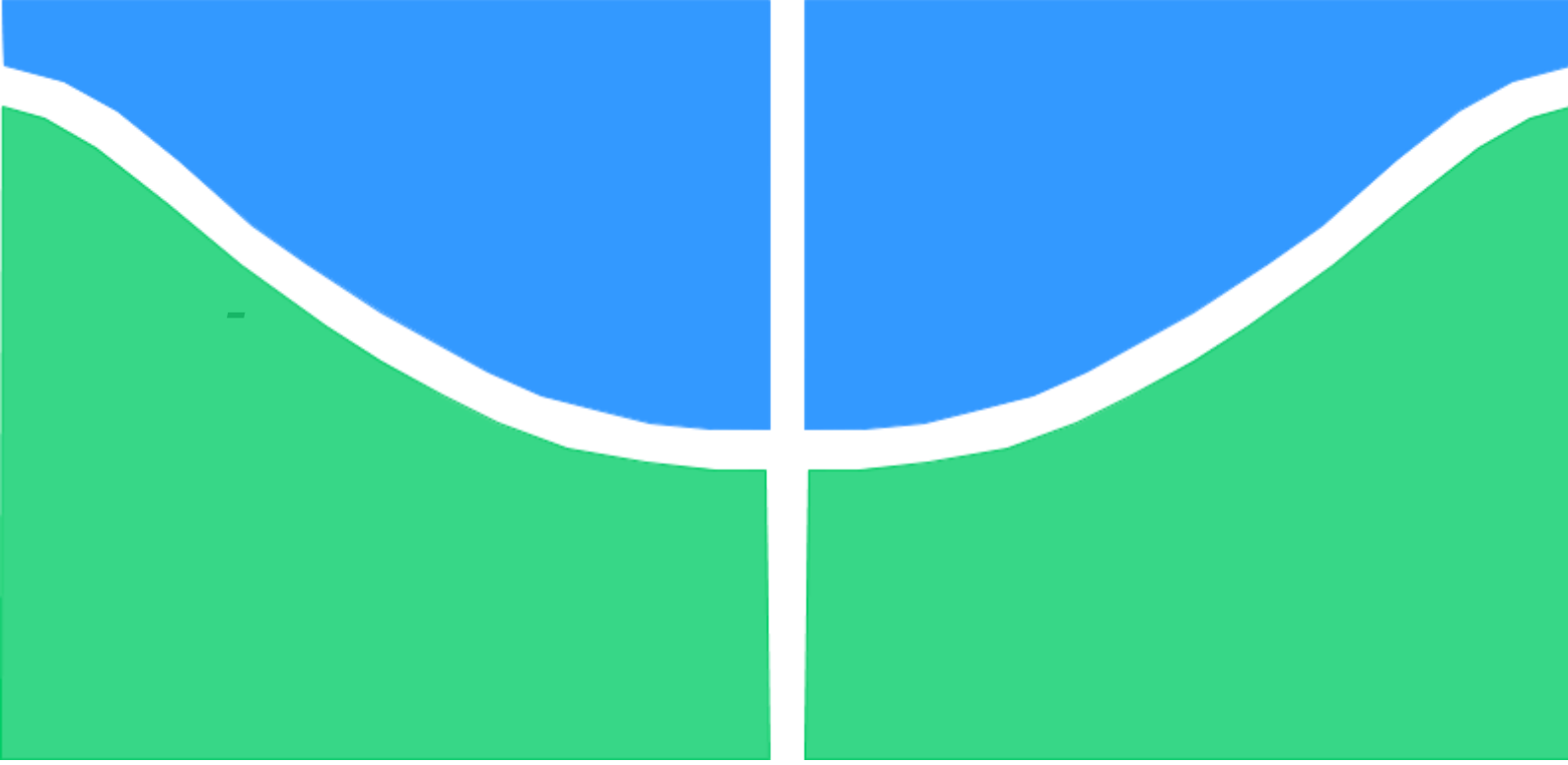

PROJETO FINAL DE GRADUAÇÃO

ESTUDOS EM MAPAS DE ALTA DEFINIÇÃO PARA NAVEGAÇÃO AUTÔNOMA

Gustavo Veloso Gianini

Igor Coutinho Soriano Lousada

Brasília, Dezembro de 2019

UNIVERSIDADE DE BRASÍLIA

FACULDADE DE TECNOLOGIA

UNIVERSIDADE DE BRASÍLIA
Faculdade de Tecnologia

TRABALHO DE GRADUAÇÃO

ESTUDOS EM MAPAS DE ALTA DEFINIÇÃO PARA NAVEGAÇÃO AUTÔNOMA

Gustavo Veloso Gianini

Igor Coutinho Soriano Lousada

Relatório submetido como requisito parcial para obtenção
do grau de Engenheiro de Redes de Comunicação

Banca Examinadora

Prof. Ricardo Lopes de Queiroz, UnB/CIC (Orientador) _____

Prof., João Luiz Azevedo de Carvalho, UnB/ENE _____

Prof., Diogo Caetano Garcia, UnB/Gama _____

RESUMO

O desenvolvimento de veículos autônomos, capazes de auxiliar no deslocamento urbano, tem sido um dos principais temas de pesquisas em universidades e empresas de todo o mundo. Neste contexto, os mapas de alta definição são capazes de oferecer para o veículo inteligente informações sobre o ambiente ao redor, como obstáculos, placas de trânsito, pedestres, informações sobre tráfego, entre outras informações pertinentes. Neste projeto, foram desenvolvidas duas contribuições no contexto de nuvens de pontos. A primeira contribuição consiste em um método para realizar a sobreposição de nuvem de pontos de uma determinada região, formando uma nuvem de pontos com maior densidade de pontos. Pelo alto custo de sensor LiDAR e impossibilidade de realizar múltiplas capturas em ambiente urbano, foi utilizado o KITTI data set. A segunda contribuição, foram aplicados métodos de voxelização e compressão de nuvem de pontos para melhor viabilizar a sua transmissão. Os resultados obtidos foram dentro do esperado, já que de fato na primeira contribuição, obteve-se nuvens de pontos globais com maior densidade de pontos em comparação às nuvens de ponto iniciais. Na segunda abordagem também for possível otimizar a transmissão já que o número final de bits foi substantivamente menor que o inicial.

Palavras-chave: Navegação autônoma, nuvem de pontos.

ABSTRACT

The development of autonomous vehicles, capable of assisting in urban travel, has been one of the main research topics in universities and companies around the world. In this context, high definition maps are able to provide the intelligent vehicle with information about the surrounding environment, such as obstacles, traffic signs, pedestrians, traffic information, among other pertinent information. In this project, two contributions were developed in the context of point clouds. The first contribution consists of a method to perform the point cloud overlap of a given region, forming a point cloud with higher point density. Due to the high cost of the LiDAR sensor and the impossibility of making multiple captures in an urban environment, the KITTI data set was used. The second contribution, were applied voxelization and point cloud compression methods to better enable its transmission. The results obtained were as expected, since in fact in the first contribution, it was obtained global point clouds with higher point density compared to the initial point clouds. In the second approach it is also possible to optimize the transmission since the final number of bits was substantially smaller than the initial one.

Key-words: Point cloud, Autonomous Navigation.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE EQUAÇÕES

1. INTRODUÇÃO

1.1. Contextualização

Automação é um tema discutido e estudado à exaustão, isso se deve ao seu potencial de impacto em tudo que nos cerca. A efemeridade da vida humana é a motivação da busca constante por tudo aquilo que possibilite melhor aproveitar o pouco tempo que nos resta. Nessa seara a automação de tarefas tem como característica o aumento da produtividade.

> Automação é a aplicação de máquinas em tarefas antes executadas por seres humanos ou, cada vez mais, em tarefas que seriam impossíveis de serem realizadas de outra maneira. Embora o termo mecanização seja frequentemente usado para se referir à simples substituição do trabalho humano por máquinas, a automação geralmente implica a integração de máquinas em um sistema autônomo. (GROOVER, 2019, tradução nossa).

A natureza intrínseca de um computador é executar tarefas precisamente definidas, com velocidade e exatidão, sendo atualmente imbatível nesse quesito. Uma barreira foi ultrapassada pela máquina na disputa com o homem em 1997 quando o IBM Deep Blue, um supercomputador à época, venceu por 3½ a 2½ o campeão mundial de xadrez Garry Kasparov (Figura 1).

Figura 1. *Garry Kasparov vs IBM Deep Blue - 1997. fonte: (FORBES, 2011)*

O supercomputador Deep Blue conseguia calcular dezenas de lances à frente em várias linhas possíveis, porém com toda essa “inteligência”, não conseguiria entender uma piada, por mais simples que fosse pois o computador é desprovido de capacidades humanas, sejam elas capacidades cognitivas ou de gerar inferências subjetivas. Um exemplo dessas capacidades é o bom senso. Quando um computador é exposto à uma situação não pré-programada, por mais simples que esta situação seja, terá sua performance prejudicada e poderá gerar resultados inesperados. Mesmo diante do expressivo aumento do poder de processamento dos computadores, inclusive possibilitando superar humanos em muitas tarefas, a natureza humana é invencível até o momento na capacidade de compreensão, dedução e inferência.

Verificando essas diferenças, cada vez mais percebe-se tarefas que não exigem níveis humanos de compreensão, dedução e inferência. Essas tarefas podem ser simplificadas à tomadas de decisões do tipo “se isso, então aquilo”, ou seja,

possuem o potencial de serem automatizadas, usufruindo da velocidade e exatidão disponíveis nos computadores, além de dispensar a utilização de recursos humanos para sua execução. O intuito é que o homem seja responsável pela parte intelectual das tarefas, utilizando de sua capacidade cognitiva para tomar decisões complexas, enquanto entidades automatizadas sejam responsáveis pelo trabalhos estimáveis ou previsíveis.

Um campo que, até então havia sido considerado complexo demais para as máquinas, tem perdido esse estigma. Com a evolução da tecnologia, novas oportunidades têm surgido para a automação veicular. Um passo relevante nesse quesito foi o início da automação de classificação para rotas automotivas. A decisão (ou pelo menos sugestões) sobre qual seria a melhor rota a ser percorrida passou a ser realizada por um programa de computador e não mais pelo motorista. A decisão é tomada de acordo com parâmetros pré-definidos, gerando diferentes tipos de escolha. Por exemplo: rota mais curta, rota mais rápida, rota com menos pedágios, rota mais rápida que não passa por rodovias estaduais e que possui postos de gasolina nos primeiros 5km, etc. Esse serviço de mapas inteligentes foi disponibilizado em larga escala na última década, inclusive disponibilizando interfaces de programação de aplicações (APIs) para que outros possam atuar diretamente personalizando a aplicação. APIs consistem em conjuntos de rotinas e padrões de programação para acesso a um aplicativo de software ou plataforma baseado na Web. Dessa forma esse tipo de serviço poderia ser utilizado para integração com outras aplicações. Por exemplo uma aplicação de entrega de comida em casa poderia utilizar esse serviço para calcular o tempo que a refeição demora para sair do restaurante e chegar ao destino (o cliente). Um exemplo de interface gráfica dessas aplicações pode ser visto na Figura 2

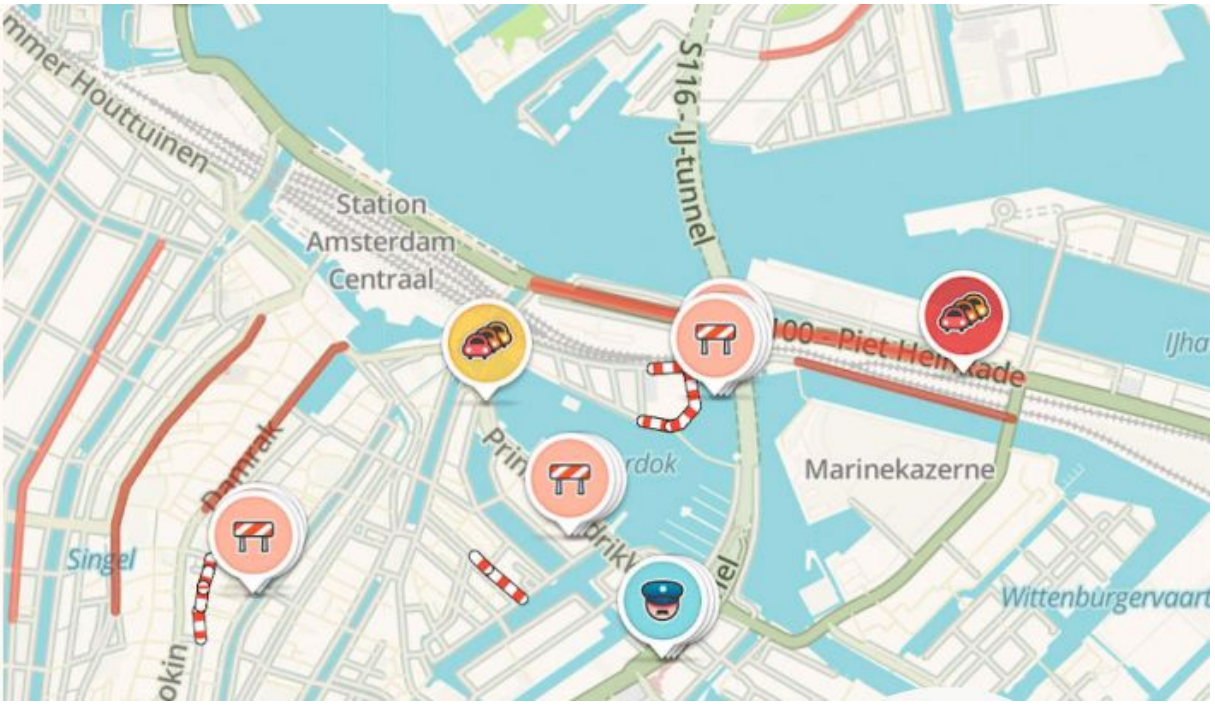

Figura 2. *Exemplo de mapa inteligente. Fonte: (WAZE, 2019)*

No contexto da automação veicular a abordagem de mapas é muito vantajosa. Embora as informações de posição baseadas em GPS sejam geralmente adequadas para guiar motoristas humanos, sua precisão não é suficiente para navegar em um veículo autônomo. A precisão da orientação via GPS nos aplicativos de mercado gira em torno de 1m, os passageiros de um carro autônomo estariam em risco com um sistema de localização com esse nível de precisão. Em vez disso, a posição e orientação precisas do SDV (Self Driving Vehicle) devem ser calculadas registrando as medições de seus sensores visuais ou de distância nos mapas 3D de alta definição (HD). (Nagy e Benedek, 2019). Um exemplo desse tipo de mapa pode ser encontrado na Figura 3

No momento em que o condutor do carro passa a ser um software, o paradigma da orientação via mapas muda, já que o usuário do mapa não é mais o motorista, mas uma máquina. Como resultado, é necessária uma nova geração de mapas criados propositadamente para máquinas. Esses mapas de última geração para máquinas vêm na forma de uma representação altamente precisa e realista da estrada, geralmente chamada de mapas de alta definição (HD).

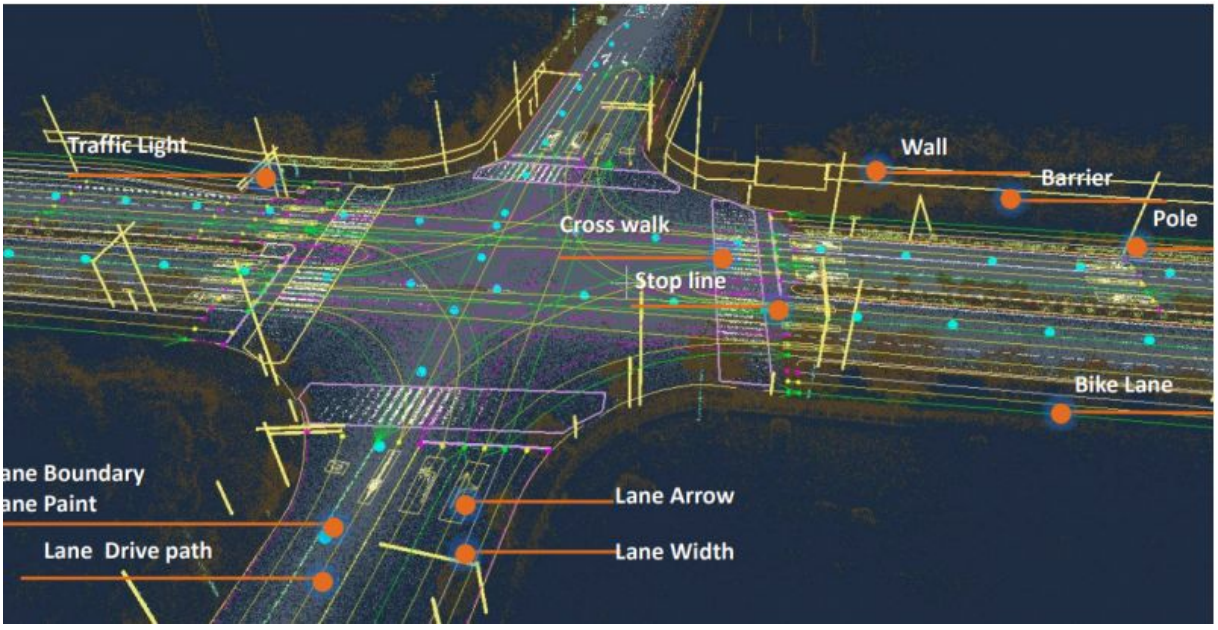

Figura 3*. Mapa de Alta definição. Fonte: (FUTURECAR, 2019)*

> O potencial da inteligência artificial e da robótica avançada executar tarefas antes reservadas para seres humanos não é mais reservado para demonstrações espetaculares como Watson da IBM, Rethink Baxter, DeepMind ou carros autônomos. Apenas observe um aeroporto: os quiosques de check-in automatizado agora dominam as áreas de emissão de bilhetes das companhias aéreas. Pilotos apenas pilotam aeronaves ativamente por em torno de três a sete minutos de muitos em vôos, com o piloto automático guiando o restante a jornada. Os processos de controle de passaportes em alguns aeroportos tem mais ênfase na digitalização de códigos de barras de documentos do que na observação dos passageiros por um humano. (CHUI, 2015, tradução nossa)

Na transição de carruagens de cavalos para automóveis, foram perdidas importantes habilidades de desvio de obstáculos, assim como a capacidade ocasional de realizar “missões autônomas”. Muitas vezes, os cavalos levavam uma carruagem para casa com segurança, mesmo que o motorista não estivesse completamente apto para a viagem. O automóvel autônomo visa recuperar essa autonomia e, sem dúvidas, ir muito além das capacidades de seu antecessor.

É perceptível que a sociedade se aproxima da próxima revolução de mobilidade humana. A tecnologia de automação veicular vem avançando a passos largos, e os carros autônomos se tornarão um elemento corriqueiro do tráfego rodoviário. A sociedade como um todo ganhará milhões de horas por dia em tempo

potencialmente produtivo durante o deslocamento dos indivíduos. As tecnologias avançam impulsionadas pela a indústria e a academia. Para essa automação é necessária uma imensa geração de dados de tudo que orbita o campo veicular. Desde a pista e os elementos físicos que com ela interagem como postes de iluminação, calçadas, pedestres, semáforos etc. Até os elementos lógicos como as diferentes legislações ou as características climáticas de uma região, por exemplo limites de velocidade ou probabilidade de chuva, neve ou raios. Os dados de elementos físicos serão fornecidos por sistemas de sensoriamento e localização compostos por câmeras e sensores. Esses sensores e sistemas determinam e monitoram a posição e o ambiente do veículo.

Figura 4. *Exemplo de ferramentas de sensoriamento e localização em um veículo autônomo. Fonte: (Winner e Wachenfeld, 2016)*

Uma das referências quando se trata de arquitetura de veículos autônomos é a empresa Navia (NAVYA, 2019), esse tipo de veículo (Figura 4) recebe um sistema de câmeras, sensores LiDAR, sensores de ultrassom e sistema de GPS. Nessa arquitetura, enquanto a navegação por satélite permite uma localização de alta precisão do veículo (precisões de 1 cm, nos casos em que métodos corretivos

adicionais são utilizados). Os sensores a laser servem principalmente como um meio de detectar objetos (por exemplo, pessoas, veículos, edifícios, obstáculos) a distâncias de médio a longo alcance. Os sensores de ultrassom são usados para detecção de objetos a curta distância (a menos de 2 m do veículo). Além disso, os sistemas de câmera junto à classificadores de imagem, fornecem informações extras sobre a forma e o tipo de objeto detectado (por exemplo, determinando se é uma pessoa ou planta), para fornecer a imagem mais detalhada possível com base no tipo de objeto, distância, direção e, em alguns casos, velocidade. Variações do sistema de referência, ou seja, obstáculos na rota planejada, são reconhecidas pela unidade de controle do veículo e a rota real pode ser atualizada de acordo com uma série de critérios de avaliação. Em outras palavras, uma rota e um caminho ótimos são calculados e criados de acordo com o destino especificado, a situação e o ambiente de tráfego especificados. Mesmo que nenhum caminho pareça acessível em um determinado momento, por exemplo, se os sensores do veículo detectarem que um obstáculo não pode ser contornado e não houver uma rota alternativa - o veículo interrompe sua jornada até que o obstáculo seja removido Os comandos de direção são enviados da unidade de planejamento de trajetos para os sistemas de direção, frenagem e acionamento e são executados. O veículo possui interfaces eletrônicas para esses componentes, para que a unidade de controle central possa retransmitir comandos de direção, frenagem e acionamento, com o resultado de que o veículo executa a jornada de forma independente, dependendo da combinação de percepção ambiental, processamento de informações e destino. Tais dados devem ser processados em tempo real por computadores dentro ou fora dos veículos. Os veículos autônomos trocarão informações permanentemente entre si e com a infraestrutura de transporte. Os softwares de mobilidade avaliarão a cada instante as decisões relacionadas por exemplo às rotas ou perigos no percurso. (NAVYA, 2019).

Figura 5*. Navia Shuttle, Fonte: (NAVYA, 2019)*

Um dos processos para veículos autônomos é denominado "Localização e Mapeamento Simultâneo" (SLAM). Esse processo é utilizado nos carros da Navya Shuttle (Figura 5). Para esse processo, o veículo é guiado pelo pessoal operacional dentro de uma área operacional planejada, enquanto as coordenadas do sistema de navegação por satélite e os dados recuperados do laser, câmera e (se necessário) sistemas de ultrassom são registrados. Um mapa digital da área operacional, que, diferentemente dos mapas convencionais, é uma representação tridimensional, é criado a partir desses dados. Esta representação descreve a situação estacionária na área operacional. Em outras palavras, todas as variações, medidas em relação aos dados salvos na operação subsequente do veículo, são classificadas como obstáculos móveis ou "novos". Tais variações requerem atenção especial e podem exigir um desvio da rota pré-programada. (WINNER e WACHENFELD, 2016). Um exemplo de diagrama de processos de carros autônomos pode ser visto na Figura 6.

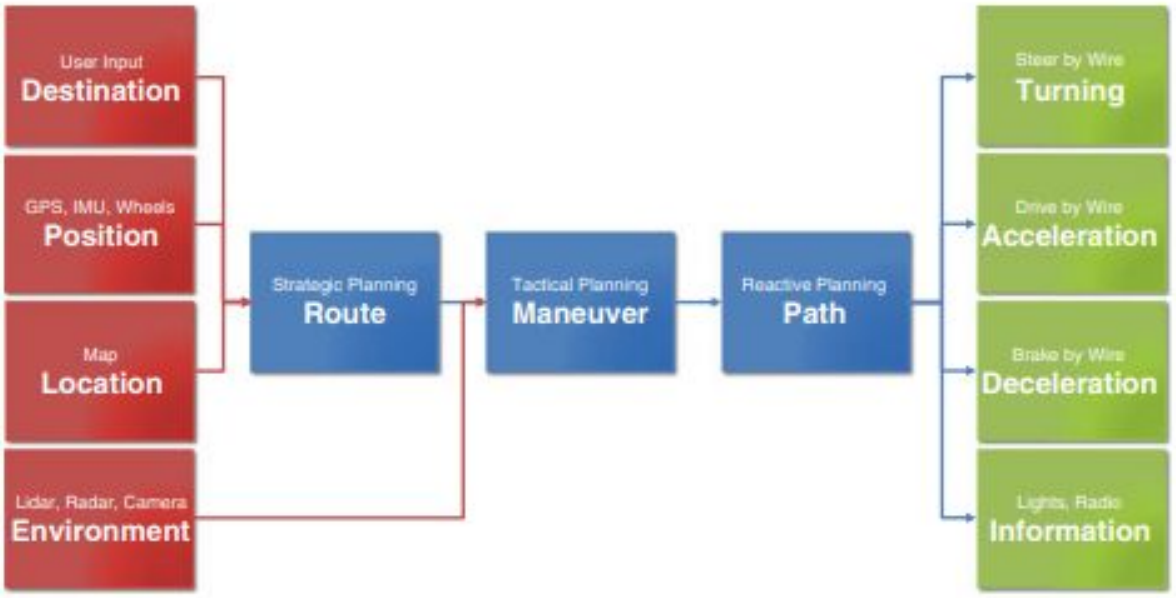

Figura 6. *Diagrama simplificado de processos em um veículo autônomo. Fonte: (Fraedrich et al., 2015)*

Pesquisas recentes relatam altos níveis de ansiedade sobre automação e outras tendências tecnológicas, ressaltando as preocupações sobre seus efeitos. Apesar dessas expectativas e preocupações, a ciência está longe de uma compreensão satisfatória de como a automação em geral, afeta o mercado de trabalho e a produtividade.

> Grande parte do debate na imprensa e os círculos acadêmicos estão centrados em uma dicotomia falsa. De um lado estão os alarmistas com argumentos de que os próximos avanços em IA e robótica significarão o fim do trabalho humano, enquanto muitos economistas do outro lado afirmam que, como os avanços tecnológicos no passado aumentaram a demanda por trabalho e salários, não há razão para se preocupar com o fato de que desta vez será diferente (ACEMOGLU e RESTREPO, 2018, tradução nossa).

Apesar desses altos níveis de ansiedade e de toda a polêmica em torno do tema, o intuito desse projeto final de graduação é colaborar para a evolução e o desenvolvimento da tecnologia de automação veicular através da apresentação de estudo que se espera útil, em seu conceito ou ferramenta prática, no contexto da automação, sem entrar nos méritos filosóficos das possíveis consequências da

evolução tecnológica nesse setor específico. Tais aspectos são amplamente abordados em (Lin, 2016).

1.2. Motivação

A constante evolução no mercado de veículos autônomos gera uma demanda por tecnologias que ofereçam suporte à tomada de decisão durante o movimento do veículo, visto que qualquer decisão errada ou imprecisão na movimentação deste em uma via contendo pedestres, ciclistas e outros veículos (autônomos ou não) pode causar acidentes.

Desta maneira, o estudo de técnicas que tenham o potencial de minimizar o número de acidentes envolvendo veículos autônomos é de extrema relevância para a sociedade.

1.3. Definição do Problema

Um cenário de rodovia que utilize um sistema de veículos autônomos necessita de uma infraestrutura capaz de fornecer a cada um destes veículos informações sobre o ambiente, tornando possível detectar os diferentes elementos que compõem o ambiente e consequentemente tomar as decisões adequadas em cada instante de tempo. Cada veículo deve ser capaz de receber e processar as informações recebidas para realizar o processo de tomada de decisões, para isso os dados enviados devem respeitar as limitações de taxa de transmissão disponíveis na rede destes veículos.

1.4. Objetivo

O objetivo desse projeto final de graduação é implementar um processo para otimização da transmissão de nuvem de pontos que compõem um mapa de alta definição, aumentando a viabilidade do envio de informações de ambiente rodoviário no contexto de automação veicular. A otimização consiste em transmitir vários quadros de nuvem de pontos, esses conjuntos são tratado como nuvem de pontos globais. A vantagem dessa abordagem é o aumento da densidade de pontos, melhorando a eficiência dos algoritmos de detecção e classificação de imagens.

Posteriormente essas nuvem de pontos globais são comprimidas para melhorar a viabilidade da transmissão de informações. Os dados serão utilizadas para na orientação de automóveis através de mapas de alta definição, com atualização em tempo real.

2. FUNDAMENTAÇÃO TEÓRICA

2.1. Imagem real e imagem digital

Imagens são representações visuais de algo. Essa representação dá-se por diversos meios como pintura, fotografia, vídeo, etc. A imagem digital é a matéria prima para processamento de imagens, mais especificamente são utilizados os dados gerados pela análise de uma imagem digital. Uma imagem digital é formada por pixels, mas pode também ser representada numericamente. Uma imagem pode ter representação colorida ou em escala de cinza. No caso de imagem colorida, a cor de cada pixel da sua representação é definida por três valores variando de 0 a 255, cada valor representando a intensidade de uma cor: Vermelha, verde e azul. Esses três valores são combinados para formar a cor do ponto específico. Esse método é denominado modelo de cores RGB, e é baseado na teoria de visão colorida tricromática, de Young-Helmholtz (YOUNG, 1800), e no triângulo de cores de Maxwell (MAXWELL, 1857) e (JUDD, 1935). No caso de imagens em escala de cinza, cada pixel é representado por um valor resultado de um cálculo que relaciona uma determinada faixa do espectro visível à intensidade da luz em cada pixel. Nesse caso a cor é definida por um único valor variando de 0 a 255, em que 0 representa a cor preta e 255 representa a cor branca, os valores intermediários são escala de cinza proporcionais à distância dos extremos.

> Uma imagem (real) pode ser definida como uma função bidimensional, onde x e y são coordenadas espaciais (planas) e a amplitude em qualquer par de coordenadas (x, y) é denominada intensidade ou nível de cinza da imagem naquele ponto. Quando x, y, e os valores de intensidade são quantidades finitas e discretas, chamamos a imagem de imagem digital. (GONZALES e WOODS, 2018)

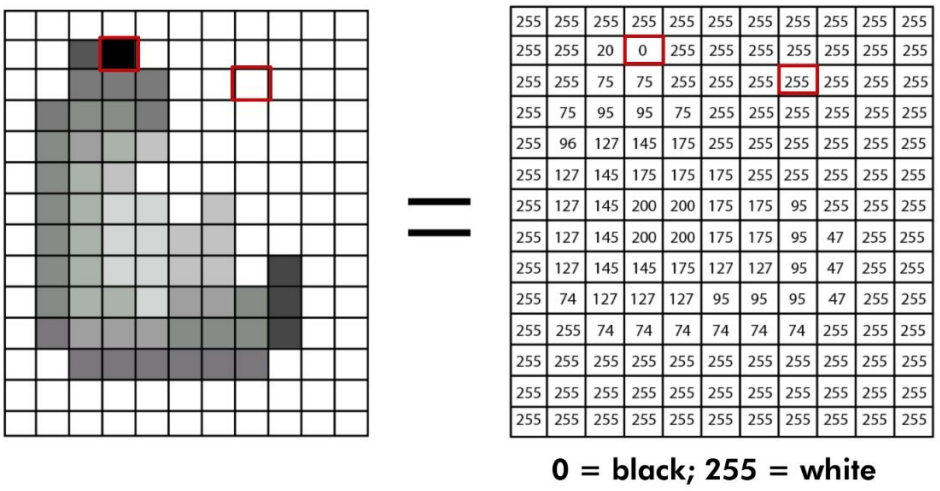

Figura 7. *Imagem digital e sua representação numérica. Fonte: (EDTECH, 2019)*

2.2. Análise de imagens digitais

A análise de imagem digital é um processo que transforma uma imagem digital em algo que não seja uma imagem digital (SHIH, 2017). A Figura 8 apresenta um exemplo de análise de imagem digital. Dentro do escopo desse projeto final de graduação, a análise de imagens não foi utilizada. Utilizou-se dados (nuvens de pontos) gerados por um sensor LiDAR (Light Detection And Ranging), disponibilizados em um repositório aberto, (GEIGER et al., 2013) portanto não houve necessidade de realizar análise prévia de imagens já que os dados foram obtidos diretamente.

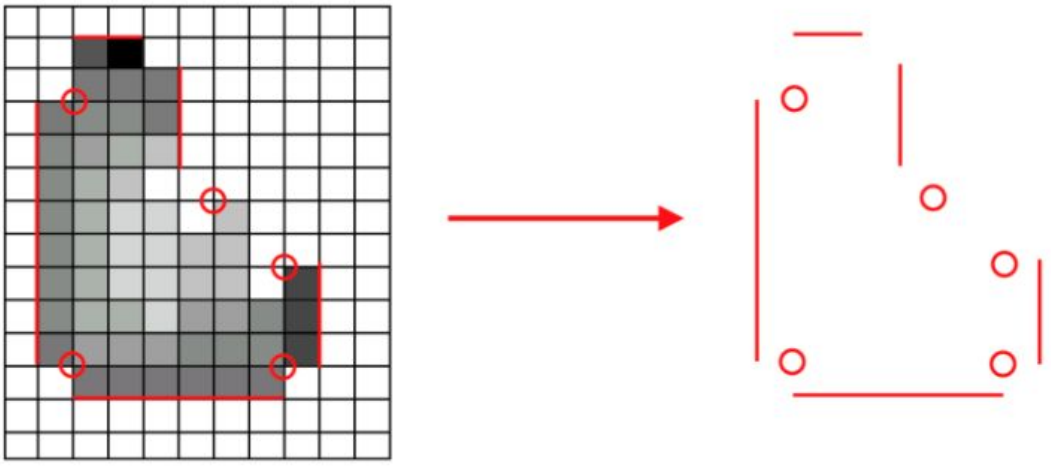

Figura 8. *Análise de imagem digital. Fonte: (EDTECH, 2019)*

2.3. Processamento de Imagens Digitais

O Processamento Digital de Imagens (PDI) é uma área da eletrônica/teoria de sinais em que imagens são convertidas em matrizes de números inteiros, sendo que cada elemento desta matriz é composta por um elemento fundamental: o pixel (uma abreviação de picture element). A Figura 9 representa visualmente um conjunto de pixels. A partir desta matriz de pixels que representa a imagem, diversos tipos de processamento digital podem ser implementados por algoritmos computacionais. A aplicação destes algoritmos realizam as transformações necessárias para que se possa, por exemplo, obter uma imagem com os realces pretendidos ou extrair atributos ou informações pertinentes. (ESQUEF et al., 2003).

Figura 9. *Imagem monocromática "Goldhill" com destaque para uma região de 17 x 17 pixels. fonte: (ESQUEF et al., 2003)*

O processamento de imagem digital começa com uma imagem e produz uma nova versão dessa imagem. (SHIH, 2017). Considerando-se que uma nuvem de pontos pode ser abstraída como uma imagem, a alteração em uma nuvem de pontos gerando uma versão modificada é, em essência, processamento de imagem digital. Em diversos casos um pré-processamento é necessário para preparar a imagem para o processamento em si.

2.4. Classificação de Imagens

Em se tratando de automação veicular, é esperado que um sistema autônomo consiga classificar objetos, reconhecendo a diferença por exemplo entre um ser humano e um poste, já que levando em conta a necessidade de tomada de decisão por veículos autônomos, em situações críticas há cenários em que é viável optar por colidir contra um poste ao invés de colidir contra um ser humano. Para esse tipo de tomada de decisão é essencial que o sistema autônomo consiga classificar objetos, atribuindo a cada classe, métricas específicas que auxiliem nas tomadas de

decisões. Segundo Chris Solomon (2011) A classificação de imagens é um campo amplo e amplamente estudado que pertence mais adequadamente dentro da disciplina de reconhecimento de padrões do que no processamento de imagens. A classificação de imagens foge ao escopo desse projeto final de graduação, no entanto espera-se que o projeto seja levado adiante por outros colegas, com a inclusão desses algoritmos. Um exemplo da utilização da classificação de imagens pode ser visto na Figura 10, onde um sistema classificador identifica um motociclista

Figura 10. *Sistema de automação veicular da Volvo utilizando classificação de imagens. fonte: (Aaron, 2016)*

2.5. Nuvem de pontos

A representação de dados em três dimensões no contexto do desenvolvimento de veículos autônomos é necessário para que seja possível representar de maneira adequada os dados relativos aos ambientes em que se encontram estes veículos. Entre as possibilidades para a representação de objetos 3D, a utilização de nuvem de pontos tem sido a principal abordagem para esta aplicação. Uma nuvem de pontos é uma coleção discreta de pontos em um sistema de coordenadas espaciais, como o sistema de coordenadas cartesiano (X,Y e Z) (Dumic et al., 2018). Desta maneira, as nuvens de pontos podem ser utilizadas para

representação de vários objetos como carros, pedestres, ciclistas, placas de trânsito, entre outros. A obtenção de nuvem de pontos pode ser feita de diferentes maneiras como através de sensores ópticos a laser, sensores de luz estruturados e LiDAR (*Light Detection And Ranging*). As Figuras 11 e 12 mostram exemplos de nuvem de pontos.

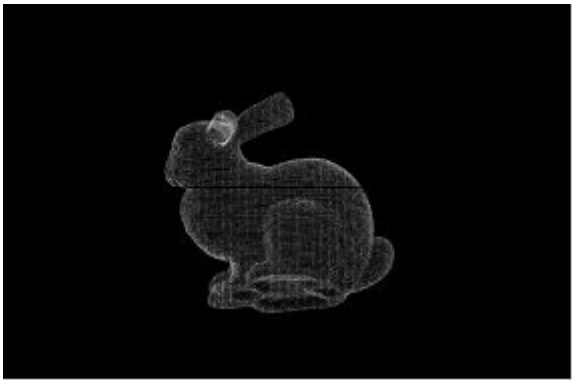

Figura 11. *Nuvem de pontos de um coelho. fonte: (DUMIC et al., 2018).*

Figura 12. *Nuvem de pontos longdress. fonte: (DUMIC et al., 2018).*

2.6. Propriedade de nuvens de pontos

A tecnologia de nuvem de ponto se trata da representação de objetos tridimensionais através de um conjunto de pontos expressos em um sistema de coordenadas que na maioria dos casos tem o intuito de representar a estrutura física do objeto. Além das três dimensões espaciais, é possível utilizar outros parâmetros para representar características de um ponto específico de uma nuvem de pontos, essas características podem ser representadas através da introdução de cores ou escala de cinza, podendo representar:

A própria cor do ponto. Essa característica é a mais simples e imediata de ser definida, já que representa diretamente a característica observada na estrutura física do objeto. A cor é definida por três valores variando de 0 a 255, cada valor representando a intensidade de uma cor: Vermelha, verde e azul. Esses três valores são combinados para formar a cor do ponto. O modelo de cores RGB é baseado na teoria de visão colorida tricromática, de Young-Helmholtz (YOUNG, 1800), e no triângulo de cores de Maxwell (MAXWELL, 1857) e (JUDD, 1935)

A cor do ponto em escala de cinza. Essa característica é resultado de um cálculo que relaciona uma determinada faixa do espectro visível à intensidade da luz em cada pixel. Nesse caso a cor é definida por um único valor variando de 0 a 255, em que 0 representa a cor preta e 255 representa a cor branca, os valores intermediários são escala de cinza proporcionais à distância dos extremos.

Característica diversa do ponto que também apresentada em formato RGB (no caso de três características distintas) ou escala de cinza (no caso de uma única característica). Nesse caso o ponto apresentará falsa cor, ou seja. A cor do ponto representado na nuvem de pontos não representa a cor real do ponto, mas sim outras propriedades de interesse como por exemplo a distância do ponto a um dado referencial, a refletância do material, a temperatura, etc.

2.7. SEGMENTAÇÃO DE nuvem de ponto

Segmentação de nuvem de pontos é o processo de classificação de nuvens de pontos em várias regiões homogêneas, nas quais os pontos na mesma região terão as mesmas propriedades. A segmentação é desafiadora devido à alta redundância, densidade de amostragem desigual e falta de explícita estrutura de dados da nuvem de pontos. Esse problema tem muitas aplicações em robótica, como veículos inteligentes, mapeamento e navegação autônomos. Muitos autores introduziram diferentes abordagens e algoritmos. Em computação gráfica, pesquisas intensivas foram realizadas com o intuito de decompor o modelo 3D em regiões funcionalmente significativas. A maneira geral é criar um gráfico a partir da malha de entrada, e agrupar o gráfico para produzir uma segmentação usando informações

como direção normal, suavidade ou concavidade ao longo dos limites (NGUYEN e LE, 2013) A segmentação de nuvem de pontos pode ser considerada uma etapa de pré-processamento quando se deseja utilizar algoritmos de classificação de nuvem de ponto. O objetivo é definir uma região homogênea na nuvem de ponto que tenha relevância semântica, ou seja, que possua representatividade lógica, caracterizando, por exemplo, um rosto. No contexto da automação veicular a segmentação de nuvem de ponto poderia delimitar uma região pontos que caracterizam um carro para que posteriormente essa imagem seja utilizada num algoritmo de classificação, onde efetivamente será constatado que dentro daquela região existe um carro. A Figura 13 mostra tentativas de segmentação de objetos específicos, por exemplo um carro, um poste de luz, uma lata de lixo e uma placa utilizando diferentes algoritmos. Observa-se que o objetivo final é obter um conjunto de pontos semanticamente consistente (o objeto em questão).

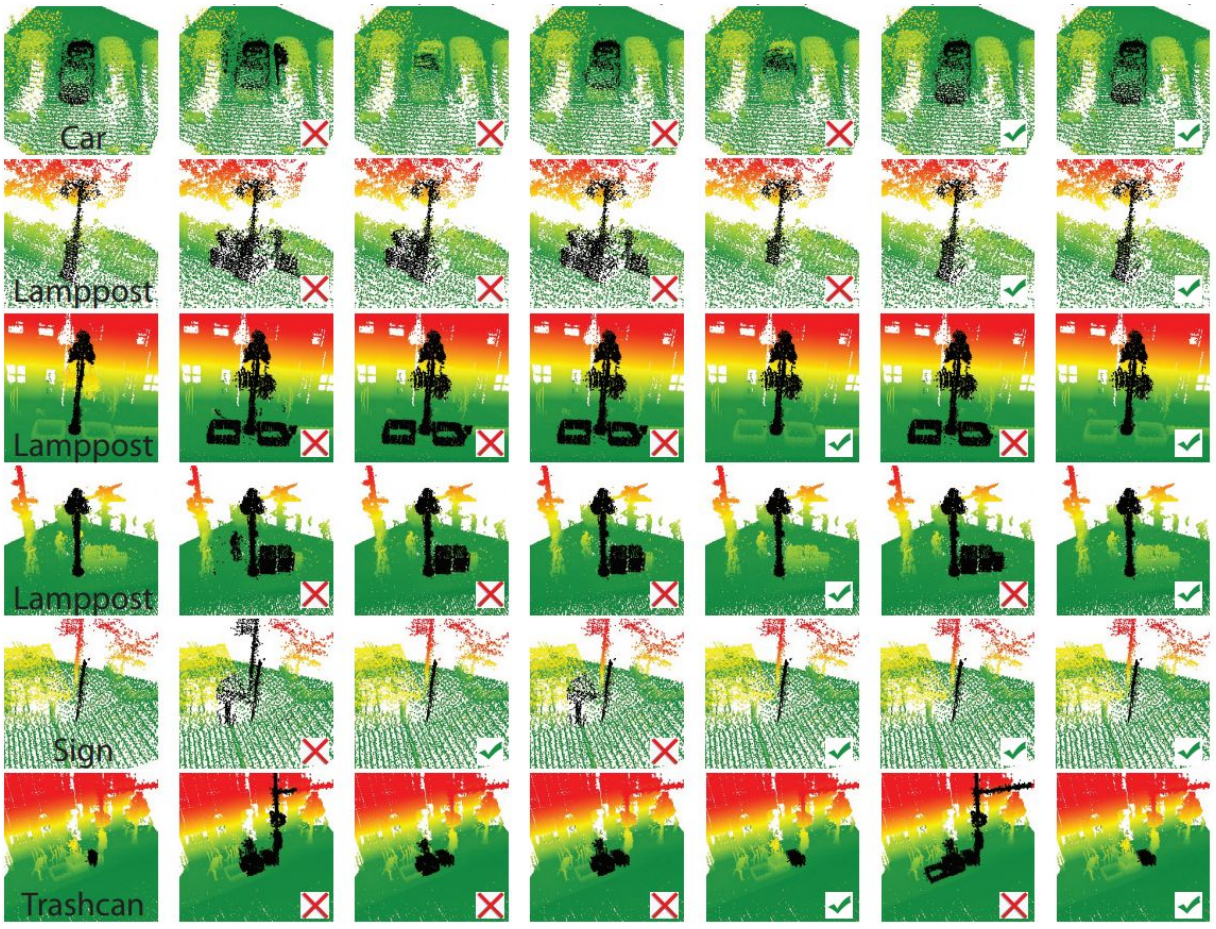

Figura 13. *Segmentação de nuvem de pontos. Fonte: (Golovinskiy e Funk, 2009)*

2.8. Classificação de nuvens de pontos

Classificação de nuvem de ponto consiste uma técnica de extração de informações a partir de análise de nuvens de pontos. Essa classificação costuma ser realizada por meio do reconhecimento de padrões gerados pelas diferenças espectrais entre os pontos analisados. (JENSEN, 2014). O reconhecimento dos padrões existentes em objetos representados em uma nuvem de pontos pode ser realizado de diversas maneiras. A maneira mais intuitiva é o reconhecimento de padrões através da análise humana, ou seja, análise visual de uma nuvem de pontos e inferência acerca do tipo de objeto existente em um determinado conjunto de pontos, classificando o objeto semanticamente de acordo com essa inferência. Esse tipo de análise subjetiva requer um tratamento personalizado de cada nuvem de ponto o que acaba exigindo um grande esforço do analisador, além disso a escalabilidade é bastante prejudicada já que não existe uma maneira automatizada de realizar o procedimento. A maneira mais eficiente é a utilização técnicas computacionais de aprendizado de máquina que por sua vez, apresentam uma série de vantagens sobre as técnicas de análise visual, particularmente em se tratando de nuvem de pontos complexas com grande quantidade de informações. Esse tipo de técnica é altamente escalável devido à sua natureza automatizada, utilizando-se da alta capacidade de processamento disponível no mercado. Os métodos de classificação automatizada de nuvem de pontos podem ser agrupados de acordo com o conhecimento a priori em supervisionado, não supervisionado e híbrido; conforme as métricas, em paramétrico ou não paramétrico, de acordo com a abordagem, por pontos ou regiões (objetos), como descrito na seção SCHOWENGERDT (2011). Em aprendizado de máquina, a abordagem supervisionada requer uma etapa de pré-análise, na qual os alvos a serem analisados são selecionados (segmentação) e amostras de referência são colhidas para alimentar o motor do aprendizado de máquina (classificador) nas iterações posteriores. (GONZALES e WOODS, 2018). Na abordagem não supervisionada, os pontos são agrupados em grupos que possuem características semelhantes. Nessa abordagem nenhum conhecimento prévio é introduzido no algoritmo, apenas as

similaridades de atributos são considerados, nem mesmo é indicado qual o alvo de interesse. Essa abordagem, portanto, é mais simples já que apenas os dados apresentados ao classificador são considerados. Na abordagem por pontos, os dados espectrais do ponto são utilizados, em detrimento dos dados da vizinhança. A abordagem por região agrupa os pontos vizinhos que possuem características similares ao ponto central analisado. (Jensen, 2014), as métricas usadas, são abordadas em Richards e Jia (1999). Para a classificação, considera-se o modelamento de classes por uma função normal, dessa maneira realiza-se uma estimativa da função densidade de probabilidade para cada classe e então, verifica-se a correspondência de cada ponto analisado em relação às possíveis classes. Com base na resposta de um dado ponto a essa função, é possível associá-lo ou não a uma das classes. De modo geral, os classificadores são condicionados a critérios de decisão, os quais fazem parte de um sistema hierárquico. Esse sistema deve especificar as classes que serão extraídas bem como os critérios usados para diferenciá-las. (RICHARDS, 2006).

2.9. Mapas de alta definição

Automação veicular completa, ou seja, veículos auto-dirigidos requer um sistema de controle inteligente que consiste em sensores de alto desempenho, já que não há interação humana para apoiar as decisões desses veículos. Os requisitos para um sistema desse tipo devem obviamente refletir a capacidades que os humanos precisam para dirigir um veículo. Simplesmente identificar onde as estradas estão é insuficiente para a condução autônoma. Um carro robótico deve ter a capacidade de detectar e evitar obstáculos. Além de obstáculos imóveis, as estradas contêm tráfego com participantes em movimento dinâmico, como outros carros e, principalmente em áreas urbanas, também há os mais frágeis: pedestres e ciclistas. Especialmente em situações críticas, o sistema de detecção e controle de um veículo autônomo deve ter tempos de reação rápidos. Um carro robótico requer uma variedade de tecnologias de sensores, como dispositivos de sonar, câmeras estéreo, lasers, radar etc. Todas essas tecnologias têm diferentes pontos de vista, e

cada tecnologia tem um propósito dedicado que é comparável a um ou mais dos cinco sentidos humanos. (Nagy e Benedek, 2019).

Mapas de alta definição são formados por capturas tridimensionais do ambiente no qual se encontra o veículo autônomo, fornecendo informações com precisão sobre a posição dos elementos que compõem uma via, como pedestres, veículos, placas de trânsito, sinalizadores e outros. Os mapas de alta definição são obtidos através de sensores como o LiDAR, e podem ser representados através de nuvem de pontos.

2.10. LiDAR (Light Detection And Ranging)

A base do sensoriamento remoto de um carro robótico é o sensor LiDAR (sigla para detecção de luz e alcance). Esse sensor é bastante semelhante a um radar, com a diferença de que ele envia e recebe pulsos de luz ao invés de ondas eletromagnéticas. Uma vantagem do sistema LiDAR em relação aos sistemas tradicionais de sensoriamento é a sua precisão, sendo capaz de realizar mais de dois milhões de leituras por segundo, gerando uma nuvem de ponto densa o suficiente para gerar imagens em alta resolução. Com um alcance de até 100m e rotação de 360 graus.

A função básica de um sistema Lidar é a medição da distância. Esse processo é realizado a partir da emissão de pulsos eletromagnéticos, que são detectados ao serem refletidos pelos obstáculos do ambiente. A partir da diferença de tempo entre emissão e recepção e dada a velocidade da luz, a distância ao objeto é calculada.

> Na Alemanha a tecnologia LiDAR é utilizada em diversas pontes de pedágio. As pontes inteligentes têm sido Desenvolvidas e fornecidas pela empresa VITRONIC GmbH, nas quais os veículos que se aproximam são registrados por um LIDAR e rastreados até a passagem. Com um LIDAR adicional, projetado para curto alcance, é determinado um contorno 3D dos veículos, a fim de diferenciar o caminhão dos outros veículos. (Picand e Dutoit, 2012, tradução nossa)

Além disso essa mesma empresa tem criado diversas outras aplicações com LiDAR para o monitoramento de trânsito, incluindo monitoramento de velocidades

diferentes para diferentes faixas, e velocidades diferentes para diferentes tipos de veículos (caminhões, carros, motos, etc.), com uma distância mínima de até 75 metros do sensor, que é posicionado na beira da estrada. O sensor tem alta precisão independente do alto tráfego de veículos, da distância entre veículos que andam muito próximos, de veículos mudarem de faixa (VITRONIC, 2016). Uma representação deste sistema pode ser vista na Figura 14

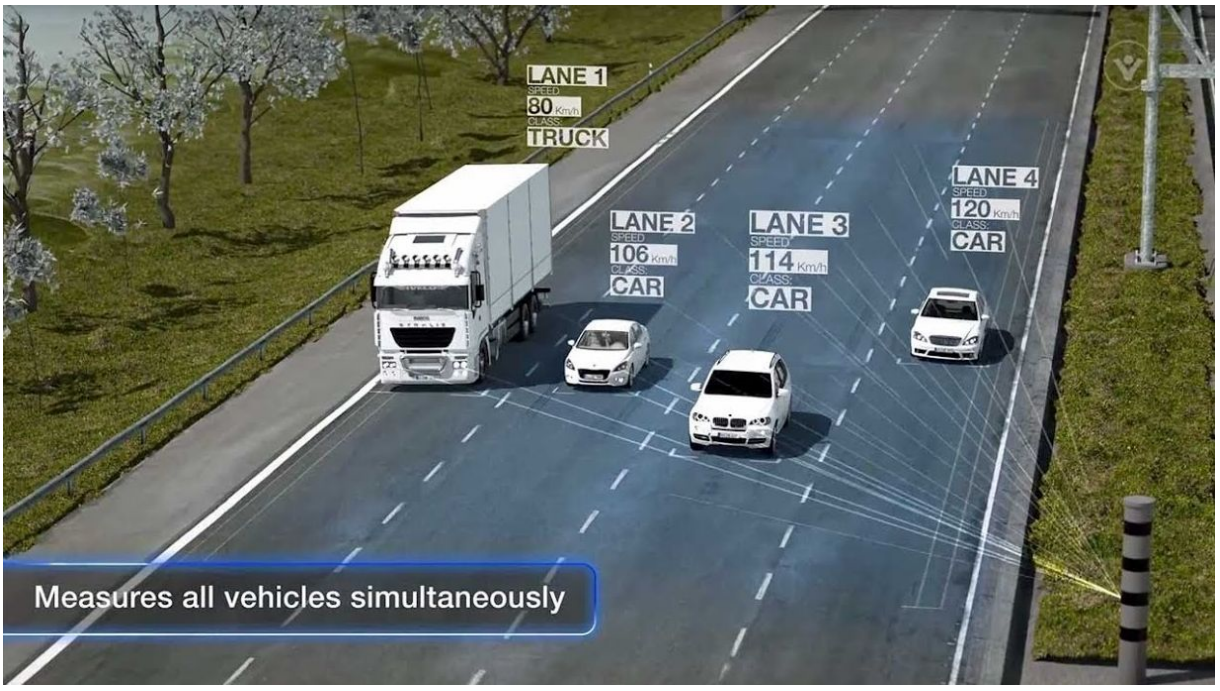

Figura 14. *Sensor LiDAR monitorando velocidade de diferentes classes de veículos em diferentes faixas. Fonte: (VITRONIC, 2016)*

Segundo Schwartz (Julho de 2010) O desempenho do LIDAR de alta definição permite que o sistema de navegação de um veículo que viaja a 65 km/h identifique automaticamente um Objeto de 15 cm de distância a 50m e ainda tem tempo suficiente para navegar em torno dele. Para manter um baixo custo o sensor utiliza lasers comerciais de comprimento de onda de 905nm. Lasers comerciais também são benéficos por seu baixo consumo de potência e segurança ocular classe 1 além de facilidade de calibração e teste de qualidade dos sensores.

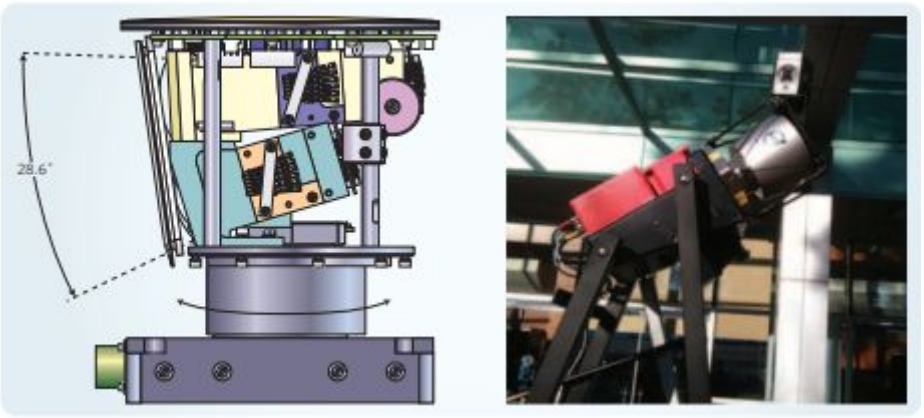

Figura 15. *Vistas interna (esquerda) e externa (direita) do LIDAR de alta definição da Velodyne. Fonte: (Schwarz, B. 2010)*

Enquanto a maioria dos sistemas LIDAR tem um único laser que se fixa em um espelho rotativo, o sensor LIDAR de alta definição da Velodyne usa uma cabeça rotativa com 64 lasers semicondutores, cada um emitindo até vinte mil vezes por segundo. Isso permite que o sensor alcance a coleta de dados em taxas de ordem de magnitude maior do que a maioria dos modelos convencionais. Cada um os 64 lasers têm seu próprio detector, e cada par de laser detector é precisamente alinhado em ângulos verticais predeterminados para obter um campo de visão vertical de 28,6 °. Ao girar a unidade inteira em velocidades de até 900 rpm (15 Hz) em torno de seu eixo vertical, é gerado um campo de visão de 360°. Mais de 1,3 milhão de pontos de dados são gerados a cada segundo independente da taxa de rotação. A nuvem de pontos resultante é tão densa que programas de computador podem identificar objetos como calçadas e fios aéreos em distâncias superiores a 100 m. A estrutura interna do LiDAR da Velodyne pode ser vista na Figura 15

2.11. Matlab

MATLAB é um software interativo de alta performance desenvolvido pela empresa MathWorks que visa a resolução e visualização de problemas e sistemas matemáticos. A característica que destaca o MATLAB é a sua unidade fundamental: a matriz numérica retangular, podendo conter elementos complexos (lembrando que um escalar é equivalente a uma matriz 1x1 e que um vetor é uma matriz linha ou

coluna). Todo o ferramental do software, além de todas as operações utilizadas são abstraídos de operações matriciais simples.

Originalmente, o MATLAB foi desenvolvido no final dos anos 70 na Universidade do Novo México. Foi desenvolvido com o intuito de facilitar a compreensão de álgebra linear para alunos que não possuíssem habilidades em programação. Para os estudantes, o software ainda está disponível hoje com menor custo. A versão de estudante foi utilizada para a realização desse projeto final de graduação.

A grande força do MATLAB é o manuseio e o cálculo com matrizes. O software combina análise numérica, cálculo matricial, processamento e visualização de sinais. É uma linguagem de programação interativa que funciona orientada a objetos. O MATLAB é independente de plataforma, e é usado para resolver problemas estatísticos, de controle, mecânicos, econômicos, de análise numérica, cálculo, processamento de sinais e construção de gráficos. O MATLAB compete com softwares disponíveis gratuitamente, como GNU Octave, FreeMat ou as bibliotecas Python Matplotlib e NumPy.

A programação com o MATLAB ocorre em uma linguagem de programação proprietária. Ele pode ser interpretado em diferentes sistemas operacionais e possibilita a implementação de programas menores como funções ou scripts em unidades separadas. A sintaxe é fácil de aprender e desenvolvida para facilitar a escrita de processos matemáticos. As interfaces permitem a integração de outras linguagens de programação como C. Para programação orientada a objetos com MATLAB, os conceitos típicos de classes, pacotes, herança e chamadas de valor por chamada estão disponíveis. Além da linguagem de programação proprietária, o software oferece um ambiente gráfico de desktop. O ambiente facilita a visão geral das variáveis e o código de programação. (MATHWORKS, 2019)

2.12. Fórmula de Haversine

A fórmula de Haversine (equação 1) é uma importante ferramenta utilizada em navegação, fornecendo a distância entre dois pontos de uma esfera através de sua

latitude e longitude. A função de Haversine é definida como $haversin(\theta) = seno^2(\theta/2)$ e partir desta, é possível calcular a distância entre dois pontos na superfície da terra.

$$D = 2rsen^{-1}(sen^2((\Phi_1 - \Phi_2)/2)) + cos(\Phi_1)cos(\Phi_2)sen^2((\Psi_1 - \Psi_2)/2)))^{1/2}$$

Equação 1. Fórmula de Haversine

D é a distância entre os dois pontos sob a esfera de raio *r*, Φ e Ψ sendo respectivamente latitude e longitude dos dois pontos em que se deseja descobrir a distância. (Mangesh e Chopde, 2013)

2.13. Voxel

Voxels são análogos aos pixels, porém são utilizados para figuras tridimensionais.

> Seja Z^3 o subconjunto do espaço euclidiano tridimensional R^3 que consiste em todos os pontos cujas coordenadas são números inteiros. Esse subconjunto é chamado de treliça inteira ou grade, para abreviar. A vizinhança Voronoi de um ponto p de uma grade é o conjunto de todos os pontos em R^3 que são pelo menos tão próximos de p quanto a qualquer outro ponto da grade. A vizinhança Voronoi de uma grade é um cubo unitário fechado e alinhado ao eixo, conhecido como voxel. (Cohen-Or e Kaufman, 1995)

Ou seja, voxels são unidades volumétricas igualmente espaçadas que existem ao redor de pontos centrais discretizados.

O mais comum é se trabalhar com volumes cúbicos com lados de dimensão igual a uma potência de 2, e resolução espacial igual a 1. isto é, cada voxel tem dimensões iguais a 1 × 1 × 1 e ocupa uma posição inteira em uma grade tridimensional regular de dimensões $Z \times Z \times Z$ (em que $Z = 2L$, $L \in N$), capaz de conter até Z^3 voxels. (Torlig, 2018)

> A representação via voxels tem várias implementações e vantagens de processamento em tempo real sobre outras representações 3D, como menor complexidade de renderização (por exemplo, em oposição a malhas

> triangulares) e remoção de redundâncias e inconsistências de mapas de sensores sobrepostos (GARCIA et al., 2019)

Um voxel é equivalente a um pixel para uma representação tridimensional. Enquanto polígonos são comumente representados pelas coordenadas de seus vértices, sendo eficientes na representação de estruturas tridimensionais simples com muitos espaços vazios ou preenchimento homogêneo; a posição de um voxel é inferida de acordo com a sua distância de outros voxels. Sendo assim, os voxels são particularmente eficientes em representar espaços amostrados regularmente com preenchimento não homogêneo.

2.14. Voxelização

Nuvem de pontos geradas por sensores de mercado como o LiDAR, não são voxelizadas, ou seja, os pontos gerados pelo sensor não são discretizados, pertencendo à totalidade do espaço euclidiano R^3.

O processo de modificação da representação de uma nuvem de ponto para o formato de voxels, é denominado voxelização. Esse processo é realizado percorrendo cada voxel do volume de representação e atribuindo ao mesmo um valor de cor dependendo dos pontos que ocupam posições contidas em seu volume. Voxels não ocupados não tem cor atribuída, equivalente a serem completamente transparentes. No caso de mais de um ponto se encontrar ocupando um mesmo voxel, a cor atribuída a tal voxel é calculada como a média dos pontos em seu interior. É de especial interesse a restrição do conteúdo voxelizado à posições inteiras pela semelhança com como imagens bidimensionais são representadas. Isso permite a visualização de maneira simples e rápida do conteúdo em um contexto bidimensional, através da projeção do conteúdo tridimensional. (Torlig, 2018). Um exemplo didático de voxelização pode ser visto na Figura 16. Um exemplo prático de voxelização pode ser visto na Figura 17.

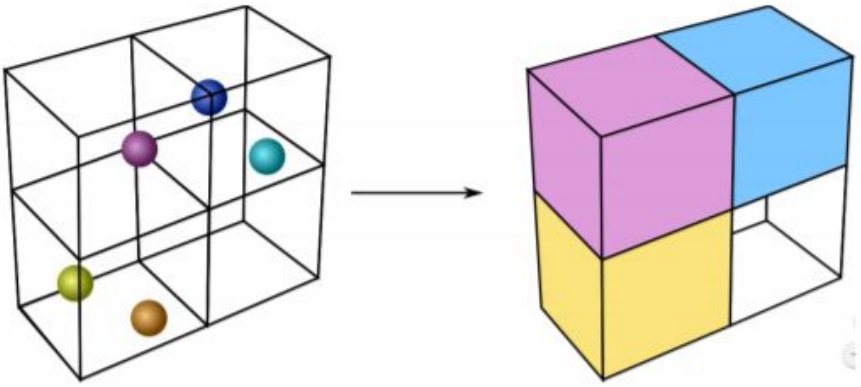

Figura 16. *Exemplo didático de voxelização. Fonte: (Torlig, 2018)*

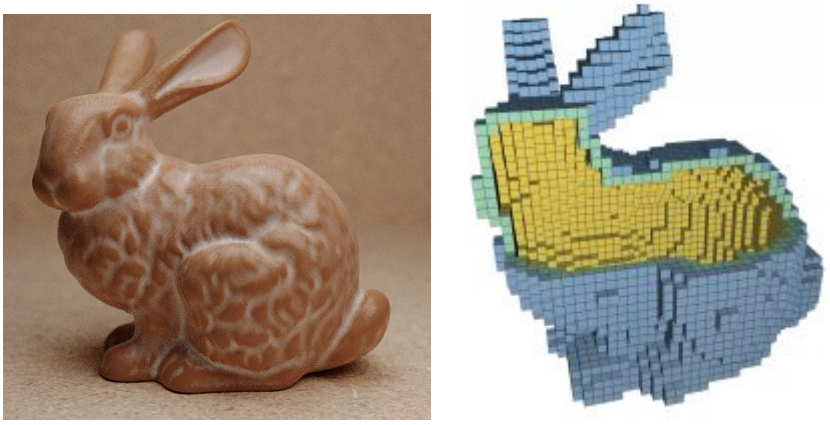

Figura 17. ***Exemplo*** *prático de voxelização. Fonte: (Schwarz, M. e Seidel, 2010)*

2.15. Algoritmos de Compressão de imagem

2.15.1. Quadtree

Quadtree é a representação em árvore de um algoritmo de compressão utilizado em imagens bidimensionais. Para uma imagem de $2^n \times 2^n$, sendo *n* a resolução da imagem, é possível realizar a segmentação da imagem em quatro quadrantes recursivamente até alcançar a resolução pretendida. As regiões com mais detalhamento na imagem são representadas apresentam maior número de divisões recursivas, enquanto regiões homogêneas utilizam menos segmentações, a identificação e tratamento dessas regiões com informação redundante é o cerne da compressão. A Figura 18 é uma figura composta apenas de pontos, apresentando um exemplo simples para o entendimento de Quadtrees. A figura foi representada

com diferentes níveis. o número relativo do nível indica o incremento da quantidade de segmentações recursivas realizadas na imagem. Nessa Figura podemos observar que inicialmente (nível *N*) não havia segmentações em quadrantes, portanto no caso de uma transmissão, toda a Figura teria que ser enviada, mesmo havendo diversas áreas em branco. No nível *N*-1 foi utilizada apenas uma segmentação em quatro quadrantes. É possível perceber, no entanto, que apenas uma segmentação não é suficiente para expor as redundâncias da imagem, que continua com todos os quadrantes ocupados, ou seja, no caso de uma transmissão, toda a imagem teria que ser enviada ao destinatário para que não houvesse perda de informação. Já nos casos dos níveis *N*-2 e *N*-3, diversos pixels aparecem vazios, não necessitando ser transmitida a informação detalhada, apenas metadados informando essa redundância nos quadrantes específicos. Inclusive caso o quadrante pai seja vazio, não ocorre a segmentação em quadrantes filhos. Quanto maior for o número de iterações realizado, maior o número de redundâncias expostas, já que os quadrantes ocupados terão maior densidade de informações. Isso faz com que haja uma troca de pré-processamento por eficiência na transmissão. No caso de uma figura complexa, com muitos detalhes, a abordagem é outra. Nesse caso todos os quadrantes estarão sempre ocupados, no entanto nem todos terão grande riqueza de detalhes.

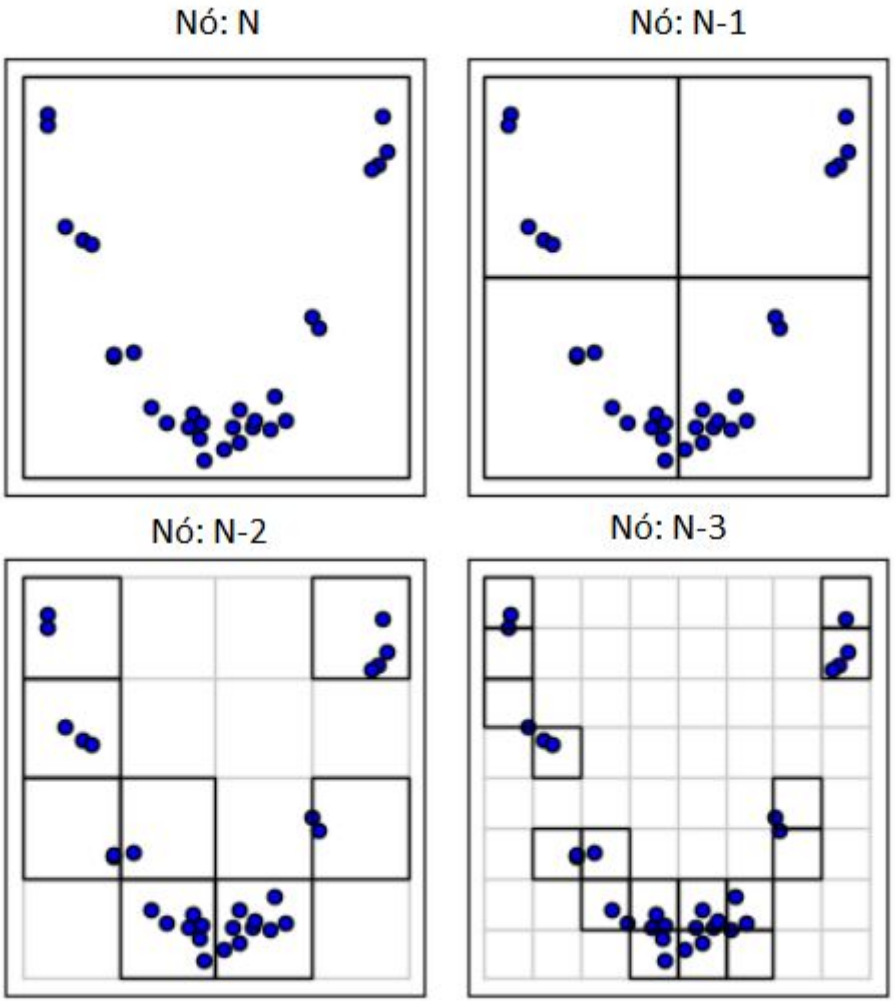

Figura 18. *Quadtrees: Diferentes níveis de segmentação em quadrantes. Fonte: (Ivezic et al., 2014)*

Com imagens complexas um grande número de iterações gera maior resolução da imagem, possibilitando observar mais detalhes, já que regiões menos detalhadas não necessitam ter uma resolução muito alta, vide Figura 19. Isso ocorre porque neste algoritmo não é obrigatório realizar a segmentação em todos os quadrantes simultâneamente. Aqueles quadrantes que possuem maior número de detalhes podem ser segmentados para aumentar a resolução, enquanto os quadrantes que possuem melhor número de detalhes, caso a resolução já atenda a demanda, não precisa ser segmentado.

Figura 19. *Algoritmo de compressão Quadtree, da esquerda para a direita: Imagem original, imagem comprimida e imagem comprimida com visualização das grades*

A manutenção de informações da imagem, apesar da compressão, é uma das vantagens mais relevantes desse algoritmo. Este esquema foi projetado com base no princípio de compactação de dados sem perdas (Liu e Chang, 1995). O nome do algoritmo se deve à uma representação em forma de árvore, composta por nós pai e filhos, sendo que os últimos nós são as folhas. Cada nó não-folha no quadtree possui quatro filhos. O nó raiz é rotulado nível n, correspondente à imagem inteira e a cada nó filho de nível *n*-1 contém um quarto do tamanho da imagem do pai. Essa representação é apresentada na Figura 20 abordada em (Liu e Chang, 1995).

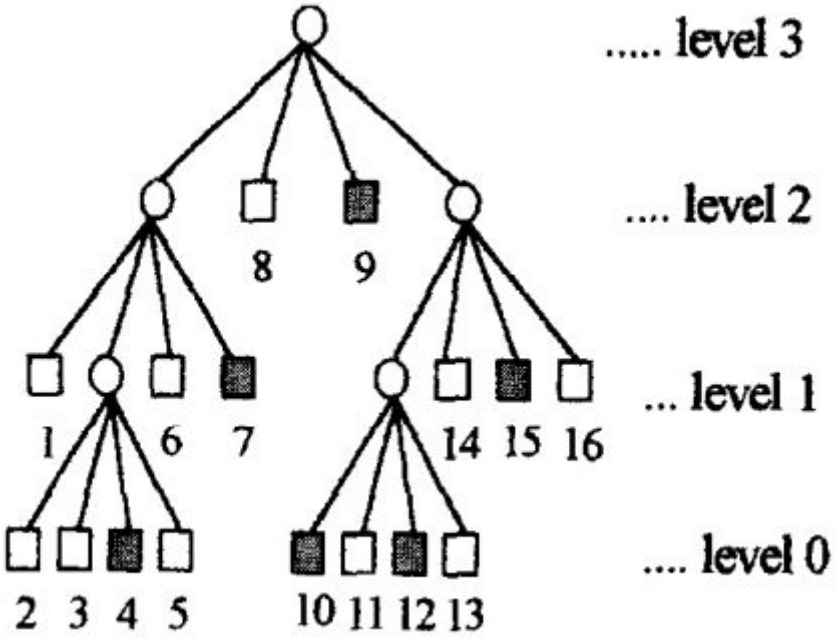

Figura 20. *Representação em árvore do algoritmo Quadtree. Fonte: (Liu e Chang, 1995)*

2.15.2. Octree

Uma octree é a representação em árvore de um algoritmo de compressão utilizado em imagens tridimensionais. Enquanto um algoritmo quadtree realiza iterações de segmentações de quatro em quatro pixels, um algoritmo octree realiza iterações de segmentações de oito em oito voxels. O número de níveis determina a precisão da quantização e a resolução da imagem tridimensional. Uma octree de profundidade *L* fornece uma precisão de *L* bits por direção da coordenada. Analogamente aos algoritmos quadtree, o esquema de compactação é sem perdas, no sentido de que essas coordenadas quantizadas são preservadas durante a compactação. A Figura 21 à esquerda apresenta o espaço tridimensional (cubo) sem nenhum tipo de pré-processamento. Esse espaço pode potencialmente ser ocupado por uma imagem tridimensional ou uma nuvem de ponto. À direita é apresentado o processo de segmentação via octree de um cubo. Vale ressaltar que podem ser realizados diferentes níveis de segmentação para cada parte do objeto, vide a área 5, que foi segmentada em um nível a mais do que o resto do cubo. e a Figura 22 apresenta a representação em árvore octree do cubo da Figura 21

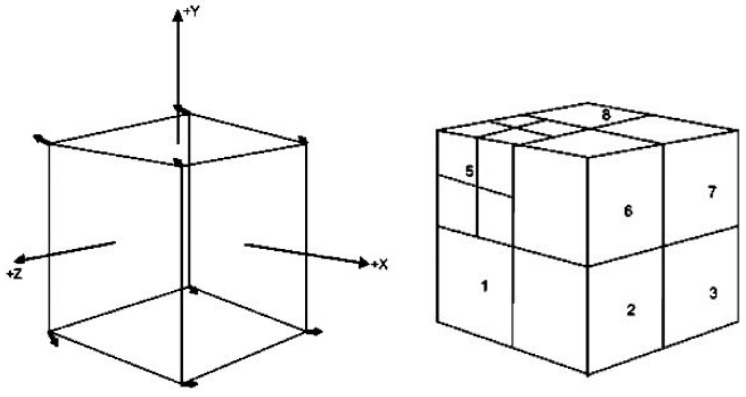

Figura 21. *Esquerda: Cubo após o processo de segmentação via octree. Direita: Cubo e suas três dimensões. Fonte: (Zhang e Owen, 2005).*

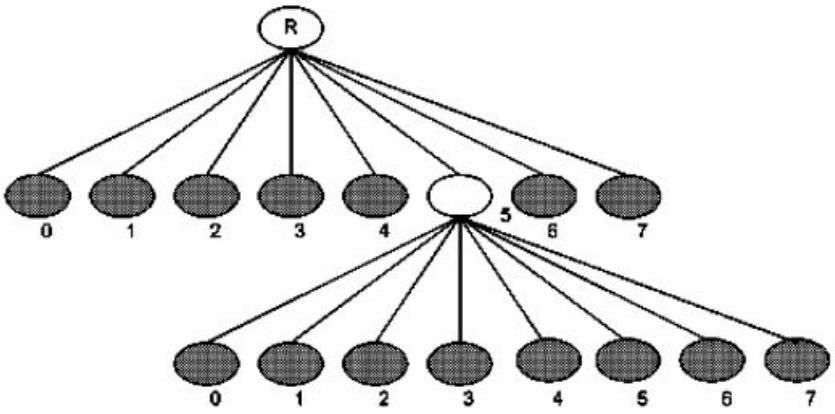

Figura 22. *Representação em árvore octree do cubo segmentado da* Figura 21. *Fonte: (Zhang e Owen, 2005).*

2.16. Polygon File Format

Polygon File Format, também conhecido como *Stanford Triangle Format* é um formato de arquivo para representação gráfica de objetos formados por um conjunto de polígonos. Também conhecido como *PLY*, este tipo de arquivo define um objeto como um conjunto de vértices, faces e outros elementos, e podem ser definidas sobre estes propriedades como cor, reflectância e direção normal. Em geral, arquivos deste tipo contém uma tupla com a lista de vértices e suas propriedades, sendo uma das mais utilizadas a informação sobre cor no formato RGB, com intensidade luminosa de 0 a 255 em cada canal de cor (vermelho,verde e azul).

Este formato de arquivo é comumente utilizado para armazenar capturas de nuvem de pontos. Para que seja possível utilizar este formato, é necessário um cabeçalho com informações em relação a quantidade de vértices que contidos no arquivo, além da definição dos eixos X, Y e Z, contendo o tipo do dado que está sendo armazenado (*float,char,double).* Para definir outras propriedades para cada ponto, como cor ou reflectância, deve ser inserido no cabeçalho esta informação.

3. DESENVOLVIMENTO

3.1. Introdução

Para a primeira contribuição, foi desenvolvido um método para criação de um mapa de alta definição de uma determinada região a partir de capturas de um sensor

LiDAR em diferentes instantes de tempo. Esse mapa de alta definição é resultado da sobreposição de nuvens de pontos espaçadas temporalmente. Serão apresentados todas as etapas de criação do método, além dos resultados obtidos e discussões relevantes sobre o tema.

A segunda contribuição consiste na voxelização e compressão da uma nuvem de pontos, diminuindo a quantidade de dados a serem enviados.

3.2. Obtenção de dados

Utilizou-se como referência para todo o trabalho prático uma série de nuvens de pontos geradas por um Velodyne HDL-64E que obteve os dados através de um percurso na cidade de Karlsruhe na Alemanha. Estes dados foram obtidos pelo *dataset* “The KITTI Vision Benchmark Suite” (Geiger et al., 2013). A escolha por um *dataset* disponível ocorreu devido alto custo de um sensor deste tipo, além da da dificuldade em gravar esta quantidade de dados.

3.3. Preparação de dados

O KITTI Vision Benchmark Suite (Geiger et al., 2013) disponibiliza várias versões de seus dados. Neste projeto, foi utilizado a versão não sincronizada e não retificada, pois é disponibilizado arquivos de texto com os dados das nuvens de pontos, dados de sensores como latitude e longitude, velocidade e aceleração sobre cada um dos eixos, entre outros. Também é disponibilizado pelo KITTI dataset os *timestamps* de cada captura obtida, tanto das nuvens de pontos, quanto das capturas dos sensores. Foi utilizado a versão não sincronizada e não retificada pois esta versão contém arquivos de texto que são mais simples de serem trabalhados e podem ser convertidos para o formato *ply*, formato comum para manipular nuvem de pontos. Para converter este arquivo de texto no formato *ply* basta adicionar um cabeçalho. A Figura 23 mostra um exemplo de cabeçalho que foi adicionado ao arquivo de texto para que pudesse ser convertido no formato *ply*.

```
ply
format ascii 1.0
element vertex 120570
property float x
property float y
property float z
property float reflect_coeff
end_header
```

Figura 23. *Exemplo de cabeçalho .ply*

O cabeçalho *ply* deve começar com uma linha contendo o nome *ply*, em seguida deve conter uma linha contendo format <formato dos dados> . Para dados de arquivo de texto o *ply* utiliza o formato ascii 1.0. A linha seguinte deve conter element vertex <vértices> contendo o número total de vértices neste arquivo, que foi obtido a partir do número de linhas que o arquivo possui antes de adicionar o cabeçalho, visto que cada linha representa um vértice da nuvem de pontos. Em seguida devem ser adicionados as propriedades x,y e z e com o tipo de formato do dado, em que neste caso é do tipo float. Além destes, é definido também uma propriedade denominado *reflect_coeff*, pois os arquivos obtidos apresentam uma quarta dimensão contendo a reflectância, ou seja, a quantidade de luz que foi refletida na captura do sensor LiDAR. Após adicionar este cabeçalho, basta mudar a extensão do arquivo para *ply* que o arquivo já está convertido e pronto para ser utilizado.

3.4. Leitura e representação dos dados

Com os arquivos convertidos para o formato de *ply*, as nuvens de pontos já estão pronta para ser manipuladas e processadas. Cada conjunto de nuvens de pontos obtidas no *dataset* representam a captura de uma determinada via da cidade de Karlsruhe localizada na Alemanha. Estas nuvens de pontos estão espaçadas temporalmente em 0,1 segundos. O processo de leitura de nuvem de pontos inclui transformá-la em uma matriz para que possa realizar operações sobre esta.

As nuvens de pontos obtidos pelo *KITTI dataset* (Geiger et al., 2013) são representadas utilizando o espaço euclidiano tridimensional com a origem centrada na posição em que se encontra o sensor Velodyne HDL-64E. O eixo z do espaço

euclidiano é perpendicular a superfície da terra, o x é paralelo ao eixo do veículo e o y é perpendicular ao x, como pode ser verificado na Figura 24.

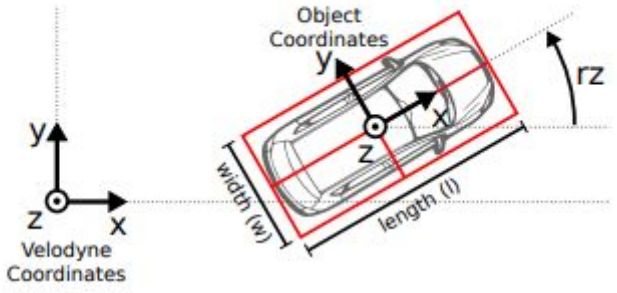

Figura 24. *Sistema de coordenadas do Velodyne HDL-64E. Fonte: (Geiger, 2013).*

3.5. Cálculo da distância

Como dito anteriormente, as nuvens de pontos utilizadas foram obtidas durante o percurso de um veículo na cidade de Karlsruhe com espaçamento temporal de 0,1 segundos entre cada um dos quadros. Montar uma nuvem de pontos global para esta cena, isto é, tornar múltiplas nuvens de pontos espaçadas temporalmente em única nuvem de pontos, exige alguns desafios. Estes diferentes quadros de nuvens de pontos apresentam dois tipos de objetos: estáticos e dinâmicos. Objetos estáticos comuns são: placas de trânsito, prédios, veículos estacionados. Objetos dinâmicos comuns são: pedestres, ciclistas, veículos em movimento. Objetos estáticos são redundantes visto que não se deslocam durante o tempo e estão sempre na mesma posição, independente do quadro. Como o veículo que realiza a captura da nuvem de pontos está em movimento, a posição relativa desses objetos está sempre mudando. Para manter objetos estáticos sempre na mesma posição em quadros consecutivos é necessário deslocar uma das nuvens de pontos em relação à outra. Neste projeto optou-se por deslocar a nuvem de pontos do quadro anterior (quadro 1) em relação ao quadro mais recente (quadro 2), no entanto vale ressaltar que deslocar no sentido contrário produziria os mesmos resultados. A solução utilizada não considera situações em que o veículo com o sensor LiDAR tenha um deslocamento considerável em relação aos eixos y e z (ver Figura 24), ou seja, apresentem velocidade não-desprezível nessas direções,

possuindo deslocamento apenas no eixo x. Considerando então que os objetos estáticos devem estar na mesma posição entre dois quadros, é preciso descobrir o quanto o veículo com o LiDAR se deslocou neste período. O *KITTI dataset* (Geiger et al., 2013) fornece informações relacionadas à latitude e longitude do veículo durante a captura pelo Velody HDL 64-E e a partir da fórmula de Haversine (2.11) foi possível calcular este deslocamento. O diagrama da Figura 25 apresenta esse procedimento.

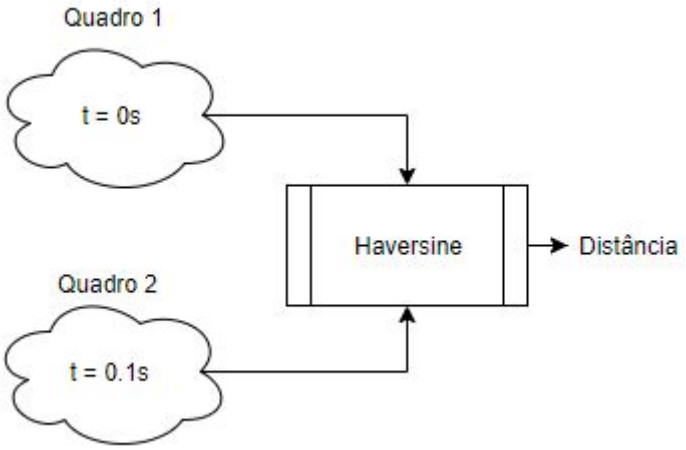

Figura 25. *Obtenção da distância com Haversine*

A partir da distância calculada, realizou-se o deslocamento da nuvem de pontos do quadro anterior em relação ao atual, fazendo com que objetos estáticos fiquem na mesma posição. Objetos dinâmicos se movimentam durante este período e por isso não ficarão na mesma posição. Após realizar o deslocamento baseado na fórmula de Haversine, realizou-se a sobreposição as duas nuvens de pontos, do quadro anterior com o quadro mais atual, criando uma nova nuvem de pontos que contém todos os pontos das outras duas. Como foi realizado um processo de deslocamento da nuvem de pontos do quadro anterior antes de realizar a sobreposição, a nova nuvem de pontos apresentará os objetos estáticos na mesma posição e objetos dinâmicos em posições diferentes, visto que estes se deslocaram durante a captura das duas nuvens de pontos. Cada novo quadro é realizado o mesmo o processo, sempre fazendo a sobreposição deste quadro mais recente com a nuvem de pontos global contendo todos os outros quadros.

3.6. Cálculo de mudanças

Ao sobrepor nuvens de pontos de quadros diferentes, gerando uma nuvem de pontos global, os objetos estáticos ficarão na mesma posição, como descrito na seção anterior, e os objetos dinâmicos estarão em posições diferentes. Devido a necessidade de atualizar pontos que representam estes objetos que estão em movimento, desenvolveu-se um algoritmo para tratar este problema. O algoritmo descrito abaixo elenca os passos utilizados:

1. Reorganiza-se as duas nuvem de pontos, de forma que a matriz que as representam estão em ordem crescente no eixo x.

2. Para cada ponto da nuvem global, calcula-se o ponto mais próximo deste na nuvem do novo quadro, ou seja, o ponto com pequenas diferenças nos eixos y e z, e que esteja mais próximo em x.

3.Para que seja considerado que este ponto se deslocou entre os período de tempo entre estes dois quadros, a distância entre estes dois pontos deve ser superior a um limiar mínimo.

4. Um limiar máximo de distância em x é definido, para evitar que pontos que representam diferentes objetos sejam interpretados como se fossem o mesmo.

Além disso, vale ressaltar que este cálculo só é feito para os cem pontos mais próximos no eixo x para cada ponto da nuvem de pontos global, para diminuir o tempo de execução do algoritmo.

3.7. Voxelização

Uma outra contribuição desenvolvida é a otimização de nuvens de pontos para a transmissão. Como dito na seção 2.13, este tipo de abordagem apresenta muitas vantagens em relação a outros tipos de representação de modelos em 3D. A voxelização consiste numa representação em voxels de uma nuvem de pontos original, utilizou-se a voxelização nesse projeto devido à relativa facilidade de se encontrar informações e artigos relacionados ao tema, tratando especificamente do manuseio de nuvens de pontos. Além disso qualquer algoritmo atuando em uma

nuvem de ponto voxelizada é mais eficiente do que esse mesmo algoritmo atuando em uma nuvem de ponto não-voxelizada, já que a informação é discretizada, deixando de pertencer ao espaço cartesiano real em R^3 e passando a pertencer a Z^3, ou seja, há menos informações a serem tratadas e a densidade de informações relevantes é maior (a depender do método de voxelização utilizado ser coerente com o objetivo). Considerando que um voxel equivale a um pixel para a representação gráfica tridimensional, a voxelização não interfere na qualidade da representação, já que em uma imagem não voxelizada uma área específica de um pixel teria mais informações do que o necessário para sua representação.

Utilizou-se a mesma abordagem apresentada em (Dong et al., 2004), cujos resultados podem ser vistos na Figura 26. Essa abordagem consiste na divisão da nuvem de ponto em 2^n níveis no espaço Z^3, tal que quanto maior for n, maior será a resolução da nuvem de ponto resultante.

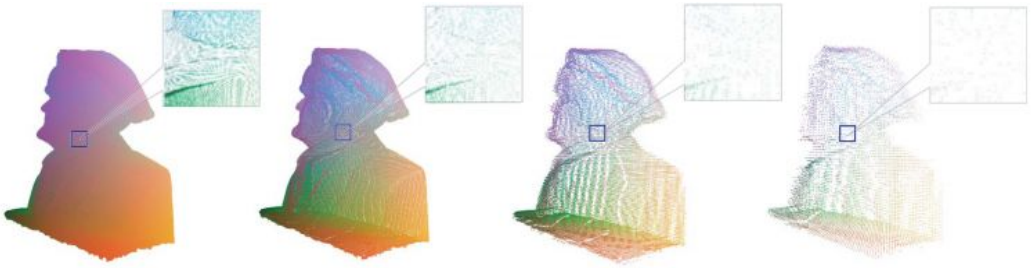

Figura 26. *Resultados de voxelização para o modelo Wagner sob diferentes resoluções. Da esquerda para a direita: 512^3, 256^3, 128^3, 64^3. Fonte: (Dong et al., 2004)*

3.8. Compressão

Ainda na segunda contribuição, após a voxelização, a nuvem de pontos passa por um processo de compressão. Optou-se por utilizar a compressão via método octree como abordado por Schnabel e Klein (2006). A compressão via octree é baseada na compressão geométrica (Botsch et al, 2002).

3.9. Conclusão

Esse tópico visa apresentar uma visão geral das duas contribuições propostas neste projeto. Inicialmente obteve-se os dados através do KITTI dataset, já

que consideradas as circunstâncias desse projeto, realizar a captura é financeiramente inviável. Os dados consistem em quadros de nuvem de pontos espaçados temporalmente em 0,1s. O tamanho médio de cada frame é 3 MB.

A fórmula de Haversine é utilizada para calcular a distância percorrida pelo carro, dados dois pares latitude e longitude.

Para obter uma nuvem de ponto global, com maior densidade de pontos, é necessário realizar a sobreposição de nuvem de pontos. A abordagem de apenas sobrepor as imagens sem nenhum tipo de tratamento obviamente gera distorções, já que o carro no qual está acoplado o sensor LiDAR se movimenta. Fazendo com que todos os objetos estáticos sejam apresentados de forma duplicada e os objetos que se movimentam na mesma direção do carro apresentam menor distorção, estando duplicados porém mais próximos. A abordagem pela fórmula de Haversine contorna o problema dos objetos estáticos, já que desloca o frame anterior pela distância percorrida pelo carro. Isso faz com que os objetos dinâmicos sejam distorcidos, sendo representados duplicados.

O quadro resultado é gerado após corrigir o problema da duplicação de objetos dinâmicos. Isso é feito através do cálculo de mudanças, que gera as distâncias percorridas pelos objetos dinâmicos através do algoritmo descrito na seção 3.6. O papel do algoritmo corrigir os objetos duplicados, movendo as coordenadas em “x” dos pontos do frame anterior para as coordenadas em “x” do ponto deslocado (frame atual). Idealmente o quadro resultado constitui uma nuvem de ponto que apresenta os objetos na mesma posição do frame atual, porém com maior densidade de pontos.

Para a segunda contribuição, voxelização e compressão foram utilizados para otimizar uma nuvem de pontos.

Na Figura 27 é apresentado o diagrama de blocos da primeira contribuição.

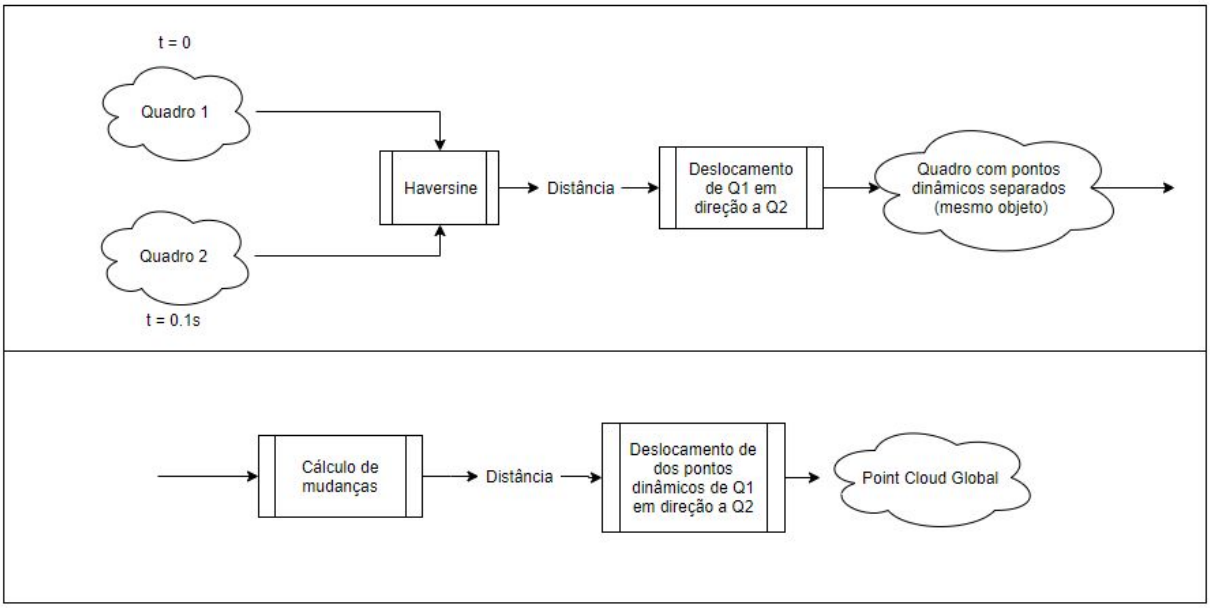

Figura 27. *Diagrama de blocos da primeira contribuição*

4. RESULTADOS E DISCUSSÕES

4.1. Introdução

O objetivo deste capítulo é apresentar os resultados práticos obtidos através das etapas descritas no Desenvolvimento (Capítulo 3). Os resultados são apresentados para as duas contribuições. Além disso realizar a análise e comentários dentro do escopo e objetivo deste trabalho.

4.2. Cenários de teste

Os cenários de teste a seguir foram utilizados para apresentar os resultados da contribuição 1. Posteriormente as nuvens de pontos geradas serão voxelizadas e comprimidas para mostrar os resultados da contribuição 2.

A inclusão do cabeçalho *.ply* como descrito na seção 3.3 permite a visualização de nuvem de pontos em programas de renderização 3D, assim como a manipulação destes dados para as demais etapas, como descrito nas seções 3.4, 3.5, 3.6, 3.7 e 3,8. Foram utilizados três diferentes cenários para realização dos testes, com

diferentes características, possibilitando uma análise mais detalhada dos resultados obtidos. As Figuras a seguir mostram quadros representantes dos três cenários de testes. Em todas as figuras desta Seção, foram destacadas as partes mais relevantes, por exemplo carros, ciclistas, etc. Na Figura 28 é possível observar três carros próximos ao centro da nuvem de pontos onde fica localizado o LiDAR que realiza as capturas e um carro isolado, mais distante do sensor e ao fundo é possível ver alguns objetos estáticos que compõem o cenário. Neste cenário os veículos se encontram parados, sendo assim objetos estáticos. Na Figura 29 é possível observar um cenário com dois ciclistas em movimento, representando objetos dinâmicos, além de objetos estáticos que compõem o cenário. Na Figura 30 é possível ver vários objetos estáticos e dois carros, que neste caso estão em movimento.

Figura 28. *Quadro do primeiro cenário de teste.*

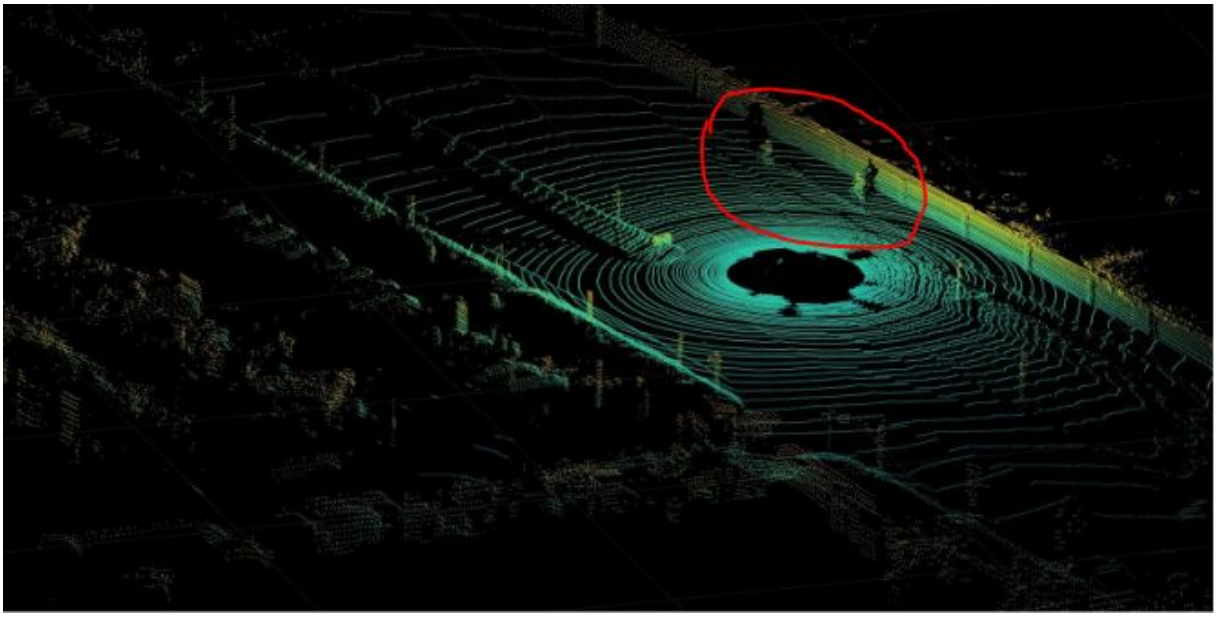

Figura 29. *Quadro do segundo cenário de teste.*

Figura 30. *Quadro do terceiro cenário de teste.*

Foram escolhidos estes cenários de testes com diferentes características para analisar os resultados obtidos sob diferentes perspectivas, sendo possível observar a validade do método em diferentes situações.

4.3. Sobreposição de nuvem de pontos

Na Figura 31 observa-se o primeiro cenário em que duas nuvens de pontos espaçadas temporalmente foram simplesmente sobrepostas sem realizar nenhum ajuste para compensar o deslocamento de uma em relação a outra (devido ao deslocamento do carro entre os quadros), fazendo com que os objetos, mesmo que estáticos apresentem-se duplicados. Uma nuvem de pontos global desta maneira tem uma grande imprecisão, pois não se sabe a verdadeira posição de um objeto. Estas nuvens de pontos representam dois quadros com *timestamps 2011-09-26 13:02:25.848114084* e *2011-09-26 13:02:25.951199337*, ou seja estão espaçados em aproximadamente 0,1 segundos.

A Figura 32 apresenta a sobreposição das nuvens de pontos a partir do deslocamento, obtido utilizando a fórmula de Haversine, deslocando-se a nuvem de pontos mais antiga em direção à atual. Para estes dois quadros apresentados, os valores de latitude e longitude do automóvel que contém o LiDAR são: latitude: 49,015006875378 longitude: 8,4343048069014 para o quadro anterior e latitude: 49,014997147797 e longitude: 8,4342801643975 para o mais recente. Assim, utilizando a fórmula de Haversine, obteve-se uma distância de 2,1 metros. As nuvens de pontos separadas possuíam 120831 e 120123 pontos, respectivamente da mais antiga para a mais recente. A sobreposição destas formou uma nuvem contendo 211702 pontos, pois este processo, devido à características da biblioteca utilizada, elimina pontos que estão na mesma posição.

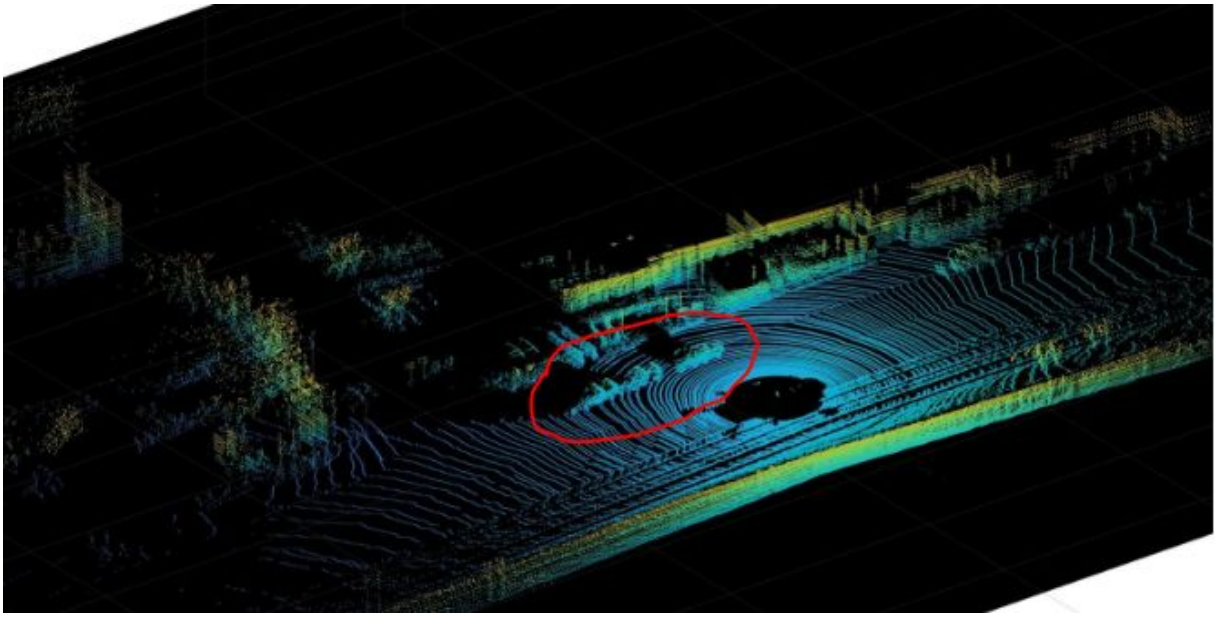

Figura 31. *Nuvem de pontos do primeiro cenário sobrepostas sem deslocamento*

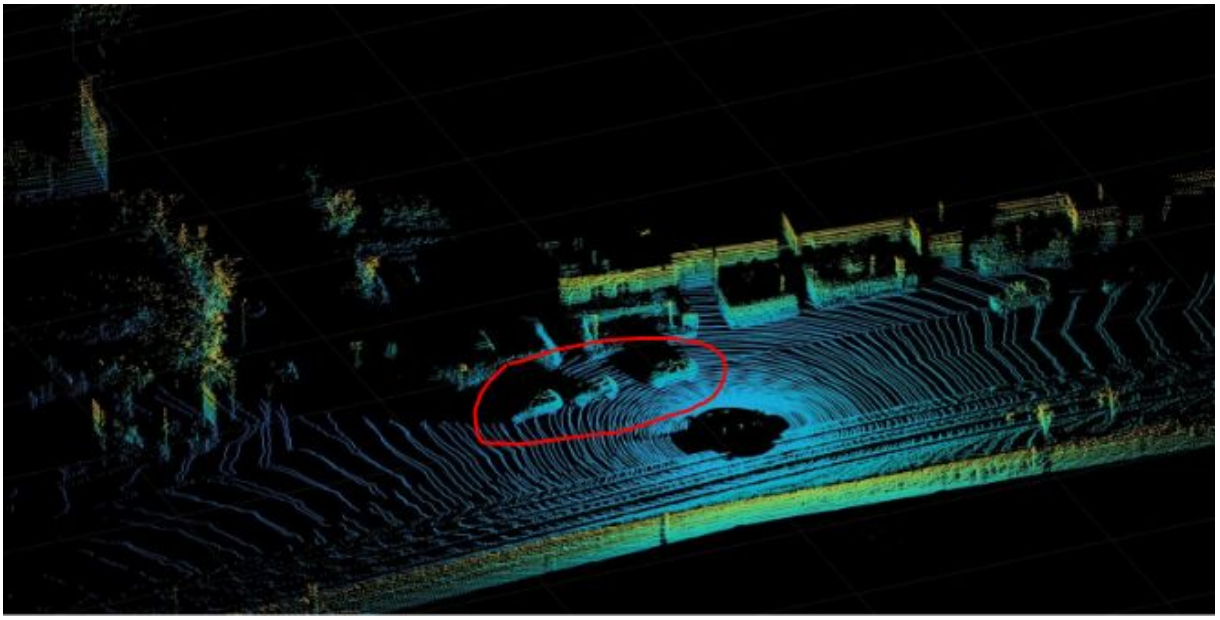

Figura 32. *Nuvem de pontos do primeiro cenário sobrepostas utilizando deslocamento obtido pela fórmula de Haversine.*

A Figura 33 mostra o segundo cenário com dois quadros sobrepostos sem deslocamento. Observa-se o efeito esperado, ou seja, objetos estáticos aparecem duplicados e os objetos dinâmicos, neste caso os dois ciclistas, não aparecem em duplicidade devido à mera coincidência de terem se movido na mesma direção e

provavelmente com velocidade próxima à do veículo que carregava o sensor LiDAR. Os *timestamps* dos quadros são: *2011-09-26 13:02:44.111111667* e *2011-09-26 13:02:44.214245818.* Neste cenário, há uma diferença entre as capturas do LiDAR e a informação de GPS e por isso, foi utilizado como deslocamento 75% do valor obtido pelo deslocamento a partir da fórmula de Haversine. Os valor utilizados são: latitude: 49.014471793038 e longitude: 8.4324199388097 para o quadro mais antigo, latitude: 49.014469886969 e longitude: 8.4324012934782 para o quadro mais atual, resultando em uma distância de 1,4 metros. Como esses valores estão dessincronizado com as capturas do sensor, foi utilizado o valor de deslocamento de 1,0322, correspondente a 75% do valor calculado por Haversine. Estas nuvens possuíam 120974 e 121156 pontos, enquanto a versão sobreposta apresenta 214513. A Figura 34 mostra esses dois quadro sobrepostos com quadro anterior sendo deslocado pelo valor obtido a partir da fórmula de Haversine. Neste caso, como previsto, foi possível colocar os objetos estáticos na mesma posição, porém os objetos dinâmicos aparecem duplicados.

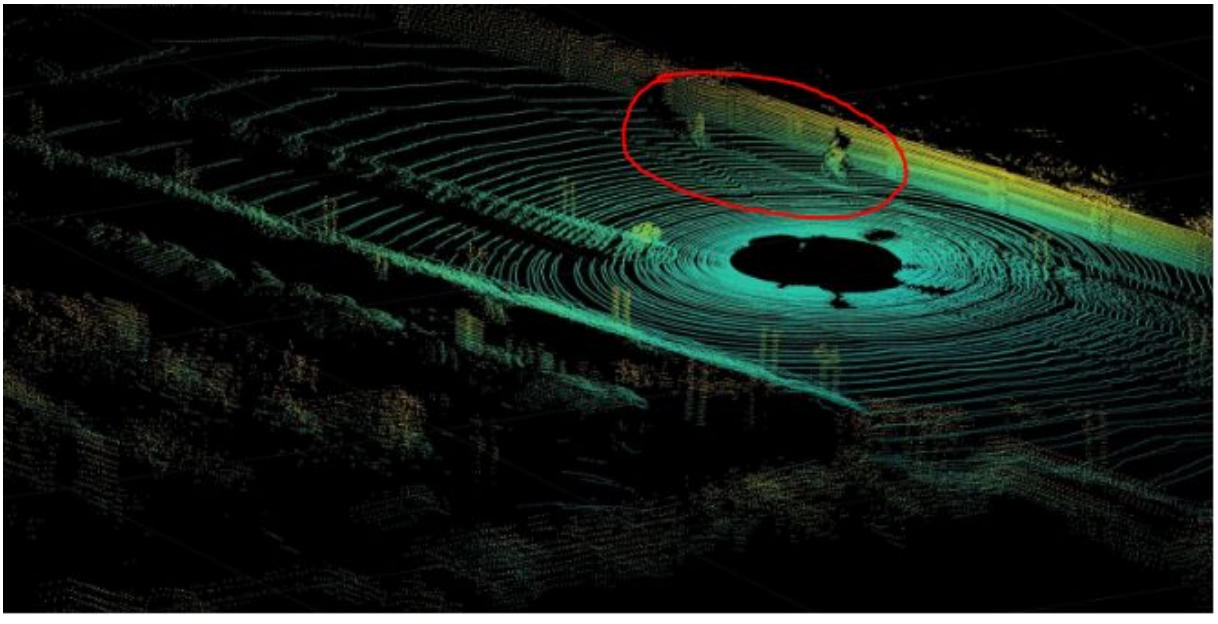

Figura 33. *Dois quadros do segundo cenário sobrepostos sem deslocamento*

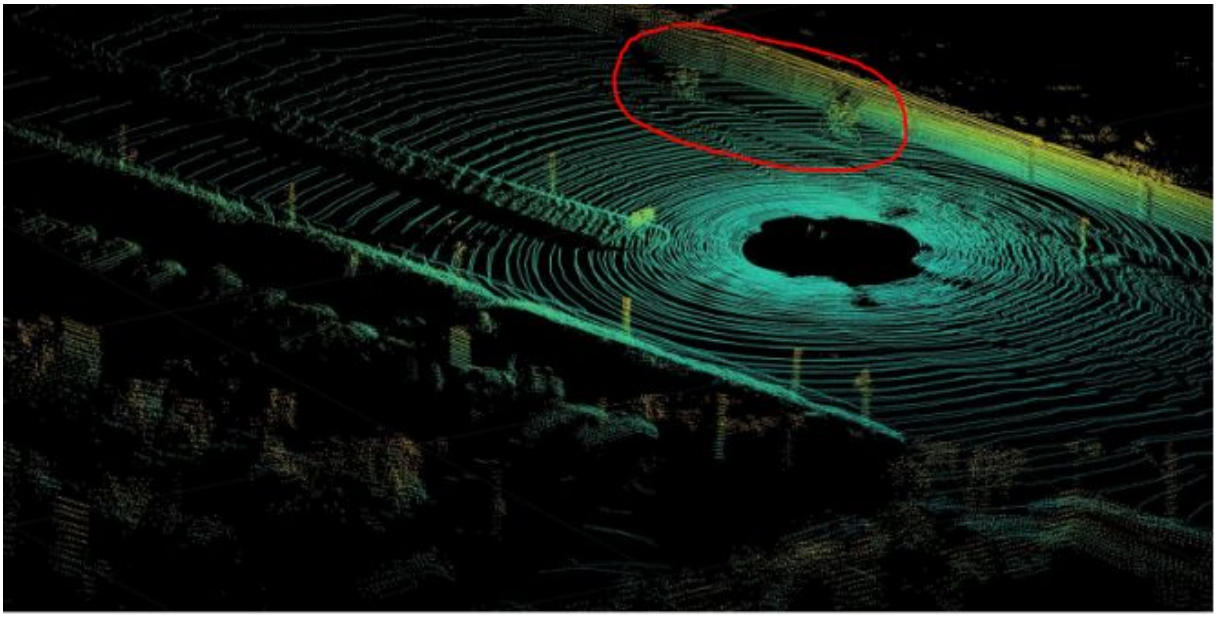

Figura 34. *Dois quadros do segundo cenário sobrepostos a partir do deslocamento obtido da fórmula de Haversine*

A Figura 35 demonstra o terceiro cenário com dois quadros sobrepostos sem realizar nenhum tipo de deslocamento. Neste caso, como previsto, é possível observar os objetos estáticos duplicados e os dois carros, que neste estão em movimento, aparecem uma única vez, devido à mera coincidência de terem se movido na mesma direção e provavelmente com velocidade próxima à do veículo que carregava o sensor LiDAR. Os *timestamps* desses quadros são: *2011-09-26 13:10:50.847425475 e 2011-09-26 13:10:50.950997178.* O valores utilizados para latitude e longitude foram: 49,013109211617 de latitude e 8,4378279928417 longitude para o quadro mais antigo, 49,01312043844 de latitude e 8,4378231085489 longitude para o quadro posterior. Pela fórmula de Haversine, o valor obtido foi de 1,2981 metros. A Figura 36 mostra os dois quadro sobrepostos com o quadro antigo sendo deslocado em relação ao primeiro. Essas nuvens apresentavam 116194 e 116026 pontos, enquanto a versão sobrepostas apresenta 205354. Novamente, como previsto, observa-se o mesmo efeito do segundo cenário, após o deslocamento, os objetos estáticos não aparecem duplicados porém o mesmo não ocorre com objetos dinâmicos.

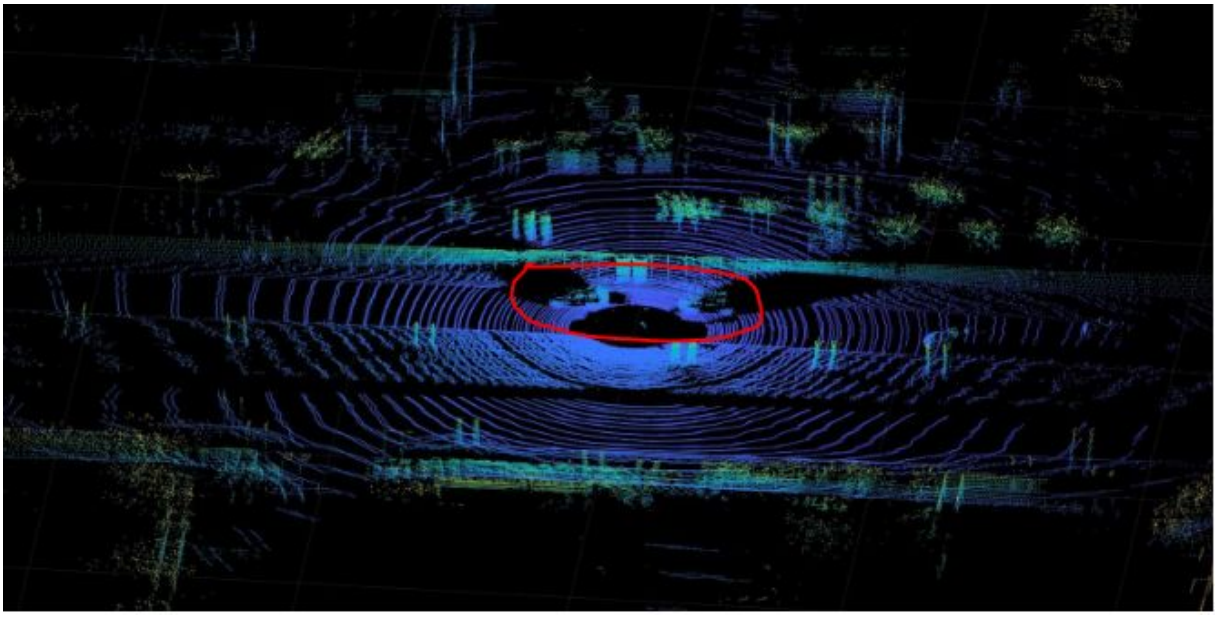

Figura 35. *Dois quadros do terceiro cenário sobrepostos sem deslocamento.*

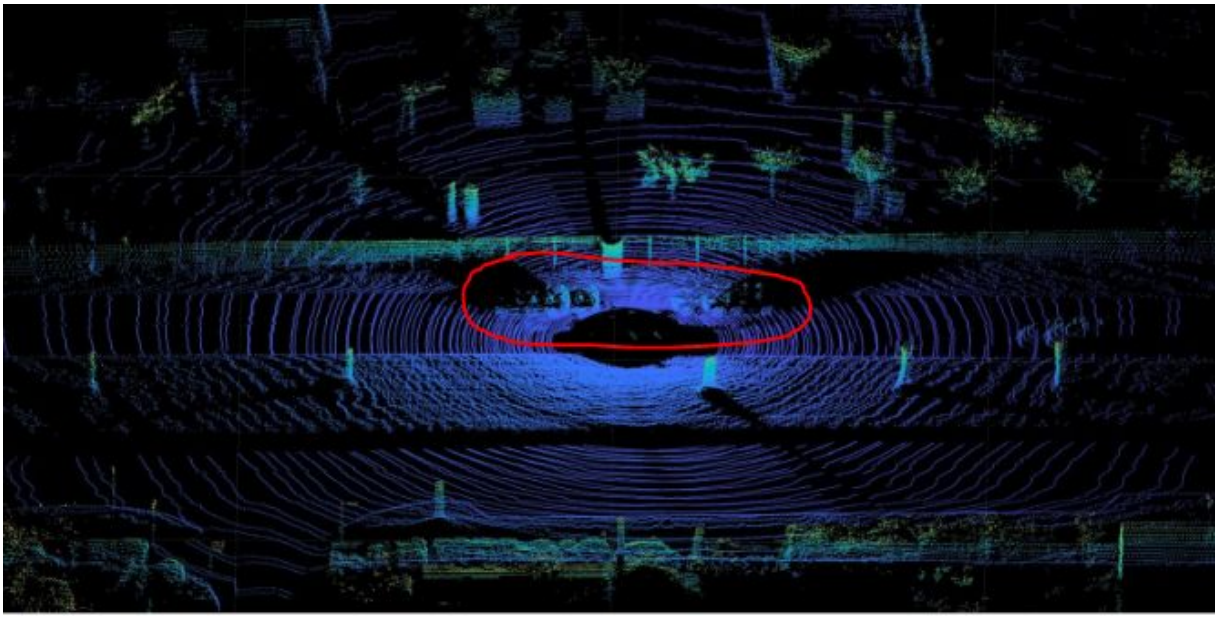

Figura 36. *Dois quadros do terceiro cenário sobrepostos utilizando o deslocamento obtido por Haversine.*

4.4. Cálculos de mudanças

Como demonstrado nas Figuras 34 e 36, a sobreposição de nuvem de pontos com objetos dinâmicos acaba causando a duplicação desse tipo de objeto, o que é indesejado. Por isso foi aplicado sobre estas nuvens de pontos o algoritmo proposto

na seção 3.6. Esse algoritmo tem como principal problema a otimização de limiares, pois não foi definido nenhuma forma de calculá-los ou prevê-los, portanto os limiares foram definidos empiricamente, isto é, através de vários testes realizados para cada caso. Essa imprevisibilidade dos limiares ocorre pois as características de tamanho (nos eixos x, y e z) de objetos variam a depender do tipo de imagem observada, por exemplo: uma nuvem de ponto de uma manada de elefantes apresentaria limiares totalmente diferentes daqueles de uma nuvem de ponto de uma colônia formigas. Uma abordagem mais adequada seria definir os limiares via algoritmos de aprendizado de máquina.

As Figuras 37 e 38 demonstram os resultados obtidos ao aplicar o algoritmo que calcula o movimento de objetos dinâmicos para evitar a duplicidade. Na Figura 37 observa-se que foi possível retirar a duplicidade de apenas um dos ciclistas, devido à imprecisões na escolha manual dos limiares. Na Figura 38 foi possível retirar a duplicidade dos dois carros, mostrando a validade do algoritmo. Para o primeiro cenário não foi necessária a aplicação do algoritmo devido à ausência de objetos dinâmicos.

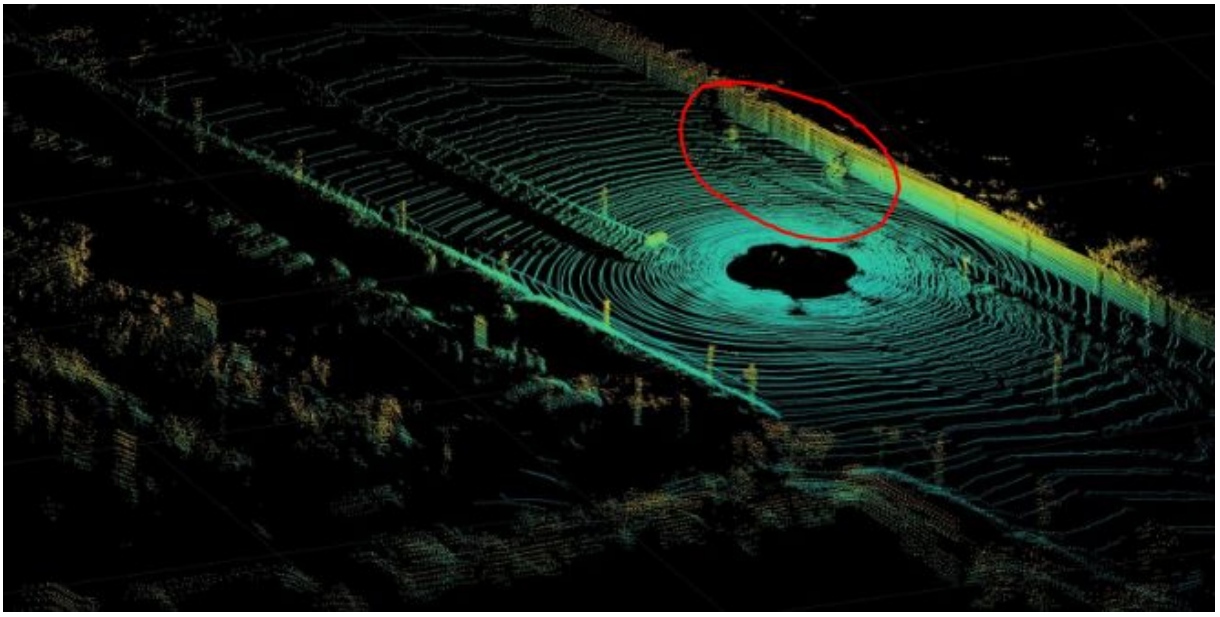

Figura 37. *Dois quadros do segundo cenário sobrepostos a partir do deslocamento obtido da fórmula de Haversine com cálculo de mudanças no cenário*

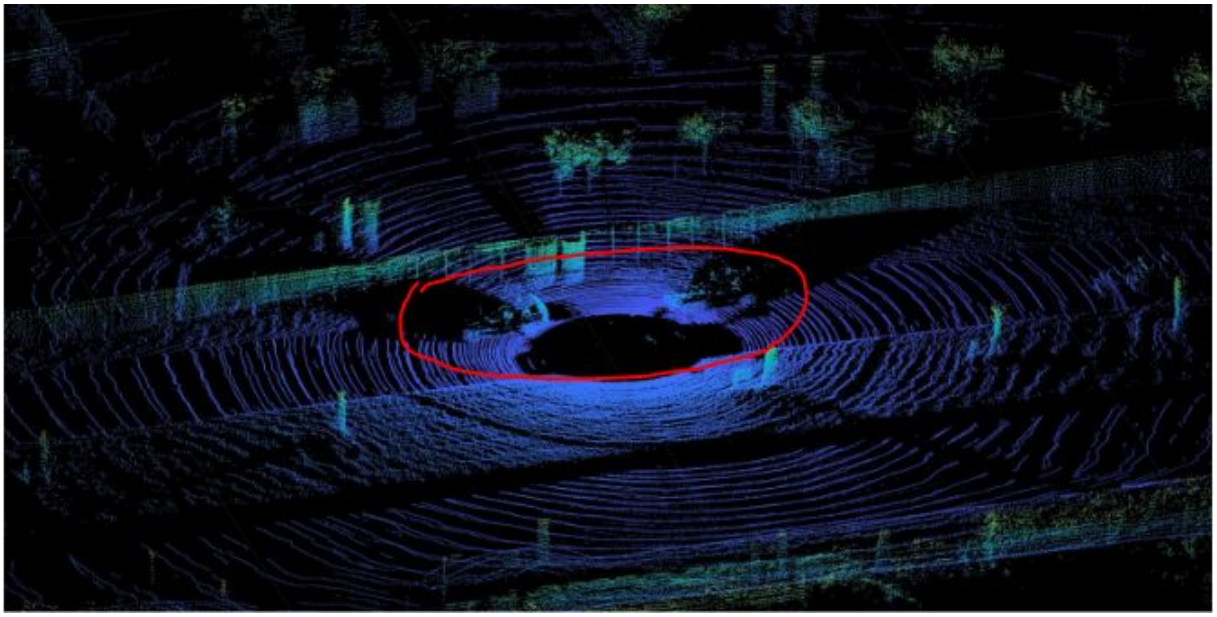

Figura 38. *Dois quadros do terceiro cenário sobrepostos a partir do deslocamento obtido da fórmula de Haversine com cálculo de mudanças no cenário*

4.5. Nuvem de pontos Global

Os passos citados anteriormente podem ser utilizados recursivamente para montar a nuvem de pontos global de uma determinada região, como apresentado no diagrama da Figura 27. A limitação do modelo adotado se dá pelo algoritmo descrito na seção 3.6, para calcular as mudanças relativas a objetivos dinâmicos, pois cada novo quadro tem um número de pontos inferior a nuvem de pontos global, o que inviabiliza a aplicação do algoritmo de maneira direta, sendo necessário a comparação do novo quadro com todos os quadros anteriores um a um que compõem a nuvem de pontos global para posterior sobreposição de todas estas, formando a versão global. A Figura 39 mostra a nuvem de pontos global formado por três quadros, contendo assim uma maior densidade de informações a respeito desse cenário, e contém 321078 pontos. Os quadros que formam esta imagem contém 120570, 120831 e 120123 pontos. Como não há objetos dinâmicos, não é necessário realizar o cálculo de mudanças. Uma característica que explicita a relevância e validade do método encontra-se no carro circulado em azul na Figura 39 que está bastante nítido, que na Figura 28 apresentavam poucos pontos, possuindo assim uma resolução pior e uma maior dificuldade de visualização, ou seja, essa aumenta a eficácia de algoritmos de classificação e detecção de imagens.

A nuvem de pontos do segundo cenário, representado pela Figura 40, contém três quadros espaçados em 0,1 segundos, correspondendo a 0,3 segundos de informação e contém 319468 pontos. As nuvens de pontos que o formam tem 120974,121156 e 121080 pontos. Um dos ciclistas que compõe o cenário apareceu de forma triplicado e isto ocorreu pelo fato de que os limiares da Seção 3.6 não serem otimizados. A Figura 41 demonstra o processo de criação da nuvem de pontos global formado por três quadros que possuíam 116194,116026 e 115713 pontos, representando o terceiro cenário, formando uma única imagem que contém 337234. As imprecisões da definição dos limiares criou uma nuvem de pontos que apesar de ter o funcionamento adequado para os objetos dinâmicos, criou imprecisões em objetos estáticos, como sinalizado pela marcação em amarelo.

Figura 39. *Nuvem de pontos global do primeiro cenário formado por três quadros.*

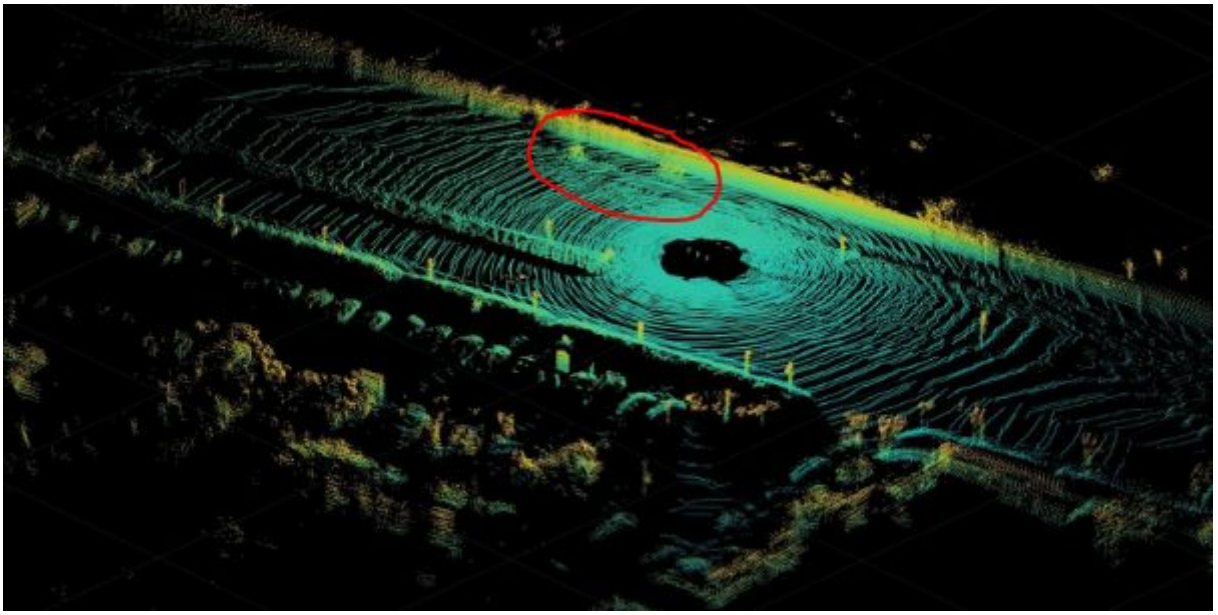

Figura 40. *Nuvem de ponto global formado por três quadros do segundo cenário*.

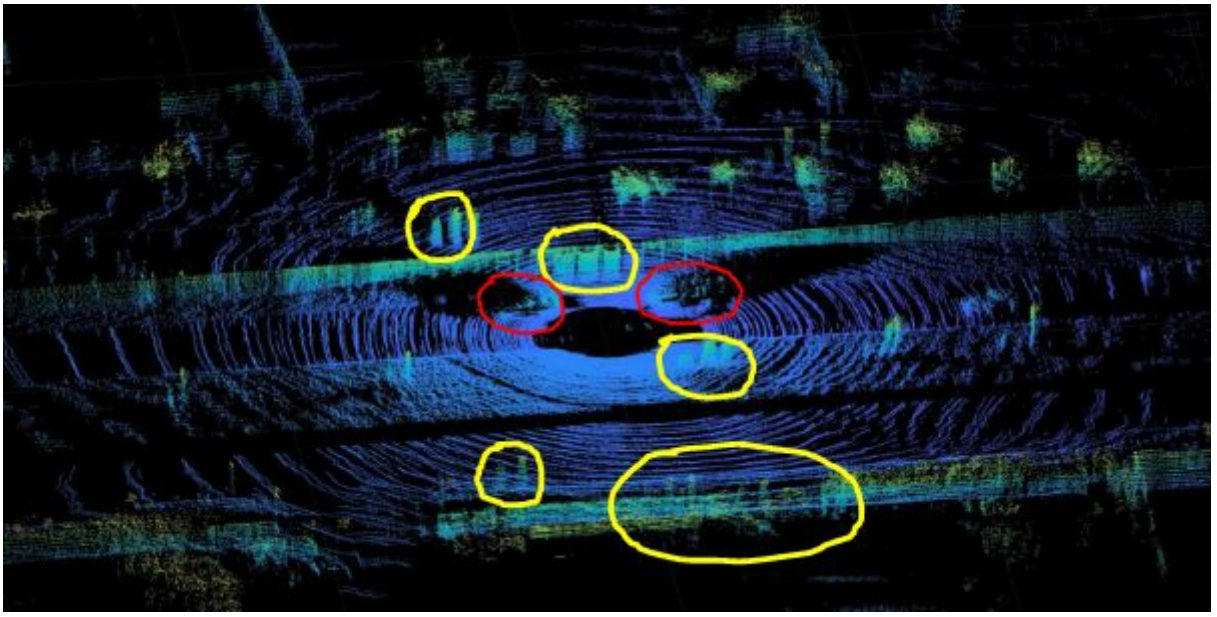

Figura 41. *Nuvem de pontos global formado por três quadros do terceiro cenário*

4.6. Voxelização e Compressão

O processo de voxelização têm sua importância devido a facilidade de representação de uma nuvem de pontos a partir de voxel e vantagens relativas a

processamento deste tipo de informação. Para a escolha do número de níveis que seriam utilizados, foram realizados testes baseado no número de pontos que permaneceram após a divisão em níveis. A tabela 1 contém o número de voxels preenchidos para cada divisão em grade regular de dimensões WxWxW com W = 2^N, sendo N o número de níveis. Escolheu-se a utilização de 11 níveis para esta nuvens de pontos pois manteve-se uma boa quantidade de número de voxels preenchidos pela quantidade de níveis utilizados. A tabela 2 contém o número de bits por voxel necessário a taxa de bits por voxel para codificação octree e o número de bits necessário para a transmissão destas nuvem de pontos para uma voxelização utilizando 11 níveis. A Figura 42 demonstra a nuvem de pontos do primeiro cenário voxelizada com 11 níveis. A divisão feita na codificação octree foi feita com 11 níveis.

	Sem Voxelização	**9 Níveis**	**10 Níveis**	**11 Níveis**
Primeiro Cenário	321078 pontos	35203 voxels preenchidos	85585 voxels preenchidos	172356 voxels preenchidos
Segundo Cenário	319468 pontos	41768 voxels preenchidos	95257 voxels preenchidos	182706 voxels preenchidos
Terceiro Cenário	337234 pontos	48211 voxels preenchidos	108572 voxels preenchidos	197731 voxels preenchidos

Tabela 1. Dados obtidos através da voxelização para os três cenários

	Taxa de bits por voxel para codificação octree	Bits para transmissão da nuvem de ponto
Primeiro Cenário	6,47	1,11 Megabits

Segundo Cenário	7,022	1,28 Megabits
Terceiro Cenário	7,38	1,46 Megabits

Tabela 2. Dados obtidos através da codificação octree para os três cenários

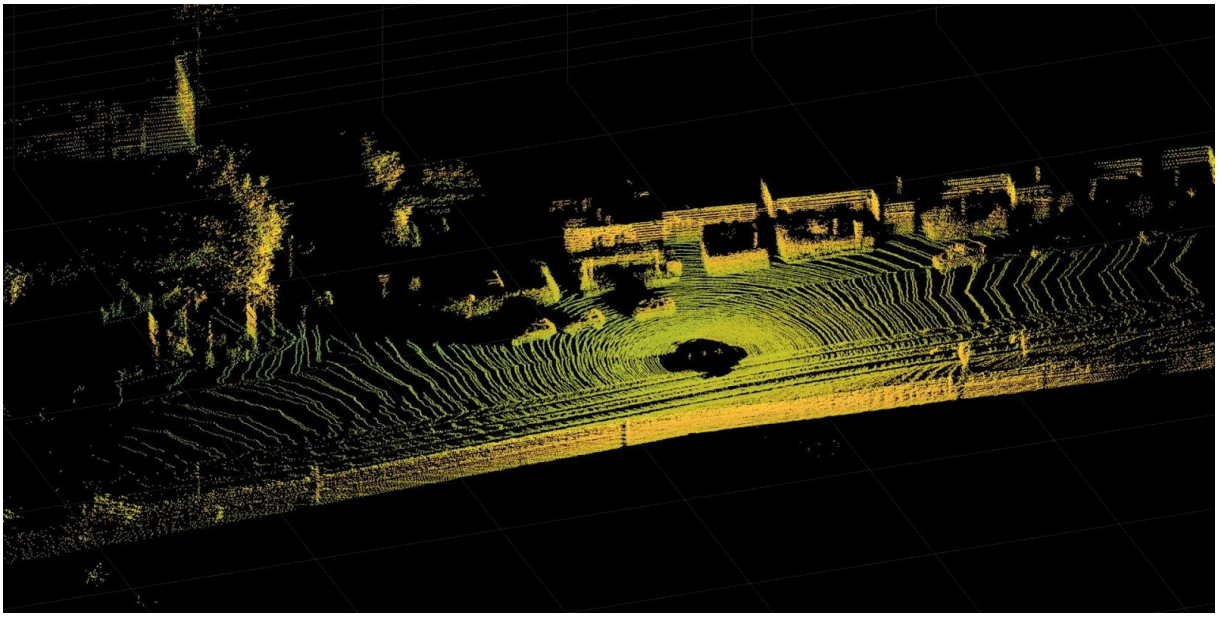

Figura 42. Nuvem de pontos global do primeiro cenário voxelizada em 11 níveis

4.7. Limitações

O algoritmo de cálculo de mudanças descrito na Seção 3.6 só funciona para objetos que se deslocam horizontalmente, já que ele organiza os pontos dos quadros em ordem crescente das coordenadas do eixo x e não considera variações significativas em torno de y e z. Isso implica que qualquer cenário fora deste padrão produzirá resultados indesejados, por exemplo alguns de seus pontos podem não ser caracterizados como pertencentes ao mesmo objeto e por isso nem todos os pontos serão deslocados para a posição correta, resultando numa figura cortada e parcialmente duplicada.

O algoritmo de cálculo de mudanças foi desenvolvido para comparar nuvens com número de pontos semelhantes, sendo assim, cada novo quadro é necessário comparar um a um com quadros anteriores, diminuindo a performance do algoritmo.

Como explicado na seção 4.4, o algoritmo exige a definição de limiares manualmente, por esses não serem previsíveis. Esse fato gerou inconsistências em alguns resultados (figuras 40 e 41), essa inconsistências, no entanto, não prejudicaram a validação do método. Como descrito na seção 4.4, uma melhor abordagem seria implementar um algoritmo utilizando aprendizado de máquina para definir os limiares baseando-se na análise das imagens.

4.8. Conclusão

Os resultados obtidos das duas contribuições foram dentro do esperado, respeitando as limitações do desenvolvimento e escopo do trabalho de conclusão de curso. Foram discutidos durante o desenvolvimento deste capítulos melhorias necessárias para trabalhos futuros.

As inconsistências apresentadas nos resultados (figuras 40 e 41), não prejudicaram a validação do método proposto. Cada nuvem de pontos obtidas diretamente do sensor LiDAR apresentam cerca de 3 MB e com a aplicação da voxelização e da codificação octree, foi possível enviar uma nuvem de pontos com 3 quadros ocupando apenas cerca de 1,5 Mb, apresentando-se como um bom ganho.

5. CONCLUSÃO

O desenvolvimento da tecnologia de automação veicular é de bastante interesse tanto da indústria quanto do meio acadêmico. Neste contexto, estudos sobre mapas de alta definição, que fornecem informações sobre o ambiente para veículos inteligentes, tem vital importância para funcionamento desta aplicação. O trabalho de conclusão desenvolvido apresenta duas contribuições no contexto de nuvens de pontos. Na primeira contribuição, sugeriu-se um método para a transmissão de nuvens de pontos, fornecendo uma nuvem global de um cenário, que possui como principal vantagem um mapa com maior densidade de pontos que representam os objeto que o compõem. Na segunda contribuição, sugeriu-se a utilização de um método da codificação do tipo octree para compressão diminui a

quantidade de dados a serem enviados, permitindo que estas informações possam ser enviadas através das infraestruturas de rede já existentes.

Do ponto de vista educacional, este trabalho além de essencial para a conclusão do curso de Engenharia de Redes de Comunicação, mostrou-se de grande relevância para o crescimento intelectual e pessoal dos integrantes do projeto, fornecendo experiência com um tema atual e aprofundando conceitos aprendidos durante a graduação, em especial o processamento de sinais e de imagens. O desenvolvimento do algoritmo de cálculo de mudanças foi uma solução própria e que exigiu esforços de estudo e pesquisa por parte dos integrantes deste projeto, processos fundamentais para um projeto final de graduação.

Em suma a análise dos integrantes é de que o projeto, em termos de aplicação prática, mostrou-se relevante para tratar o tema proposto, apesar das limitações descritas na Seção 4.7. Já em termos de conhecimento acadêmico o projeto foi um grande sucesso, devido à riqueza dos temas estudados e o esforço empenhado pelos integrantes, que absorveram grande quantidade de informação extremamente relevante nos campos de automação veicular, processamento de imagens e nuvem de pontos

REFERÊNCIAS

1. AARON. **Vehicle Automation**. Disponível em: <http://www.evmeme.com/vehicle-automation/>. Acesso em: 27 nov. 2019.
2. **Four fundamentals of workplace automation**. Disponível em: <http://www.mckinsey.com/business-functions/mckinsey-digital/our-insights/four-fundamentals-of-workplace-automation>. Acesso em: 27 nov. 2019. Tradução nossa

3. ACEMOGLU, D.; RESTREPO, P. Artificial Intelligence, Automation and Work. **The National Bureal of Economic Research**, 2018. Tradução nossa
4. BOTSCH M., WIRATANAYA A., KOBBELT L, Efficient high quality rendering of point sampled geometry. In EGRW ’02: Proceedings of the 13th Eurographics workshop on Rendering, Aire-la-Ville, Switzerland, 2002, Eurographics Association, pp. 53.
5. COHEN-OR, D.; KAUFMAN, A. Fundamentals of Surface Voxelization. **Graphical Models and Image Processing**, v. 57, n. 6, p. 453–461, 1995.
6. DONG, Z. et al. Real-time voxelization for complex polygonal models. **12th Pacific Conference on Computer Graphics and Applications, 2004. PG 2004. Proceedings.**, Nov. 2004.
7. **Driving Directions, Traffic Reports & Carpool Rideshares by Waze**. Disponível em: <http://www.waze.com/>. Acesso em: 27 nov. 2019.
8. DUMIC, E.; DUARTE, C. R.; CRUZ, L. A. D. S. Subjective evaluation and objective measures for nuvem de pontos — State of the art. **2018 First International Colloquium on Smart Grid Metrology (SmaGriMet)**, 2018.
9. ESQUEF, I. A.; DE ALBUQUERQUE, M. P.; DE ALBUQUERQUE, M. P. Processamento Digital de Imagens. 18 Feb. 2003.
10. EWALT, D. M. **Garry Kasparov vs IBM Deep Blue**. Disponível em: <http://www.forbes.com/sites/davidewalt/2011/05/03/kasparov-vs-deep-blue/#2e84cc1730f8>. Acesso em: 27 nov. 2019.
11. FRAEDRICH, E.; BEIKER, S.; LENZ, B. Transition pathways to fully automated driving and its implications for the sociotechnical system of automobility. **European Journal of Futures Research**, v. 3, n. 1, 2015.

12. FUTURECAR. **BMW China and Beijing-based Mapping Company NavInfo to Develop HD Maps for Autonomous Driving**. Disponível em: <https://www.futurecar.com/3346/BMW-China-and-Beijing-based-Mapping-Company-NavInfo-to-Develop-HD-Maps-for-Autonomous-Driving>. Acesso em: 27 nov. 2019.
13. GARCIA, D. C. et al. Geometry Coding for Dynamic Voxelized nuvem de pontos Using Octrees and Multiple Contexts. **IEEE Transactions on Image Processing**, v. 29, p. 313–322, 2019.
14. GEIGER, A. et al. Vision meets robotics: The KITTI dataset. **The International Journal of Robotics Research**, v. 32, n. 11, p. 1231–1237, 2013.
15. GOLOVINSKIY, A.; FUNK, T. Min-cut based segmentation of nuvem de pontos. **2009 IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops**, 2009.
16. GONZALEZ, R. C.; WOODS, R. E. **Digital image processing**. Traducao . New York, NY: Pearson, 2018.
17. GROOVER, M. P. **Automation**. Disponível em: <http://www.britannica.com/technology/automation>. Acesso em: 27 nov. 2019.
18. IVEZIC , Zeljko; CONNOLLY, Andrew J.; VANDERPLAS, Jacob T. **Statistics, Data Mining, and Machine Learning in Astronomy: A Practical Python Guide for the Analysis of Survey Data. 1.** ed. Oxford, OX, United Kingdom: Princeton University Press, 2014. ISBN 13: 9780691151687.
19. JANSSEN, C. P. et al. History and future of human-automation interaction. **International Journal of Human-Computer Studies**, v. 131, p. 99–107, 2019.

20. JENSEN, J. R. **Remote sensing of the environment an earth resource perspective**. Traducao . Harlow: Pearson, 2014.
21. JUDD, D. B. A Maxwell Triangle Yielding Uniform Chromaticity Scales*. **Journal of the Optical Society of America**, v. 25, n. 1, p. 24, 1935.
22. LIN, P. Why Ethics Matters for Autonomous Cars. **Autonomous Driving**, p. 69–85, 2016.
23. LIU, Jiang-Long; CHANG, Henry Ker-Chang. **A linear quadtree compression scheme for image encryption.** Signal Processing: Image Communication 10, [*S. l.*], p. 1-2, 9 jan. 1995.
24. **Makers of MATLAB and Simulink**. Disponível em: <http://www.mathworks.com/>. Acesso em: 27 nov. 2019.
25. MANGESH, N. K.; CHOPDE, N. R. Landmark based shortest path detection by using A* Algorithm and Haversine Formula. **International Journal of Innovative Research in Computer and Communication Engineering**, 28 Apr. 2013.
26. MAXWELL, J. C. 1. Experiments on Colour as perceived by the Eye, with Remarks on Colour-Blindness. **Proceedings of the Royal Society of Edinburgh**, v. 3, p. 299–301, 1857.
27. NAGY, B.; BENEDEK, C. Real-Time nuvem de ponto Alignment for Vehicle Localization in a High Resolution 3D Map. **Lecture Notes in Computer Science Computer Vision – ECCV 2018 Workshops**, p. 226–239, 2019.
28. **NAVYA**. Disponível em: <https://navya.tech/en/about-navya/>. Acesso em: 27 nov. 2019.
29. NGUYEN, A.; LE, B. 3D nuvem de ponto segmentation: A survey. **2013 6th IEEE Conference on Robotics, Automation and Mechatronics (RAM)**, 2013.

30. PICAND, Y.; DUTOIT, D. **Lidar**. Disponível em: <http://dictionary.sensagent.com/LIDAR/de-de/>. Acesso em: 27 nov. 2019.
31. **POLISCAN SPEED FIXED**. Disponível em: <http://www.vitronic.com/traffic-technology/applications/traffic-enforcement/speed-enforcement/poliscan-speed-fixed.html>. Acesso em: 27 nov. 2019.
32. SCHNABEL, Ruwen; KLEIN, Reinhard. **Octree-based Point-Cloud Compression**. Eurographics Symposium on Point-Based Graphics, Institut für Informatik II, Universität Bonn, Germany, jun. 2006.
33. SCHOWENGERDT, R. A. **Remote sensing: models and methods for image processing**. Traducao . Amsterdam: Elsevier, 2011.
34. SCHWARZ, B. Mapping the world in 3D. **Nature Photonics**, v. 4, n. 7, p. 429–430, 2010.
35. SCHWARZ, M.; SEIDEL, H.-P. Fast parallel surface and solid voxelization on GPUs. **ACM SIGGRAPH Asia 2010 papers on - SIGGRAPH ASIA '10**, 2010.
36. SHIH, F. R. A. N. K. Y. **IMAGE PROCESSING AND MATHEMATICAL MORPHOLOGY: fundamentals and applications**. Traducao . Place of publication not identified: CRC Press, 2017.
37. SMAGRIMET 2018 colloquium program. **2018 First International Colloquium on Smart Grid Metrology (SmaGriMet)**, 2018.
38. SOLOMON, C.; BRECKON, T. **Fundamentals of digital image processing: a practical approach with examples in Matlab**. Traducao . Chichester: Wiley-Blackwell, 2012.
39. TORLIG, E. D. E. M. MÉTRICAS OBJETIVAS BASEADAS EM PROJEÇÕES PARA AVALIAÇÃO DE QUALIDADE DE NUVENS DE PONTOS. Nov. 2018.

40. TULLEKEN, HERMAN **Automation**. Disponível em: <http://devmag.org.za/2011/02/23/quadtrees-implementation/>. Acesso em: 02 dec. 2019.
41. WINNER, H.; WACHENFELD, W. Effects of Autonomous Driving on the Vehicle Concept. **Autonomous Driving**, p. 255–275, 2016.
42. YANG, B.; LUO, W.; URTASUN, R. PIXOR: Real-time 3D Object Detection from nuvem de pontos. **2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition**, 2018.
43. YOUNG, T. The Bakerian Lecture. On the Theory of Light and Colours. **Proceedings of the Royal Society of London**, v. 1, p. 63–67, 1800.
44. ZHANGA, Jinghua; OWEN, Charles B. **Octree-based animated geometry compression.** Computers & Graphics 31, Michigan State University, East Lansing, MI 48824, USA, p. 1-11, 8 dez. 2005.
45. ZHOU, Y.; TUZEL, O. VoxelNet: End-to-End Learning for nuvem de ponto Based 3D Object Detection. **2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition**, 2018.