1915 0
-354.20651 1304.7765 0
-338.97101 1307.7843 0
-322.89096 1308.7572 0
-372.19501 1310.1583 0
-390.09622 1314.3309 0
-1295.4703 1329.0947 0
-1304.9753 1369.9282 0
-1285.4368 1381.4562 0
-1275.1993 1403.0975 0
-1245.9976 1443.6548 0
-1288.3892 1457.0787 0
-1270.0016 1505.9867 0
-1280.8562 1555.5167 0
-559.42157 1560.7727 0
-685.23328 1568.8848 0
-665.61725 1574.0504 0
-992.61707 1574.1768 0
-544.10077 1574.6461 0
-893.92401 1574.9987 0
-917.94019 1575.0286 0
-523.72998 1575.2164 0
-872.00677 1579.2467 0
-501.27499 1579.3583 0
-1181.7909 1579.9049 0
-1022.9631 1580.8971 0
-779.75494 1581.1866 0
-626.01422 1581.6154 0
-826.37598 1582.1908 0
-735.73657 1583.4165 0
-974.00543 1584.5862 0
-948.67664 1584.7466 0
-482.74493 1585.1202 0
-758.99792 1585.7131 0
-1106.6962 1586.0579 0
-852.44873 1586.4843 0
-715.88293 1589.2013 0
-807.30157 1589.7651 0
-587.45886 1591.0085 0
-1078.2424 1591.0616 0
-609.65192 1593.3489 0
```

```
-1052.8928 1593.5862 0
-652.47797 1593.5986 0
-1164.359 1595.2721 0
-1283.8276 1596.9307 0
-1253.2798 1596.9314 0
-1140.3486 1601.4871 0
-1231.8724 1607.2073 0
```

APÊNDICE D - MEDIÇÕES DE TEMPO DOS DOIS MÉTODOS DE DETECÇÃO DE OBJETOS

Medição	Método Proposto	Método PCL
1	114 ms	113 ms
2	100 ms	108 ms
3	97 ms	116 ms
4	89 ms	113 ms
5	99 ms	114 ms
6	105 ms	89 ms
7	97 ms	100 ms
8	94 ms	93 ms
9	83 ms	90 ms
10	87 ms	86 ms
11	93 ms	122 ms
12	97 ms	82 ms
13	93 ms	98 ms
14	97 ms	87 ms
15	112 ms	105 ms
16	92 ms	104 ms
17	101 ms	117 ms
18	108 ms	111 ms
19	82 ms	93 ms
20	91 ms	104 ms
21	94 ms	85 ms
22	96 ms	99 ms
23	113 ms	102 ms
24	109 ms	91 ms
25	120 ms	79 ms
26	103 ms	104 ms
27	91 ms	108 ms
28	109 ms	79 ms
29	90 ms	93 ms
30	87 ms	84 ms
31	110 ms	92 ms
32	97 ms	93 ms
33	102 ms	76 ms
34	87 ms	98 ms
35	111 ms	98 ms
36	89 ms	113 ms
37	95 ms	102 ms
38	88 ms	86 ms
39	105 ms	92 ms
40	96 ms	91 ms