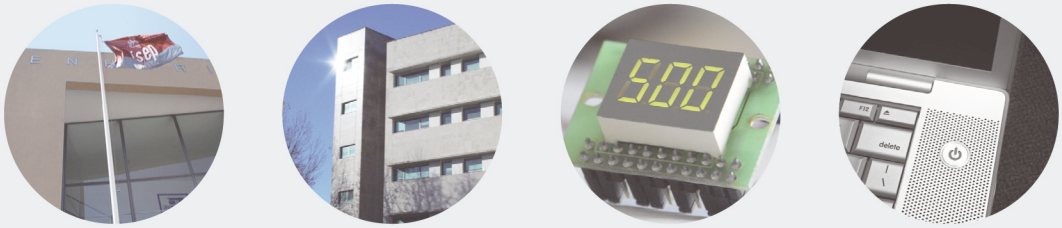
INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA ELECTROTÉCNICA E DE COMPUTADORES



Deep Learning LiDAR-based Power Lines Detection for Unmanned Aerial Vehicles

TIAGO FRANCISCO PEREIRA CUNHA
novembro de 2023

POLITÉCNICO
DO PORTO

Politécnico do Porto
Instituto Superior de Engenharia do Porto

Deep Learning LiDAR-based Power Lines Detection for Unmanned Aerial Vehicles

Tiago Francisco Pereira Cunha

Mestrado em Engenharia Electrotécnica e de Computadores
Área de Especialização em Sistemas Autónomos

Departamento de Engenharia Eletrotécnica
Instituto Superior de Engenharia do Porto

Novembro, 2023

Esta dissertação satisfaz, parcialmente, os requisitos que constam da Ficha de Unidade Curricular de Tese/Dissertação, do 2º ano, do Mestrado em Engenharia Electrotécnica e de Computadores, Área de Especialização em Sistemas Autónomos.

Candidato: Tiago Francisco Pereira Cunha, Nº 1180922,
1180922@isep.ipp.pt

Orientação Científica: André Miguel Pinheiro Dias, apd@isep.ipp.pt

Coorientação Científica: Tiago André Miranda Dos Santos, mir@isep.ipp.pt

Departamento de Engenharia Eletrotécnica
Instituto Superior de Engenharia do Porto
Rua Dr. António Bernardino de Almeida, 431, 4200–072 Porto

Novembro, 2023

Resumo

O crescente grau de dependência das sociedades modernas na utilização da eletricidade e a relutância das populações em aceitar a instalação de novas infraestruturas elétricas fazem com que as infraestruturas existentes operem constantemente no limite de sua capacidade. Esta realidade implica a necessidade premente de prevenir qualquer ocorrência de falhas, uma vez que estas teriam repercussões económicas significativas para as empresas de eletricidade e resultariam em perturbações no fornecimento de energia aos consumidores. Este cenário tem enfatizado cada vez mais a importância crescente da eficácia na supervisão e manutenção das redes elétricas.

Neste sentido, a utilização de *Unmanned Aerial Vehicles* (UAV) representa uma das alternativas mais atraentes para a tarefa de inspeção de linhas de transmissão. A inspeção de linhas elétricas através de UAVs elimina a necessidade de expor indivíduos a situações de risco. Isto permite que os trabalhadores permaneçam em segurança no solo, enquanto um operador de UAV avalia a condição das linhas de transmissão. No entanto, isso requer o desenvolvimento de algoritmos para garantir que o processo de inspeção seja fiável e autónomo.

A identificação das linhas de transmissão e dos seus componentes associados é tipicamente conduzida através de técnicas de deteção visual. Estas técnicas, de forma geral, mostram-se sensíveis às condições atmosféricas e a fundos com ruído visual, especialmente em situações de iluminação inadequada ou na presença de fundos com alto contraste. Para superar essas limitações, o presente estudo aborda a problemática da identificação e modelação das linhas elétricas com base na utilização de um sensor *Light Detection And Ranging* (LiDAR).

Com este propósito, foi desenvolvido um algoritmo com a capacidade de identificar a localização e orientação das linhas de transmissão, através da utilização dos dados recolhidos por um sensor LiDAR. Em especifico, o algoritmo desenvolvido faz uso de um *deep learning LiDAR-based 3D object detection model* para não só prever a posição, mas também estimar a direção da trajetória das linhas, mediante a localização e orientação das *3D oriented bounding boxes* que o algoritmo prevê.

Adicionalmente, é integrado no sistema um pré-processamento e um pós-processamento, com o intuito de aumentar a sua eficácia. O pré-processamento tem como função reduzir o volume de dados de entrada e prepará-los de forma adequada para garantir que o modelo possa aprender e generalizar eficazmente. E o pós-processamento atua para aprimorar as previsões do modelo para alcançar resultados mais precisos.

O sistema foi testado em três configurações com um conjunto de dados reais, e a partir dos resultados obtidos, foi possível validar a eficácia do sistema proposto, tanto em termos dos resultados gerados como de tempo de processamento.

Palavras-Chave: *deep learning*, LiDAR, *object detection*, UAV, linhas de transmissão, linhas elétricas

Abstract

The increasing dependence of modern societies on electricity and the reluctance of populations to accept the installation of new electrical infrastructure lead existing infrastructures to constantly operate at the edge of their capacity. This reality necessitates the urgent need to prevent any occurrence of failures, as these would have significant economic repercussions for electricity companies and result in disruptions in energy supply to consumers.

This scenario has increasingly emphasized the growing importance of effectiveness in the supervision and maintenance of electrical grids.

In this regard, the use of Unmanned Aerial Vehicles (UAVs) represents one of the most attractive alternatives for the task of inspecting transmission lines. The inspection of electrical lines using UAVs eliminates the need to expose individuals to risky situations. This allows workers to remain safely on the ground while a UAV operator assesses the condition of the transmission lines. However, this requires the development of algorithms to ensure that the inspection process is reliable and autonomous.

The identification of transmission lines and their associated components is typically conducted through visual detection techniques. These techniques, in general, prove to be sensitive to atmospheric conditions and backgrounds with visual noise, especially in situations of inadequate lighting or in the presence of high-contrast backgrounds. To overcome these limitations, this study addresses the issue of identification and modeling of electrical lines based on the use of a Light Detection and Ranging (LiDAR) sensor.

For this purpose, an algorithm has been developed with the ability to identify the location and orientation of transmission lines based on the data collected by a LiDAR sensor. Specifically, the developed algorithm makes use of a deep learning LiDAR-based 3D object detection model to not only predict the position but also estimate the direction of the trajectory of the lines, based on the location and orientation of the 3D oriented bounding boxes predicted by the algorithm.

Additionally, pre-processing and post-processing are integrated into the system to increase its effectiveness. Preprocessing functions to reduce the volume of input data and prepare it appropriately to ensure that the model can learn and generalize effectively. Post-processing works to enhance the model's predictions to achieve more accurate results.

The system was tested in three configurations with a real dataset, and based on the results obtained, it was possible to validate the effectiveness of the proposed system, both in terms of the generated results and processing time.

Keywords:*deep learning*, LiDAR, *object detection*, UAV, transmission lines , power lines

Índice

Lista de Figuras

Lista de Tabelas

Listagens

Lista de Acrónimos

2D *Two-Dimensional*

3D *Three-Dimensional*

AMOTA *Average Multi-Object Tracking Accuracy*

BEV *Bird's Eye View*

CNN *Convolutional Neural Network*

DL *Deep Learning*

FCN *Fully Convolutional Network*

FPN *Feature Pyramid Network*

FPS *Farthest Point Sampling*

IA Inteligência Artificial

INESC TEC Instituto de Engenharia de Sistemas e Computadores, Tecnologia e Ciência

IoU *Intersection over Union*

ISEP Instituto Superior de Engenharia do Porto

LiDAR *Light Detection And Ranging*

lr *learning rate*

LSA Laboratório de Sistemas Autónomos

mAP *mean Average Precision*

mAR *mean Average Recall*

MLP *Multi-Layer Perceptron*

NDS *Normalized Detection Score*

PMF *Perception-aware Multi-sensor Fusion*

RGB *Red Green Blue*

RPN *Region Proposal Network*

RS *Random Sampling*

SAC *Spatially-Adaptive Convolution*

UAV *Unmanned Aerial Vehicles*

VFE *voxel feature encoding*

Capítulo 1

Introdução

A distribuição de energia elétrica é vital para o funcionamento de uma sociedade moderna. Para garantir a distribuição regular de eletricidade é necessário adotar métodos eficazes de monitoramento e manutenção das linhas de energia. Considerando a crescente dependência da sociedade na energia elétrica, este tema tem ganho bastante importância, uma vez que assegurar a distribuição da energia de forma regular é um ponto chave na qualidade do serviço prestado pelas empresas distribuidoras de energia elétrica.

Dado o desenvolvimento progressivo de novas centrais de produção de energia verde na Europa, a necessidade de um transporte elétrico com maior capacidade e mais confiável tem crescido. No entanto, a instalação de novas linhas de transmissão geralmente não são aceitas pela população. Isso leva a uma operação constante na capacidade máxima das linhas de energia, sem redundâncias ou reservas para compensar falhas. Nesse sentido, para evitar possíveis perdas económicas e apagões para os consumidores, as empresas de energia elétrica precisam de adotar uma filosofia de manutenção preventiva e preditiva por meio, por exemplo, de uma inspeção visual e térmica periódica[1].

Redes elétricas normalmente são construídas por uma rede de transmissão nacional, redes regionais e redes de distribuição. Sendo que maior parte da rede está localizadas dentro de florestas. Então a supervisão das linhas de energia recai essencialmente em dois aspetos: estado da linha de transmissão e objetos ao redor, especialmente vegetação [2].

No que diz respeito à monitorização da condição dos componentes das linhas de alta tensão, geralmente estas cruzam milhares de quilómetros em ambientes suburbanos, montanhas e florestas e muitas vezes são expostas por longos períodos a condições ambientais severas, como variações térmicas, chuva, gelo, vento, vibração induzida, temperaturas extremas e agentes agressivos (chuvas ácidas). Estas causas, juntamente com a corrosão, induzem rupturas por fadiga que reduzem a vida útil das linhas e causam perdas significativas. Além disso, raios também podem causar danos graves, como rupturas nos fios e fusão parcial dos cabos. Portanto a condição dos componentes precisa ser verificada regularmente para detetar falhas causadas.

Em relação ao segundo aspeto de inspeção, enquanto as estruturas artificiais criadas pelo homem podem ser regulamentadas por leis de construção, a vegetação é fruto de um fenómeno natural e, particularmente em áreas rurais, o seu crescimento é descontrolado. A queda de uma única árvore sobre uma linha de energia é amplamente reconhecida como a principal causa de falhas no fornecimento de energia, podendo resultar em interrupções generalizadas[3]. As tempestades são um dos principais fatores que contribuem para o problema das interrupções de energia causadas pela vegetação, pois trazem um clima húmido que enfraquece o solo e ventos fortes que podem derrubar galhos ou até mesmo árvores inteiras sobre as linhas de energia. Em áreas mais secas as árvores que crescem próximas às linhas podem facilmente se incendiar e dar origem a grandes incêndios florestais. Por isso, é responsabilidade da empresa de fornecedora de energia fazer inspeções regulares para detectar e prevenir o crescimento excessivo de vegetação perto das linhas de energia [4].

Nos últimos anos, houve um aumento significativo de cenários de aplicação de *Unmanned Aerial Vehicles* (UAV) devido ao crescente esforço de pesquisa no campo da robótica aérea. Este conjunto mais amplo de aplicações está relacionado com uma mudança contínua no foco da pesquisa. No início deste século, a pesquisa de UAV estava principalmente relacionada com o desenvolvimento de hardware e controlo. Porém recentemente, o foto de pesquisa está associado operações de busca e salvamento, vigilância e inspeção de estruturas, entre outras [5].

Neste sentido o uso de UAV é uma das alternativas mais atrativas para a tarefa de inspeção de linhas de transmissão. Inspeções tradicionais exigem uma equipa de *linemen* experientes em andaimes e guindastes para inspecionar visualmente todas as linhas numa zona e avaliar as linhas em busca de danos ou possíveis problemas. A inspeção de linhas elétricas por meio de UAVs elimina a necessidade de expor pessoas a situações de risco, permitindo que os trabalhadores permaneçam em segurança no solo enquanto o operador do UAV avalia a linha. Adicionalmente pode ser utilizado para monitorar a invasão de vegetação ao longo das linhas de energia, especialmente em áreas de difícil acesso. Uma vez que são capazes de navegar entre árvores densas, além de decolar e pousar em lugares apertados.

Apesar das vantagens citadas, o uso de um UAV em um ambiente complexo e

restrito, como aquele em que uma linha de energia pode se inserir, pode ser desafiador tanto para o piloto quanto para o planeamento de missões autónomas.

A presença de vegetação ou outros obstáculos pode dificultar a deteção visual de *power lines* pelos pilotos, o que ressalta a importância de um módulo autónomo capaz de detetar obstáculos e evitar colisões em modos assistidos ou autónomos. Além disso, a deteção de linhas de transmissão pode ser utilizada para aprimorar o sistema de navegação e as manobras de seguimento de linha.

Este trabalho tem como objetivo melhorar a capacidade de perceção dos UAVs permitindo uma operação mais segura na tarefa de inspeção de linhas de transmissão. Para isso, propõe-se o desenvolvimento de um algoritmo capaz de detetar *power lines* em tempo real, recorrendo a dados de um sensor *Light Detection And Ranging* (LiDAR). Especificamente, o algoritmo em questão emprega técnicas de *deep learning* para identificar pontos a partir de informação obtida de um LiDAR, que correspondam a uma linha de transmissão, uma vez que estes podem ser erradamente interpretados como pontos esparsos, o que pode levar à sua não deteção pelo sistema de *collision avoidance* do UAV .

1.1 Contextualização

O Instituto de Engenharia de Sistemas e Computadores, Tecnologia e Ciência (INESC TEC) e o Laboratório de Sistemas Autónomos (LSA), do Instituto Superior de Engenharia do Porto (ISEP), têm vindo a unir esforços nos últimos anos para o desenvolvimento de sistemas autónomos. Esta parceria já resultou no desenvolvimento de diversos sistemas robóticos para múltiplos domínios: terrestre, subaquático e aéreo.

Neste sentido, este projeto surge como seguimento de dois projetos anteriormente desenvolvidos na mesma área, Cunha e Oliveira et al. (2023) e Azevedo et al. (2018) [6, 7].

Ambos os projetos propunham o mesmo objetivo de aprimorar a autonomia e a segurança na tarefa de inspeção de linhas de transmissão. No entanto, em Cunha e Oliveira et al., eram utilizadas a informação de uma câmara do espectro visual e uma câmara do espectro infravermelho para, a partir de um *deep lerning model*, detetar as linhas de transmissão.

Já Azevedo et al. desenvolveram um algoritmo chamado Power Line LiDAR-based Detection and Modeling (PL^2DM), que, por meio de uma análise minimalista de todos os pontos presentes em uma dada nuvem de pontos obtida por um sensor LiDAR, realiza a deteção de linhas elétricas.

Deste modo, esta dissertação procura utilizar a informação de um sensor LiDAR para, através de técnicas de *deep learning* aplicadas a nuvens de pontos, detetar linhas de transmissão e, assim, oferecer uma perspetiva diferente à resolução do problema central.

1.2 Objetivos

Neste estudo, o objetivo principal é identificar a localização e orientação das linhas de energia, com base numa nuvem de pontos obtida através de um sensor LiDAR. Isto possibilita a melhoria dos módulos de deteção de obstáculos e evasão de colisões do UAV. Dessa forma, as capacidades de perceção do UAV são aprimoradas, o que resulta em um aumento da autonomia e da segurança na tarefa de inspeção de linhas de transmissão.

Para tal, foram consideradas os seguintes objetivos a que deverão ser atendidos de forma criteriosa ao longo do projeto:

- Estudo dedicado aos métodos de *deep learning* desenvolvidos para *point clouds*;
- Desenvolvimento de um processamento preliminar da *raw point cloud* para garantir que o modelo possa aprender e generalizar com eficácia;
- Treino de um *deep learning model* capaz de detetar as linhas de transmissão de forma eficaz;
- Formulação de um sistema robusto, apto a operar em tempo real e a identificar a localização e orientação das linhas de energia elétrica;
- Validação do sistema desenvolvido em diferentes ambientes.

1.3 Estrutura do Relatório

Esta dissertação está dividida em seis capítulos principais. No próximo capítulo, é apresentado um estudo preliminar dedicado aos métodos de *deep learning* desenvolvidos para *point clouds*, relevantes no contexto do desenvolvimento do projeto. No encerramento deste capítulo, será realizada uma análise concisa das vantagens e desvantagens predominantes das abordagens apresentadas, seguida de uma avaliação dos métodos que se destacam.

No Capítulo 3, é apresentado um *overview* da arquitetura de software definida para a implementação do sistema proposto. Neste capítulo, são abordados todos os objetivos e motivações de cada uma das etapas que compõem a arquitetura apresentada. No seguinte capítulo, é realizada uma descrição detalhada da implementação de todas as etapas do sistema.

No Capítulo 5, é analisada a performance das etapas de processamento e os resultados obtidos pelos sistemas implementados com um conjunto de dados reais.

Finalmente, no último capítulo, apresentam-se as considerações finais e realiza-se uma análise geral do trabalho desenvolvido, onde também são formuladas sugestões para trabalhos futuros que possam aprimorar o sistema desenvolvido.

Capítulo 2

Estado da Arte

Ao longo deste capítulo, é apresentado um estudo dedicado aos métodos de *deep learning* desenvolvidos para *point clouds*, que têm relevância no contexto do desenvolvimento do projeto.

No encerramento deste capítulo, realiza-se uma análise concisa das vantagens e desvantagens predominantes das abordagens apresentadas, seguida de uma avaliação dos métodos que se destacaram.

2.1 *Deep Learning for 3D Point Clouds*

A perceção é um dos tópicos fundamentais no ramo da visão computacional. Nos últimos anos, *Deep Learning* (DL) tem ganho bastante relevância nesta área devido a uma sequência notável de sucessos empíricos [8]. Este sucesso é em grande parte atribuído à arquitetura das *deep learning neural networks*, que contrariamente às *neural networks* tradicionais possui a capacidade de aprender de forma progressiva a extrair *higher-level features* dos dados de entrada durante o processo de treino sendo esta considerado a principal razão para os resultados excecionais [9].

Os algoritmos de perceção com base *deep learning* podem ser divididos em *Three-Dimensional* (3D) e *Two-Dimensional* (2D) dependendo se o algoritmo leva em consideração a posição e orientação do objeto no espaço 3D real ou apenas a posição do objeto no plano de imagem. Apesar da perceção 2D com recurso a *deep learning* ter avançado significativamente e ter se consolidado como uma área madura

na visão computacional, a perceção 3D ainda é um tópico emergente e pouco explorado. Relativamente à perceção 2D, a compreensão 3D do espaço é uma tarefa mais complexa, além de que a sua representação é mais complicada, visto que requer requisitos computacionais acrescidos devido à adição de uma dimensão extra .

A eficácia de *Deep learning* tem contribuído significativamente para o avanço de diversos sistemas autónomos que cada vez mais apresentam a necessidade de compreensão do espaço tridimensional. Por conseguinte, o estudo de implementação de *deep learning neural networks* em *point clouds* tem recebido um grande impulso, com a proposta de inúmeros métodos para resolver diversos problemas de processamento de *point clouds*. Nomeadamente *3D shape classification*, *3D object detection and tracking*, *3D point cloud segmentation*, *3D point cloud registration*, *6-DOF pose estimation* e *3D reconstruction* [10, 11, 12].

Levando em consideração o objetivo proposto nesta dissertação, das tarefas de perceção previamente citadas, aquelas que possivelmente se adequam à implementação do sistema são *3D object detection* e *3D point cloud segmentation*.

Analisando um *3D object detector* típico, percebe-se que este recebe como *input* uma *point clound* e gera como *output* uma *3D bounding box* que vai delimitar o objeto alvo. Em contrapartida, o método de *3D point cloud segmentation* através do mesmo *input* segmenta a *point clound* em vários subconjuntos com base nos significados atribuídos pela rede de *deep learning*.

Logo esta escolha é justificável, uma vez que o principal objetivo do projeto consiste em identificar, a partir de uma *point clound*, os pontos que representam uma *power line*, e ambas as abordagens, ou seja, *3D object detection* e *3D point cloud segmentation* são capazes de alcançar esse objetivo.

Neste sentido, nos próximos subcapítulos serão apresentados alguns dos métodos existentes para a *3D object detection* e *3D point cloud segmentation*.

2.1.1 *3D Object Detection*

Como mencionado, um *3D object detector*, a partir de uma *point clound*, gera como saída uma *3D bounding box* que localiza no espaço tridimensional o objeto pretendido detetar, como é evidenciado na Figura 2.1.

Os métodos de *3D object detection* podem ser categorizados em *Projection-Based Methods*,*Voxel-Grid Based Methods*,*Point-based Methods*, *Hybrid methods* e *Multimodal Fusion-Based Methods*.

Projection-Based Methods

Os *Projection-Based Methods* convertem a *point clound* em projeções bidimensionais, tais como *spherical views*, *cylindrical views* e *top-views* (*bird's eye views*). Estas imagens 2D são então processadas por uma *deep learning neural networks* convencional

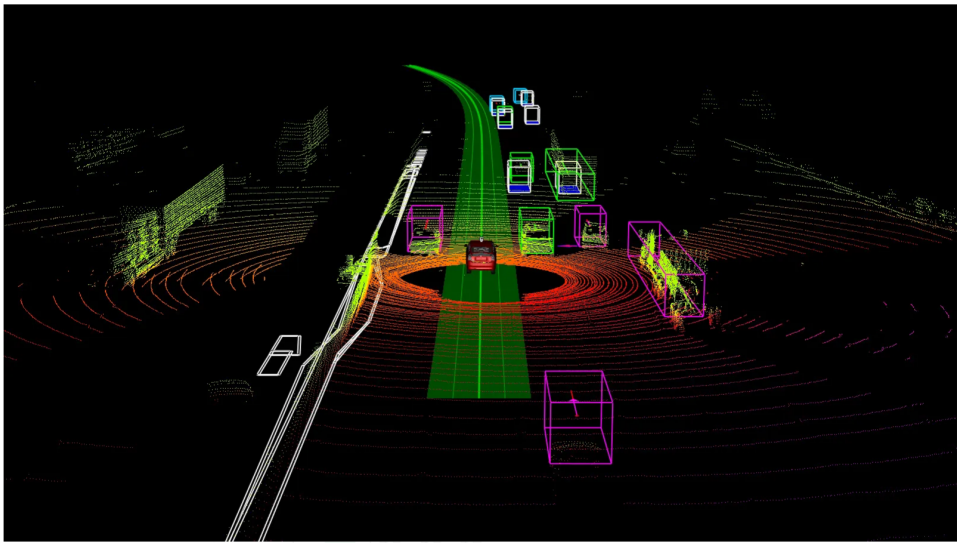


Figura 2.1: Ilustração do *3D object detection* [13]

para obter 2D bounding boxes. Seguidamente, a partir da posição e tamanho das *bounding boxes* detetadas nas diversas projeções, é estimada as *3D bounding boxes* correspondentes.

Em [14], Li et al. introduz a implementação de uma *Fully Convolutional Network* (FCN) para a detetar veículos numa *point cloud* de um LiDAR. Especificamente, a *point cloud* é projetada numa *cylindrical image* para obter um *depth map* bidimensional que é utilizado como *input* da FCN. Naturalmente, como output a rede retorna as *3D bounding boxes* classificadas.

Num conceito similar, Minemura et al. em [15] também utiliza a projeção cilíndrica de uma *point cloud* para implementar um sistema deteção em *real-time*, o LMNet. No entanto, diferentemente do método anterior, a partir da *cylindrical projection*, a *point cloud* é projetada em cinco representações frontais: *Reflection*, *Range*, *Forward*, *Side* e *Height*, que posteriormente são empregadas como entrada de uma *dilated convolutional network*. Deste modo. a divisão em cinco *feature channels* permitiu então aprimorar a precisão da localização e orientação das *bounding boxes*.

Em contrapartida dos métodos citados anteriormente, que codificam a *point cloud* em representações frontais, a representação *Bird's Eye View* (BEV) evita problemas de oclusão, uma vez que nesta perspetiva os objetos ocupam sempre áreas diferentes no mapa 2D. Adicionalmente, a BEV preserva o comprimento e largura dos objetos e ainda fornece a posição dos objeto diretamente em relação ao plano do solo, tornando a tarefa de localização mais simples.

Uma possível abordagem fundamentada nesta perspetiva, consiste em converter a *point cloud* numa representação BEV em células de três canais.

Seguindo esta abordagem, Yu et al. [16] utiliza a representação BEV como

elevation images, na qual cada pixel da imagem é codificado com os valores de altura máxima, mediana e mínima correspondentes, como ilustrado na Figura 2.2.

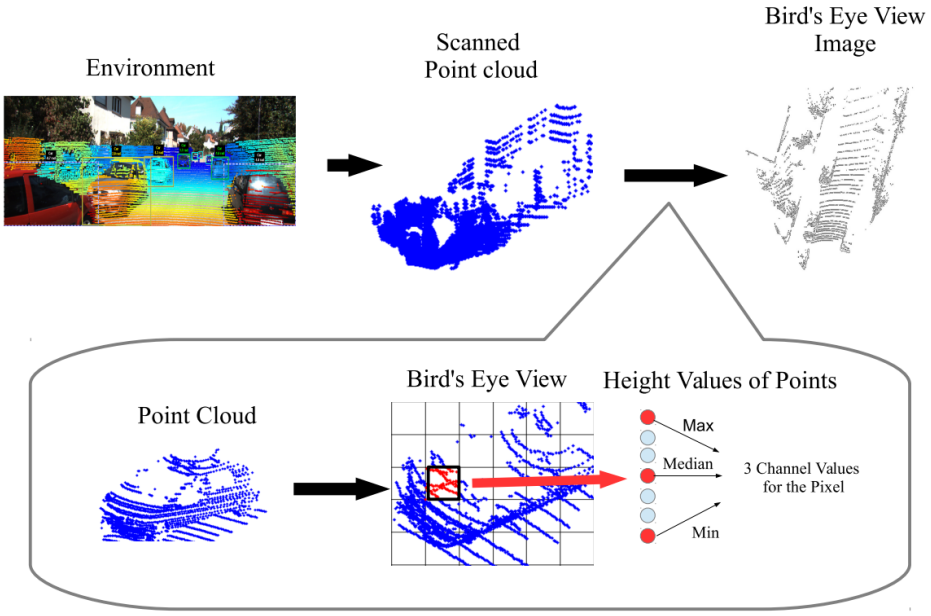


Figura 2.2: *BEV image construction* [16]

De forma semelhante, o BirdNet [17] e o BirdNet+ [18] utilizam informações da altura, intensidade e densidade para codificar os três canais da representação BEV.

Uma grande vantagem desta abordagem reside no fato de que, assim como uma imagem *Red Green Blue* (RGB) comum, esta representação é constituída por três canais. Portanto, esta pode ser utilizada em redes convencionais de *object detection* indicadas para imagens, sem a necessidade de modificações adicionais.

Os três métodos mencionados utilizam como base a arquitetura *Faster R-CNN*. Esta arquitetura é classificada como um *two-stage detector*, onde, na primeira etapa, são gerados *region proposals* para na segunda etapa, a partir de uma rede neuronal, prever e classificar as 3D *bounding boxes*.

Além da utilização de *two-stage detectors*, outras técnicas também foram exploradas na detecção de objetos em 3D, como o YOLO3D [19] e CG-SSD [20], que alcançaram excelentes resultados ao empregar *single-stage detectors* na representação *BEV*.

Voxel-Grid Based Methods

Voxel-grid based methods, discretizam uma *3D point cloud* numa representação volumétrica 3D (*voxel-grid*) na qual cada elemento (*voxel*) desta representação é classificado como ocupado ou vazio. Esse processo preserva a silhueta da nuvem de pontos e assim esta pode ser diretamente aplicada numa *3D convolutional network*. No entanto, devido à natureza esparsa e irregular de uma *3D point cloud*, muitos *voxels* ficam vazios, o que reduz consideravelmente a eficiência com o processamento das células vazias. Além disso, dado que as 3D *convolutional networks* usadas para processar estes dados são computacionalmente intensivas, a resolução da representação volumétrica é limitada pela capacidade computacional.

Em contraste com os métodos previamente expostos, que dependem de representações criadas manualmente, como *bird's eye view projection*, o VoxelNet [21] elimina esta necessidade e aceita diretamente como entrada uma *point cloud*.

A arquitetura deste método essencialmente está dividida em três blocos: (1) *Feature learning network*, (2) *Convolutional middle layerse* (3) *Region proposal network*, como representado na Figura 2.3.

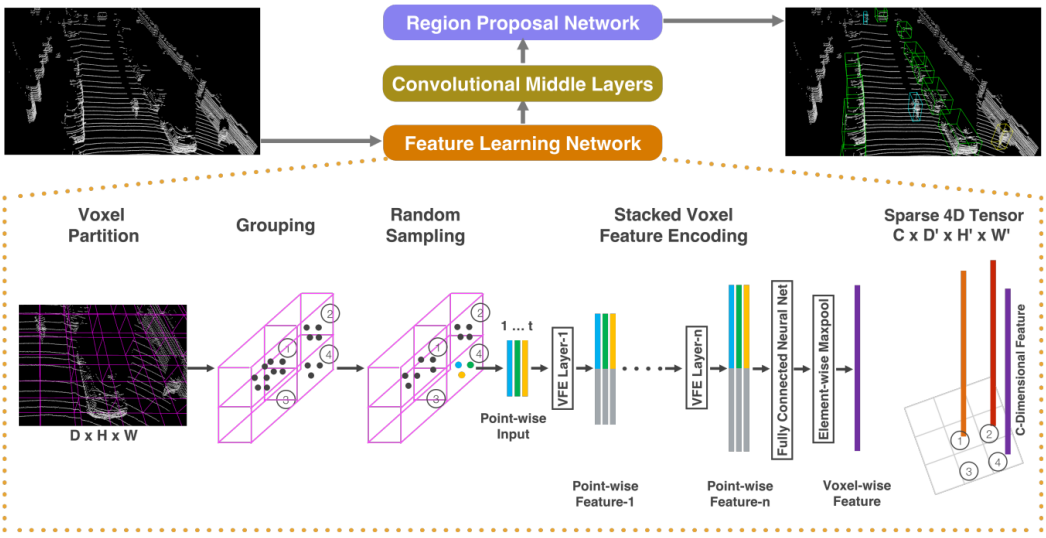


Figura 2.3: VoxelNet *architecture*[21].

Na primeira etapa, a *Feature learning network*, decompõe uma *point clound* em *voxels* de tamanho uniforme e, em seguida, o grupo de pontos dentro de cada voxel é codificado através de vários *voxel feature encoding* (VFE) *layers* em um único *set* de *features*. O VFE *layer* é uma das principais contribuições desta técnica, pelo que possibilita a extração de *point-wise features* e *local features*. Então, encadeando vários *VFE layers* é então possível estabelecer uma representação volumétrica dos dados de alto nível.

O bloco seguinte, aplica uma série de convoluções às *voxel-wise features* anteriormente extraídas . Isto permite aumentar a dimensionalidade dos dados e assim adicionar mais contexto ao *shape descriptor*.

Finalmente, o *feature map* fruto dos *convolutional middle layers* é utilizado como *input* de uma *Region Proposal Network* (RPN).

Apesar da alta eficácia do VoxelNet, o principal *bottleneck* desta técnica é o seu elevado custo computacional. Como mencionado, isto deve se o fato de que as 3D *convolutional networks* são computacionalmente intensas, onde a exigência computacional desta aumenta de forma cúbica com a resolução da representação volumétrica em *voxels*.

Em [22], a introdução do SECOND implementou diversas melhorias para o VoxelNet, com o objetivo de aumentar a velocidade de treino e de inferência. Especificamente, SECOND, após os VFE layes, emprega *sparse convolutional layers* para converter os dados de voxel em imagens 2D e ainda utiliza uma nova técnica de *angle loss regression* para aprimorar a estimativa da orientação. Contudo, esta abordagem apresenta o mesmo *bottleneck* de outros *voxel-grid based methods*, que é a dependência do uso de 3D *convolutional networks*.

Point-based Methods

Tipicamente *Convolutional Neural Network* (CNN) requerem que os dados de entrada estejam num formato regular, então como já abordado, devido à natureza irregular de uma point cloud, grande parte das técnicas transformam uma *point cloud* em projeções bidimensionais (*projection-based methods*) ou numa *voxel-grid* (*voxel-grid based methods*) .

Contudo, no processo de pré-processamento, é inevitável que ocorram perdas de informações espaciais. Ademais, *voxel-grid based methods* exigem 3D CNNs, que são altamente intensivas em termos de recursos computacionais, o que restringe a sua aplicação em cenários práticos. Então, diante destas desvantagens, foi desenvolvidos métodos alternativos para lidar diretamente com os dados de uma *point clouds*.

PointNet [23] é uma arquitetura de deep learning que opera diretamente numa *unordered raw point clouds* sem a necessidade de realizar operações que resultam na perda de informação, como os métodos já mencionados. Esta é considerada uma arquitetura unificada que aprende *global e local point features*, proporcionando uma abordagem simples, eficiente e eficaz para uma série de tarefas de perceção 3D, nomeadamente *3D object detection* e *semantic segmentation*.

Essencialmente, a rede recebe n pontos de *input*, no qual são aplicadas *input transformations* e *point-wise feature transformations* a estes, para então a partir de um *max-pooling layer* agregar todas as *features* provenientes. Desta forma, são extraídas *features* locais e globais para que, em seguida, estas possam ser concatenadas

para servirem como *input* de rede indicada para diferentes tarefas de perceção, nomeadamente: *3D object detection*, *3D object part segmentation* e *pointwise semantic segmentation*.

Embora a PointNet seja capaz de extrair *global e local point features*, não consegue capturar *local features* em diferentes escalas. Considerando que a habilidade de caracterizar estruturas locais é crucial para o desempenho de uma CNN, isto é dado como um grande *bottleneck* da PointNet.

Então, com a introdução da PointNet++ [24], foi proposta uma solução que ultrapassa as limitações da PointNet, pelo que esta é capaz de aprender *local structures features*. A PointNet++ constrói uma hierarquia de partição de pontos, onde ao longo da hierarquia, esta isola regiões locais cada vez menores, como está ilustrado na Figura 2.4.

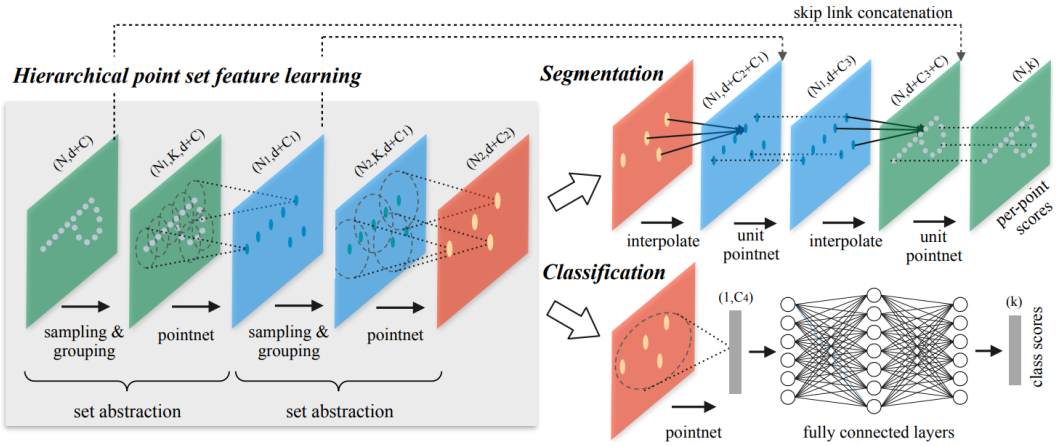


Figura 2.4: Arquitetura hierarquia da PointNet++ [24].

Esta estrutura é composta por *n set abstraction levels*. Sendo que, em cada nível um dado conjunto de pontos é processado num conjunto com menos elementos, segundo o algoritmo *Farthest Point Sampling* (FPS). Em seguida, cada partição é submetida a uma PointNet, que, por sua vez, extrairá *local features*. Desta forma, é possível extrair *local features* em diferentes escalas.

Hybrid Methods

Hybrid Methods, combinam diferentes técnicas ou abordagens para melhorar a precisão e robustez do processo de deteção de objetos 3D. Estes métodos, por norma, tiram proveito de informações 2D e 3D para ultrapassar as limitações das suas abordagens individuais. Neste sentido, o CenterPoint [25] é um método de destaque que combina a projeção da representação volumétrica 3D (*voxel-grid*) da *point cloud* para uma análise 2D e a extração de *point-features* para uma análise 3D.

Numa fase inicial, este utiliza um *keypoint detector* para identificar o centro dos objetos e, em seguida, calcula outros atributos, como tamanho 3D, orientação 3D e velocidade. Seguidamente, na segunda etapa, essas estimativas são aprimoradas com base nas *point features* extraídas do objeto. A Figura 2.5 ilustra a estrutura global do modelo CenterPoint.

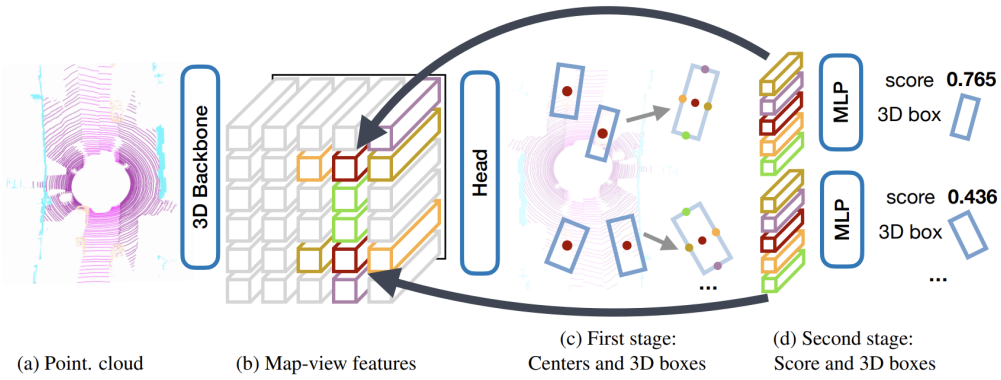


Figura 2.5: *Overview* da estrutura global do CenterPoint[25].

Especificamente, o CenterPoint emprega uma *Lidar-based backbone network*, como o VoxelNet [21], que foi previamente abordada no Capítulo 2.1.1, ou o PointPillars [25], com o intuito de criar uma representação da *point cloud* a ser utilizada como entrada na rede, e deste modo atuar essencialmente como um *encoder*.

Posteriormente, essa representação é convertida numa visualização aérea 2D (*overhead map-view*), para através de um *image-based keypoint detector* padrão encontrar os centros dos objetos [26]. Nesta sequência, para cada centro detetado, é efetuada uma regressão para obter todas as outras propriedades do objeto, tais como tamanho 3D, orientação e velocidade, a partir da *point-feature* da localização central.

Na segunda etapa, são extraídas *point-features* dos centros tridimensionais de cada face da *3D bounding box* estimada. Assim, é possível recuperar informação geométrica local perdida na conversão para *overhead map-view*, proporcionando um aumento de desempenho notável com custos mínimos.

Esta metodologia, baseada numa representação centrada no centro de um objeto, apresenta diversas vantagens fundamentais: Ao contrário das *bounding boxes*, pontos não possuem uma orientação intrínseca. Isto reduz significativamente o espaço de pesquisa do detetor de objetos, ao mesmo tempo que possibilita que a *backbone* aprenda a invariância rotacional dos objetos e a equivalência rotacional da sua rotação relativa.

Adicionalmente, a extração de *point-features* permite desenvolver um processo de duas etapas altamente eficiente, significativamente mais rápido do que as abordagens anteriores.

Multimodal Fusion-Based Methods

As técnicas de *object detection* baseadas em imagens extraem unicamente informações de textura, sem considerar as informações de profundidade. Em contrapartida, métodos de processamento que utilizam *point clouds* como base. extraem informações de profundidade, mas são carentes em informações de textura.

A informação de textura desempenha um papel crucial na deteção e classificação de objetos, enquanto a informação de profundidade é fundamental na estimativa precisa da localização 3D e do tamanho dos objetos. Além disso, à medida que a distância entre o sensor e o objeto a ser detetado aumenta, a densidade da *point clouds* diminui proporcionalmente. Então, como ambas as informações são cruciais para a detecção de objetos em 3D, algumas técnicas combinam imagens e *point clouds* com o intuito de aprimorar o desempenho global do sistema.

Substancialmente, existem três esquemas de fusão: *early fusion*, *late fusion* e *deep fusion*, conforme ilustrado na Figura 2.6. A técnica de *early fusion* combina dados brutos ou pré-processados, enquanto no esquema de *late fusion* a fusão é dada pela concatenação do ultimo *layer of feature maps*. Por outro lado, *deep fusion* mistura features hierarquicamente [10].

Em geral, as abordagem mais recorrentes combinam *features* extraídas de 2D CNNs de ambas fontes (*point cloud* e imagens RGB). Para tal abordagem, existem diversos métodos como a projeção da *point cloud* para um plano 2D, onde esta é processada com o uso de convoluções 2D. Nesse sentido, algum estudos fundem diretamente *features* da BEV com imagens RGB, enquanto outros projetam a *point cloud* no plano de imagem ou imagens RGB no plano BEV, com o objetivo de alinhar as fatures de ambos sensores.

AVOD, é o [27] utiliza uma arquitetura de *early fusion*, no qual utiliza projeções BEV de uma *point cloud* e imagens RGB para gerar *features* que são compartilhadas por duas sub-redes: uma RPN e um *two-stage detector*, como ilustrado na Figura 2.7.

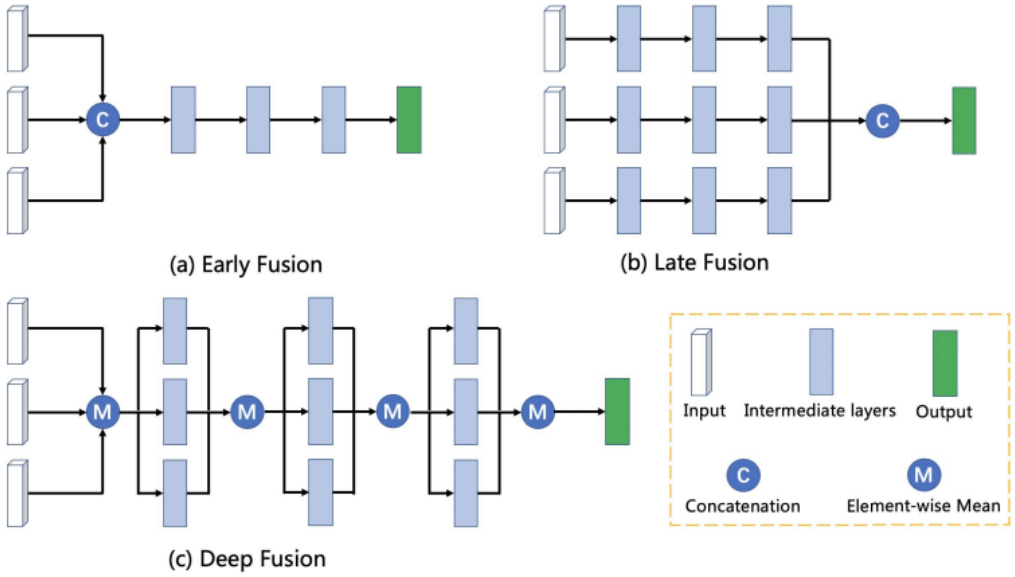


Figura 2.6: Ilustração dos três esquemas de fusão: *early fusion*, *late fusion* e *deep fusion* [10].

A RPN proposta (demarcada a rosa) segue uma arquitetura capaz de fundir as *features* multimodais extraídas pelos *feature extractors* (a azul), para então seguidamente gerar inúmeros *3D region proposals* através de uma FCN. Estes *3D region proposals* são transferidos para um *two-stage detector* que realiza uma regressão precisa das *3D bounding boxes* e classificação de classes, para então prever as extensões, orientação e classificação dos objetos no espaço 3D.

Uma contribuição do AVOD é que esta técnica codifica as caixas delimitadoras 3D através do deslocamento dos cantos superiores e inferiores em relação ao plano do solo e 4 cantos, em oposição à codificação padrão com 8 cantos ou alinhada pelos eixos.

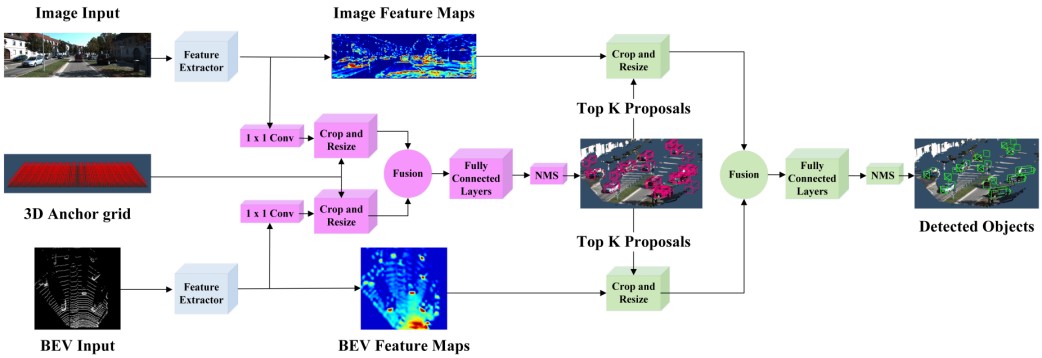


Figura 2.7: Arquitetura de *early fusion* da rede AVOD [27].

Uma abordagem alternativa é o F-PointNet [28], que tira proveito da maturidade dos 2D *object detectors* e a arquitetura PointNet/PointNet++ para detetar objetos 3D. O F-PointNet é um sistema que recebe imagens RGB e informações de profundidade (obtidas de um LiDAR) como entrada e é essencialmente composto por três módulos: *3D frustum proposal*, *3D instance segmentation* e *3D bounding box estimation*, segundo a representação gráfica na Figura 2.8.

No módulo *3D frustum proposal*, é adotado a rede *Feature Pyramid Network* (FPN) para gerar *2D region proposal* na imagem RGB, as quais delimitam o espaço de busca na *point clound*. Então cada *2D region proposal* é mapeada num *3D frustum proposal* correspondente, que inclui todos os pontos da *point clound* que estão dentro da região 2D proposta quando projetados no plano da imagem.

Em seguida, estes *3D frustum proposal* são encaminhadas para o bloco de *3D instance segmentation*. Aqui, utilizando a arquitetura PointNet, incluindo a T-Net, é realizada uma classificação binária para cada ponto, determinando se o ponto pertence ou não ao objeto detetado.

Finalmente, todos os pontos que são classificados como positivos são alimentados no módulo *3D bounding box estimation*. Neste módulo é utilizado outra PointNet para prever os parâmetros da *3D bounding box* para o objeto que está a ser detetado.

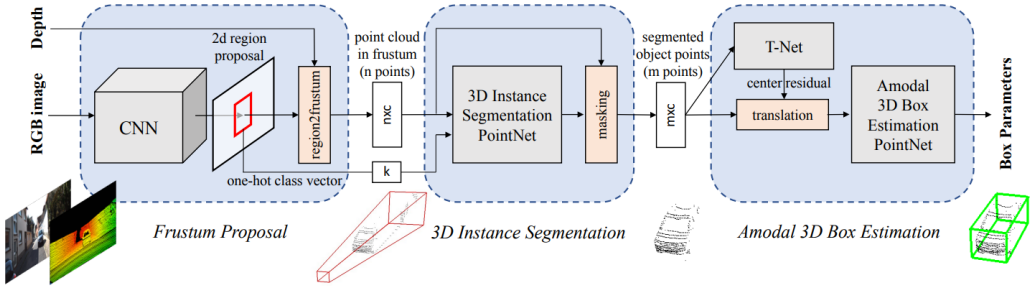


Figura 2.8: Arquitetura da F-PointNet[28].

Embora a F-PointNet tenha alcançado resultados excepcionais em benchmarks de detecção de objetos 3D, a sua arquitetura torna difícil a aprendizagem *end-to-end* (o sistema aprende todos os passos desde a fase inicial de entrada até o resultado final de saída.). Além disso, existem problemas de sincronização entre sensores, e a fusão de dados lidar com dados de câmera resulta em uma complexidade de tempo aumentada.

Tendo em conta isto, Shin et al. apresentaram a RoarNet [29], que visa melhorar a precisão de deteção 3D, abordando os problemas de sincronização entre as imagens 2D e *point clouds 3D* do LiDAR. O RoarNet é composto por dois módulos principais.

O primeiro componente utiliza uma imagem 2D para prever as *2D bounding boxes* e as poses 3D dos objetos presentes na imagem. Essencialmente, isto gera inúmeros

region proposals para cada objeto, o que efetivamente reduz o número de 3D *region proposals* viáveis.

Em seguida, o segundo bloco emprega estes 3D *region proposals* a um *two-stage detector*, que utiliza uma versão simplificada da PointNet sem T-Net como *backbone*. Esta *framework* reduz progressivamente o espaço de busca com base nos 3D *region proposals* obtidos previamente no primeiro módulo. Adicionalmente, ao contrário da F-PointNet, que usa *2D bounding boxes* para filtrar a *point cloud*, a RoarNet seleciona a porção da *point cloud* que está localizada dentro dos *region proposals*, tornando-a mais resiliente a problemas de sincronização entre sensores.

Em última análise, os métodos de fusão têm demonstrado ter melhor desempenho em comparação com os métodos baseados apenas em imagem ou em *point clouds*, principalmente devido à sua capacidade de combinar *features* de diferentes modalidades. Enquanto as *point clouds* oferecem informações geométricas, as imagens fornecem informações cruciais sobre textura necessárias para a identificação de classes. Ao combinar essas duas modalidades, os métodos de fusão utilizam informações discriminativas para melhorar o desempenho.

2.1.2 *3D Point cloud segmentation*

Nos últimos anos, *3D point cloud segmentation* tem apresentado avanços significativos com a introdução de técnicas de *deep learning*. Isto resultou na proposta de uma ampla variedade de sistemas de *3D segmentation* baseados em *deep learning* que superam até mesmo algoritmos tradicionais, alcançando melhores resultados em testes de *benchmark*. As técnicas de *3D segmentation* podem ser divididas em duas categorias principais: *semantic segmentation* e *instance segmentation*.

No caso da *3D semantic segmentation*, o foco está em prever *semantic labels* para cada ponto da *point cloud*, com o intuito de dividir a cena em seções relevantes com *labels* específicos para cada ponto. Por outro lado, *3D instance segmentation* tem como objetivo identificar as bordas dos objetos de interesse e gerar máscaras específicas para cada objeto, acompanhadas pelos seus respectivos *labels* de classe.

Devido às características do problema apresentado nesta dissertação, a quantidade de pontos presentes numa *point cloud* correspondentes às linhas de transmissão é reduzida e dispersa. Portanto, considerando a natureza de ambas técnicas de *3D point cloud segmentation*, a *3D semantic segmentation* é a técnica mais adequada para a detecção de *power lines*, uma vez que segmentar linhas de transmissão pelos contornos produzidos por estas na *point cloud* é uma tarefa desafiadora e ineficiente.

Os métodos de *3D semantic segmentation* de forma similar à tarefa de 3D *Object Detection* podem ser divididos em: *Projection-based Semantic Segmentation*, *Point-based Semantic Segmentation*

Projection-based Methods

As abordagens de segmentação semântica baseadas em projeção visam transformar a representação da *point cloud* 3D em uma imagem 2D. Desta forma esta pode ser ser processada por redes com os mesmos fundamentos utilizados no processamento convencional de imagens para 2D *semantic segmentation*, similar à abordagem apresentada em 2.1.1 para 3D *object detection* [30].

O SqueezeSegV3 [31] é uma versão aprimorada da arquitetura SqueezeSeg, cuja a sua *framework* está ilustrada na Figura 2.9. Assim como seus predecessores [32], o SqueezeSegV3 projeta a *point cloud* numa representação esférica, com o objetivo de transformar a disposição esparsa e invariante das *point clouds* numa imagem 2D. A escolha da representação esférica está fundamentada no facto de que a varredura de LiDAR típico se assemelhe a uma esfera.

Seguidamente, esta representação é então processada por um série de convoluções. Contudo, o uso de convoluções padrão para o processamento de imagens LiDAR não é o ideal, uma vez que os filtros destas capturam *features* locais que apenas são relevantes em regiões específicas da imagem. Isto, naturalmente, leva à subutilização da rede e à diminuição do desempenho de segmentação. Então a SqueezeSegV3 propõe o uso de *Spatially-Adaptive Convolution* (SAC) para processar a projeção esférica da *point cloud*. Desta forma a rede adota diferentes filtros para diferentes regiões espaciais de acordo com a imagem de entrada, aprimorando a eficaciada da mesma. Finalmente a rede gera uma *point-wise prediction* que é utilizada para segmentar a *point cloud* em secções especificadas com *labels* específicos para cada ponto.

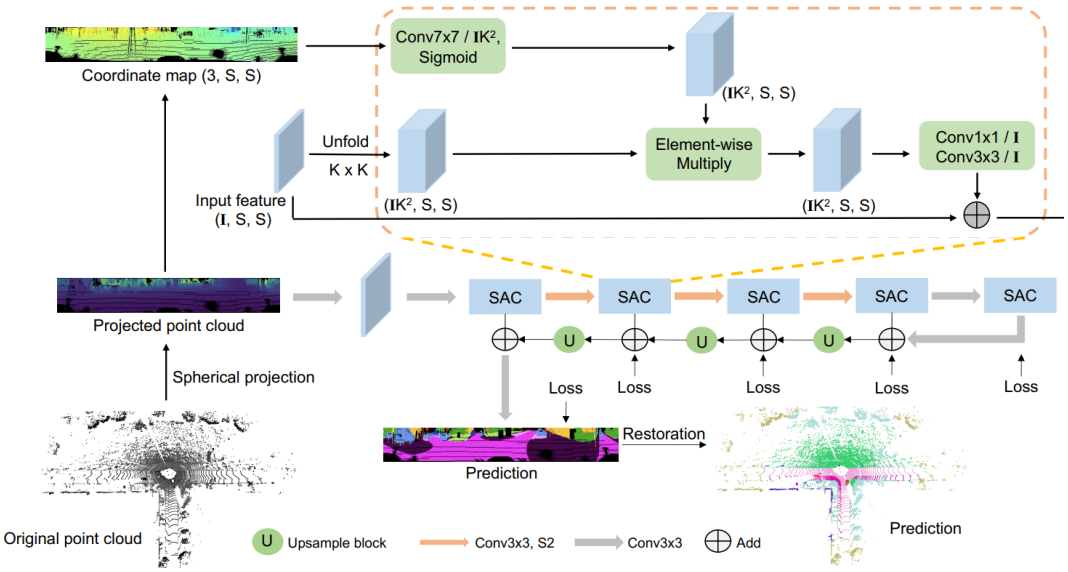


Figura 2.9: *Framework* da SqueezeSegV3[31].

SqueezeSegV3 é uma técnica eficiente e rápida em termos de processamento, no entanto, como depende exclusivamente da projeção esférica da *point cloud*, há inevitavelmente uma perda de alguma informação. Neste sentido, o método MPF [33] propõe uma técnica que explora as *spherical* e BEV *projections* da *point cloud* por meio de duas 2D CNNs altamente eficientes, que em seguida combina os resultados de segmentação de ambas as representações numa única predição. Assim, o MPF minimiza a perda de informação, permitindo obter resultados mais precisos quando comparado a outros métodos de *projection-based semantic segmentation*, sem comprometer significativamente a velocidade de processamento.

Point-based Methods

Similar aos métodos de *point-based 3D object detection* abordado no capitulo 2.1.1, *point-based semantic segmentation* visa capturar features diretamente dos dados da *point cloud*, sem convertê-la para outro domínio, como projetá-la em uma imagem bidimensional. Como previamente justificado, isto possibilita explorar a *point cloud* sem perder informação relevante.

PointNet [23] é uma das técnicas pioneiras neste contexto. O seu conceito, assim como a sua versão melhorada PointNet++, já foi abordado no capítulo 2.1.1 e, como mencionado, a sua arquitetura é flexível para várias tarefas de perceção, nomeadamente *semantic segmentation*. Analisando a versão indicada para *semantic segmentation*, ilustrado na Figura 2.4, a PointNet++ adota uma estrategia hierarquia inversa do *hierarchical point set feature learning*, onde cada *subset* de *features* é interpolado. Seguidamente, as *interpolated features* são concatenadas com as *skip linked point features* dos *abstraction level* correspondentes [24]. Sucessivamente, as *features* concatenadas são submetidas a *unit PointNet*. Este processo é então repetido até obter uma classificação para cada ponto.

Com a introdução da PointNet surgiram outras técnicas inspiradas, nomeadamente a RandLA-Net [34]. Esta é uma rede neural eficiente e leve especializada em *semantic segmentation* para *point clouds* de grande escala. Para processar *point clouds* de grande escala com centenas de milhares ou até milhões de pontos que se estendem por centenas de metros, é necessário reduzir progressivamente e eficientemente a densidade de pontos ao logo da rede sem perder *point features* relevantes.

Neste sentido, o ponto chave da RandLA-Net é a utilização da técnica de *Random Sampling* (RS) para reduzir drasticamente a densidade de pontos, possibilitando o processamento eficiente de *point clouds* de grande escala. Comparado ao método de downsampling da PointNet++, o FPS leva cerca de 200 segundos para processar nuvens de pontos de grande escala (com milhões de pontos), enquanto o RS demora cerca de 0.04 segundos. Esta diferença deve-se ao facto de que o RS tem um baixo grau de complexidade computacional, o que o torna independente do número total de pontos de entrada e utiliza um método de tempo constante, tornando-o inerentemente escalável.

No entanto, o *random sampling* pode descartar informações relevantes, especialmente para objetos com pontos esparsos, como linhas de energia. Então, para mitigar o potencial impacto prejudicial do *random sampling*, o RandLA-Net propõe um módulo eficiente de agregação de *features* locais (*local feature aggregator*), para capturar estruturas locais complexas num conjuntos de pontos.

De forma semelhante ao PointNet++, o RandLA-Net é implementado empilhando múltiplos módulos de *local feature aggregation* e *random sampling layers* (o equivalente a *set abstraction levels* na PointNet++), possibilitando capturar local

features progressivamente menores. Isto, essencialmente, permite a rede alcançar um excelente equilíbrio entre eficiência e eficácia.

Hybrid Methods

Hybrid Methods, conjugam várias técnicas ou abordagens de forma a melhorar a precisão e a robustez do processo de deteção de objetos 3D. Normalmente, esses métodos tiram partido de informações bidimensionais e tridimensionais para superar as limitações das suas abordagens individuais.

Uma abordagem de sucesso é o Cylinder3D [35]. Com o objetivo de proporcionar uma solução mais robusta para cenários *outdoor*, este método propõe uma estrutura inovadora para a segmentação de dados LiDAR, a qual combina *point-based methods* e *projection-based methods*. No Cylinder3D, é utilizada a técnica de *cylindrical partition* com o intuito de criar uma distribuição de pontos mais homogénea e menos suscetível a variações de densidade. Em seguida, esta abordagem é complementada pela implementação de *point-wise refinement module*, com o propósito de atenuar as interferências decorrentes da codificação da *point cloud* em *voxels*, a qual pode acarretar em perdas de informação.

Conforme ilustrado na Figura 2.10, o esquema é composto por dois componentes principais, nomeadamente a *cylindrical partition* e *asymmetrical 3D convolution networks*.

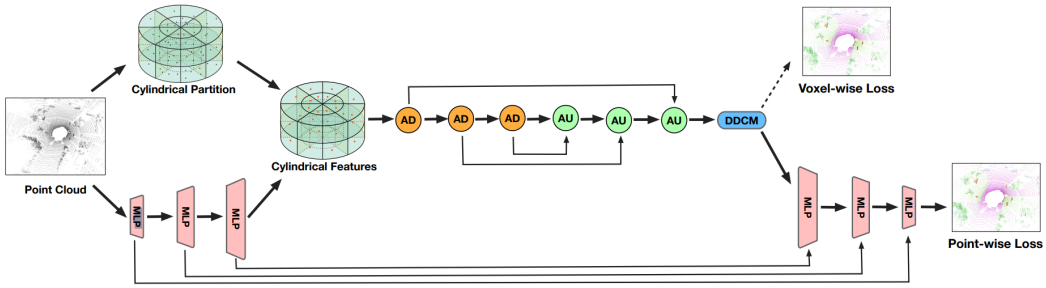


Figura 2.10: *Framework* do Cylinder3D [35].

De maneira concreta, o processo inicia-se com a divisão de uma nuvem de pontos através da *cylindrical partition* e, em seguida, as *features* extraídas por uma *Multi-Layer Perceptron* (MLP) são realocadas com base nessa partição. Posteriormente, são empregues *asymmetrical 3D convolution networks* para gerar *voxel-wise outputs*. Finalmente, é incorporado o *point-wise module* para minimizar as perdas de informação causada pela codificação imperfeita, o que resulta numa melhoria do resultado final.

Multimodal Fusion-Based Methods

Numa circunstância análoga aos *multimodal fusion-nased methods* para 3D *object detection* enunciado no Capitulo 2.1.1, as técnicas de 3D *segmentation* que se baseiam em imagens extraem apenas informações relacionadas com a textura, sem considerar informações acerca da profundidade. Por outro lado, os métodos de processamento que se fundamentam em *point clouds* obtêm informações acerca da profundidade, embora sejam deficientes no que diz respeito às informações de textura.

Então, considerando que a informação de textura é crucial para detetar e classificar objetos, enquanto a informação de profundidade é fundamental para a estimativa precisa da localização 3D e do tamanho dos objetos, algumas técnicas combinam imagens e *point clouds* para aprimorar o desempenho do sistema de segmentação. Para tal, em conformidade com a explicação no Capitulo 2.1.1, existem três esquemas de fusão: *early fusion*, *late fusion* e *deep fusion*, conforme exibido na Figura 2.6.

No artigo [36], Zhuang et al. apresentaram um esquema de fusão denominado de *Perception-aware Multi-sensor Fusion* (PMF), que tem como objetivo realizar uma fusão efetiva de informações perceptuais provenientes de imagens RGB e *point clouds*. Especificamente, como demonstrado na Figura 2.11, o PMF é composto por três componentes: (1) *perspective projection*, (2) *twostream network* (TSNet), (3) *perception-aware losses*.

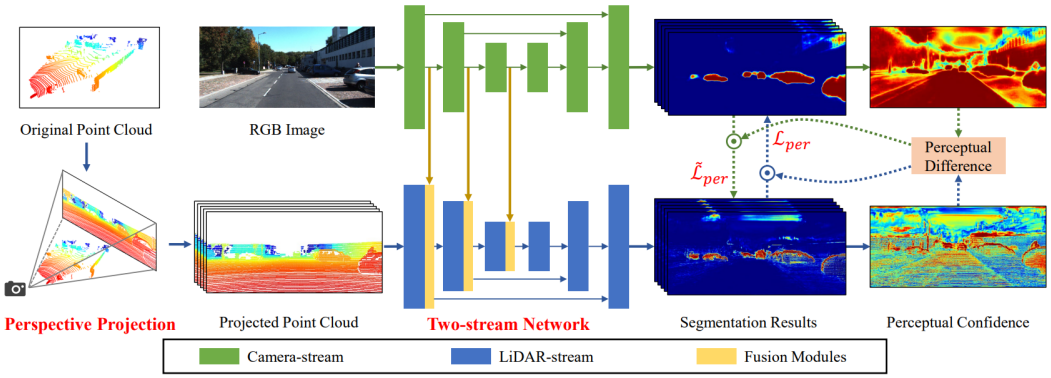


Figura 2.11: Ilustração da arquitetura de fusão PMF [36].

Na primeira fase, a nuvem de pontos é projetada no sistema de coordenadas da câmara com o intuito transferir as *features* da *point cloud* para 2D. Essencialmente, isto converte a nuvem de pontos numa imagem 2D que representa as *features* desta em cinco canais: (d, x, y, z, r). Sendo que d representa a distância de cada ponto, enquanto (x, y, z) indicam a posição do ponto no espaço tridimensional e r simboliza o valor de reflexão.

Em seguida, é utilizada uma *two-stream network* constituída por uma sub-rede indicada para imagens RGB e outra para a *point cloud* projetada, com o objetivo de extrair *features* perceptuais das duas modalidades, separadamente. Adicionalmente,

ao longo da rede, as *features* extraídas da sub-net da câmara são fundidas com as *features* extraídas da sub-net do LiDAR por meio de *residual-based fusion modules*, conforme ilustrado na Figura 2.12 e 2.11.

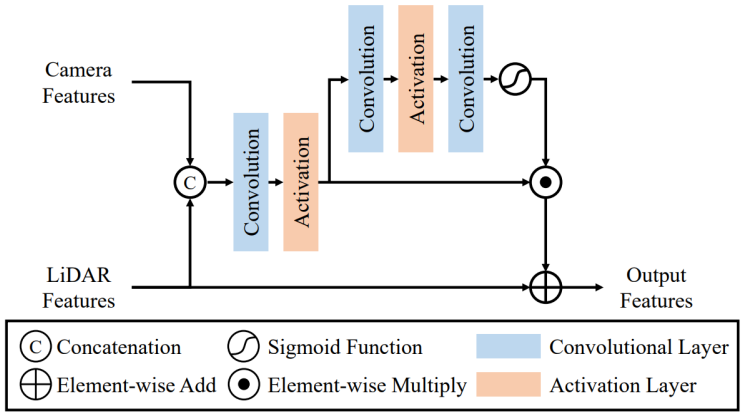


Figura 2.12: Esquema do *residual-based fusion (RF) module* [36].

Uma vez que as nuvens de pontos são muito esparsas, apenas as *features* locais dos pontos são aprendidas pela sub-net do LiDAR, ignorando a forma dos objetos. Em contraste, a forma e a textura dos objetos podem ser facilmente capturadas pela sub-net da câmara. Em suma, as *features* perceptuais extraídas por ambas sub-nets apresentam diferenças significativas. Então na fase final do processo, são incorporadas *perception-aware losses* com o objetivo de direcionar a rede de fusão às *features* perceptuais relevantes advindas da câmera e do LiDAR e desta forma atimizar a rede.

2.2 Discussão

Dada a diversidade de abordagens dos diversos métodos e enfoques, é crucial analisar os objetivos do projeto e optar pelo método mais apropriado. O objetivo principal deste projeto de engenharia consiste em identificar e prever a orientação da trajetória das linhas de transmissão.

Neste sentido, foram delineadas múltiplas soluções, voltadas para as duas tarefas distintas de operação: *3D Point cloud segmentation* e *3D Object Detection*.

Naturalmente, ambas as alternativas apresentam as suas próprias vantagens e desvantagens inerentes. A arquitetura de *3D point cloud segmentation* destaca-se pela sua capacidade de fornecer detalhes semânticos ricos sobre a cena. Além disso, distingue-se igualmente pela sua robustez em lidar com objetos ocultos, uma vez que não se fundamenta exclusivamente em caixas delimitadoras de objetos.

Contudo, em termos computacionais, a segmentação de cada ponto dentro de uma nuvem de pontos pode ser exigente em termos de recursos, sobretudo para

dados mais extensos. Além disso, embora a segmentação de nuvens de pontos seja exímia em detalhes locais, pode ter dificuldade em capturar o contexto global e as relações entre objetos distantes.

No que diz respeito ao *3D Object Detection*, por contraste, a metodologia empregue dispensa a necessidade de segmentar cada ponto de forma individual, que confere a este método uma eficácia superior. Acresce a isso, o facto de que este método assenta em informações contextuais globais do cenário, o que proporciona uma maior compreensão das interações entre os objetos e da dinâmica no ambiente.

Apesar disso, a deteção de objetos em 3D também apresenta as suas limitações. Nomeadamente, este método falha em fornecer o mesmo nível de informação semântica detalhada em comparação com a técnica de *3D point cloud segmentation*, uma vez que o seu propósito primordial é a deteção da presença e posicionamento dos objetos. Adicionalmente, a deteção de objetos parcialmente ocultos e a distinção entre múltiplas instâncias da mesma categoria de objetos representam um desafio substancial, dado que ambos os casos estão intrinsecamente dependentes das representações das caixas delimitadoras [10, 11, 12].

Dos métodos introduzidos para *3D point cloud segmentation* e *3D Object Detection*, aqueles que se destacam mais no contexto do projeto são o Cylinder3D e o CenterPoint, respetivamente. A razão fundamental subjacente a esta seleção baseia-se na proeminência de desempenho quando comparada com as alternativas, em grande medida devido à sua arquitetura híbrida. No caso, as duas metodologias têm como base *projection-based methods* e, posteriormente, empregam um *point-based module* para recuperar informações geométricas que foram perdidas durante a projeção da nuvem de pontos. Além disso, ambas as situações foram concebidas com enfoque na aplicação num contexto *outdoor*.

Considerando todos os pontos referidos e o contexto de aplicação, a metodologia mais apelativa é a *3D object detection* e, por conseguinte, o CenterPoint. De facto, as vantagens inerentes a esta abordagem demonstram uma relevância substancial, sem que as desvantagens associadas exerçam uma influência negativa significativa sobre o projeto.

Um beneficio claro em comparação à *3D point cloud segmentation* é facto que esta abordagem não apenas faculta a localização das linhas de transmissão, como também viabiliza a estimativa da sua orientação. No qual, ambas as informações são fornecidas pela localização e orientação da caixa delimitadora. Ainda, no que concerne à previsão precisa da orientação de objetos, o CenterPoint exibe uma excelência notória, constituindo assim mais um aspeto positivo a considerar.

Capítulo 3

Projeto

Este capítulo apresentada um *overview* da arquitetura de software definida para a implementação do sistema proposto, na qual são abordados todos os objetivos e motivações de todas as etapas constituintes da arquitetura.

3.1 Arquitetura

O principal objetivo deste projeto recai em detetar e prever a orientação da trajetória das linhas de transmissão. De acordo com o exposto no relatório, a abordagem adotada para realizar dita tarefa terá como base técnicas de *deep learning*.

Nesse sentido, a ideia fundamental é utilizar um 3D *object detector* para não só prever a posição, mas também estimar a direção das linhas, mediante a localização e orientação das *3D bounding boxes*.

Naturalmente, o sistema não se restringe apenas à parte de inferência da rede neural, sendo essencial realizar um pré-processamento e um pós-processamento. Na Figura 3.1 encontra-se ilustrado o esquema da arquitetura de software do sistema proposto, e conforme mencionado, este é principalmente composto por três etapas: pré-processamento, inferência e pós-processamento.

Pré-processamento

O pré-processamento de dados desempenha um papel crucial no sucesso de um modelo, seja durante o treino ou na fase de inferência. Durante esta etapa preliminar, os dados de entrada são preparados de forma adequada para garantir que o modelo

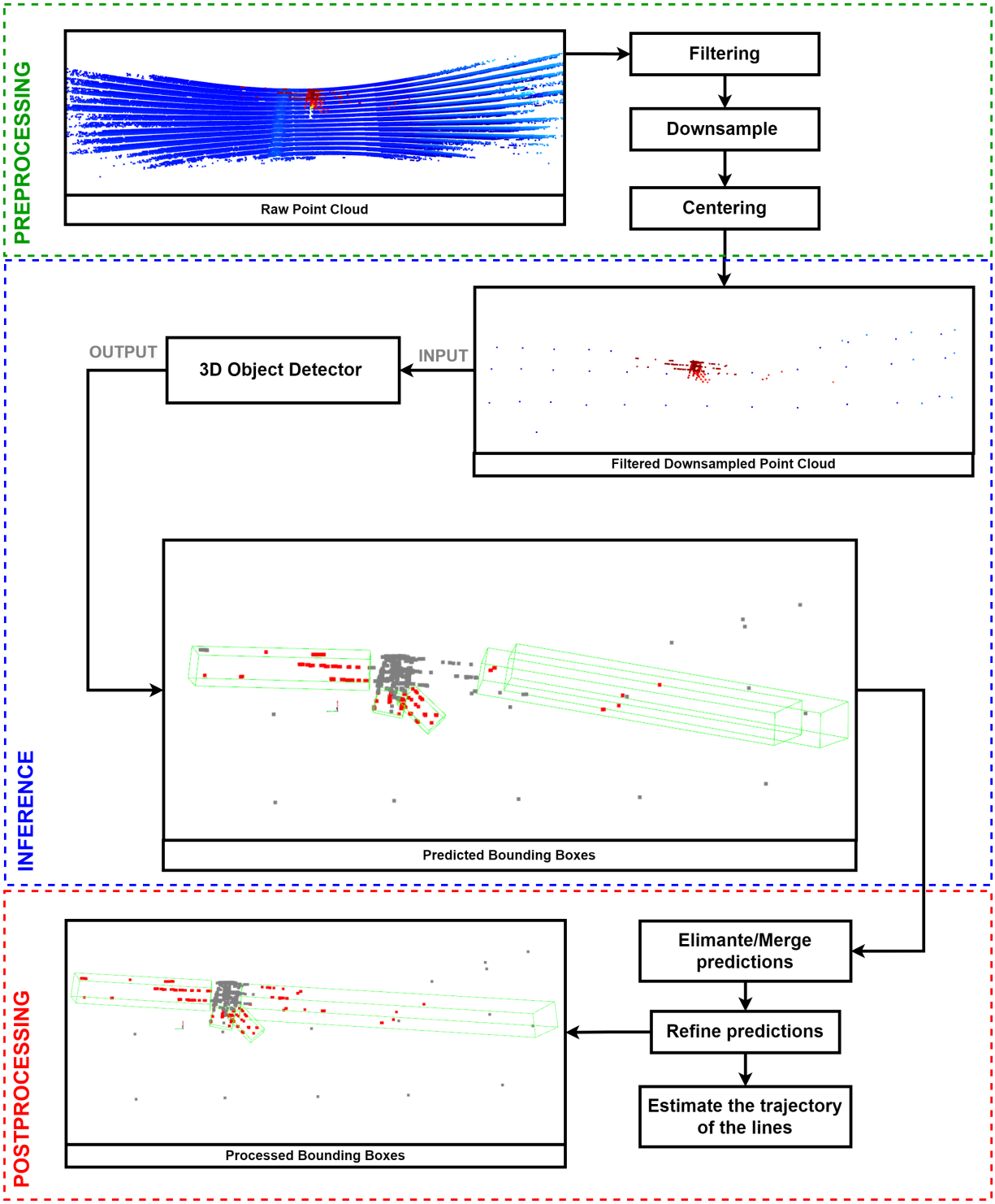


Figura 3.1: Representação gráfica da arquitetura de software do sistema proposto

possa aprender e generalizar com eficácia, com base nas informações disponíveis. Este processo de processamento é dividido em três passos: *Filtering and Rotating*, *Downsampling* e *Centering*

No bloco de *filtering*, são aplicados filtros para remover ruídos e *outliers* capturados pelo LiDAR. Posteriormente, o segmento de *downsample* é responsável por diminuir a densidade de pontos no solo e vegetação, deste modo o tamanho da *point cloud* é substancialmente reduzido sem comprometer informações relevantes das linhas de transmissão. Finalmente, na secção *centering*, a *point cloud* é centralizada em todos os eixos com o propósito de normalizar os dados de entrada da rede e, assim, assegurar uma escala e distribuição mais homogénea.

Este procedimento, naturalmente, além de preceder a inferência, é igualmente aplicado aos dados utilizados para treinar e, consequentemente, garantir a consistência entre treino e inferência.

Inferência

Como estabelecido, é empregado um 3D *object detector* para prever *3D bounding boxes*, que traduzem a localização e orientação da trajetória das linhas. Neste sentido, a *point cloud* processada é utilizada como entrada pela rede neuronal que, por sua vez, retorna como saída as coordenadas, dimensões e orientações das *3D bounding boxes* de todas as instâncias de linhas de transmissão detetadas. Idealmente, cada *3D bounding box* prevista deve englobar todos os pontos que correlacionam com as linhas na respetiva instância. Além disso, a orientação da caixa delimitadora deve se aproximar da direção das linhas nessa instância.

Relativamente à escolha do 3D *object detector*, é importante ter consideração que num cenário tridimensional, os objetos não seguem uma orientação específica, o que dificulta a deteção de todas as orientações dos objetos e no ajuste de uma *bounding box* alinhada com os eixos dos objetos rotacionados. Portanto, é crucial optar por um método que demonstre uma abordagem eficaz na extração da orientação dos objetos.

Considerando o exposto, a arquitetura do projeto e os requisitos estabelecidos, o CenterPoint [25] destaca-se como uma opção viável com um grande potencial, dado à sua capacidade de prever a orientação de objetos de forma precisa (Figura 3.2). O CenterPoint, já abordado em 2.1.2, é um *two-stage 3D detector* adequado para cenários outdoor, que alcançou resultados *state-of-the-art* no benchmark nuScenes tanto para deteção 3D quanto para rastreamento, com 65.5 *Normalized Detection Score* (NDS) e 63.8 *Average Multi-Object Tracking Accuracy* (AMOTA).

Conforme mencionado, esta metodologia utiliza uma representação centrada no objeto, o que traz benefícios como a diminuição do espaço de pesquisa, a aprendizagem da invariância rotacional e uma maior eficiência em comparação com outras abordagens.

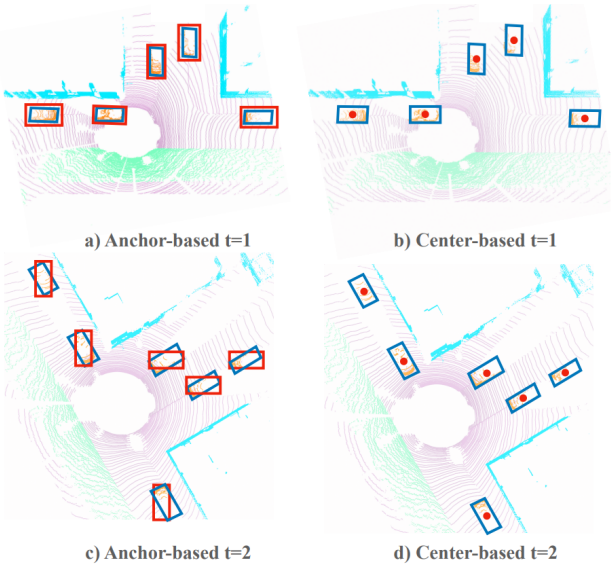


Figura 3.2: Comparação entre métodos tradicionais *(anchor-based methods)* e o CentePoint *(center-based method)* [25]

Pós-processamento

Nesta etapa conclusiva, procede-se ao tratamento dos dados de saída da rede com vista a obter os resultados refinados pretendidos. Este processo é constituído por três blocos: *Eliminate/Merge predictions*, *Refine predictions* e *Estimate the trajectory of the lines*.

Inicialmente, é imprescindível efetuar uma avaliação criteriosa dos diversos resultados da rede, neste ótica todas as previsões com um *score* inferior a um dado *threshold* são descartadas. Adicionalmente, as caixas delimitadoras que tentam representar a mesma instância de linhas são combinadas numa única previsão.

Sucessivamente, no segmento *Refine predictions*, é efetuada uma estimativa mais precisa da orientação das *3D bounding boxes*, com base nas características físicas das linhas.

Finalmente, para cada *3D bounding box*, calcula-se a equação da trajetória das linhas com base nos pontos contidos dentro dela.

Capítulo 4

Implementação

Neste capítulo, é realizada uma análise detalhada da implementação de todas as etapas do sistema. Primeiramente, é abordada a fase de pré-processamento, na qual é descrito todo o processo lógico, tal como as motivações que o fundamentam. A seguir, descrevem-se detalhadamente todos os passos realizados durante o processo de treino do *3D object detection model*. Finalmente, é analisado o processo lógico utilizado, assim como os seus fundamentos, para a concretização do pós-processamento.

4.1 Pré-processamento

O pré-processamento de dados é fundamental para o sucesso de um modelo, tanto no treino como na inferência. Nesta etapa, os dados de entrada são devidamente preparados para permitir que o modelo aprenda e generalize com eficácia. Conforme descrito no Capitulo 3, este processo encontra-se dividido em três subetapas: *Filtering and Rotating*, *Downsampling* e *Centering*. Estas subetapas contribuem para eliminar ruídos, reduzir a quantidade de dados e manter a coerência durante o treino e a inferência. Isto potencia a aptidão do modelo para aprender e aplicar as informações disponíveis com maior precisão.

Na Figura 4.1 está ilustrado o fluxograma que retrata o processo lógico da implementação das duas primeiras subetapas: *Filtering and Rotating*, *Downsampling*.

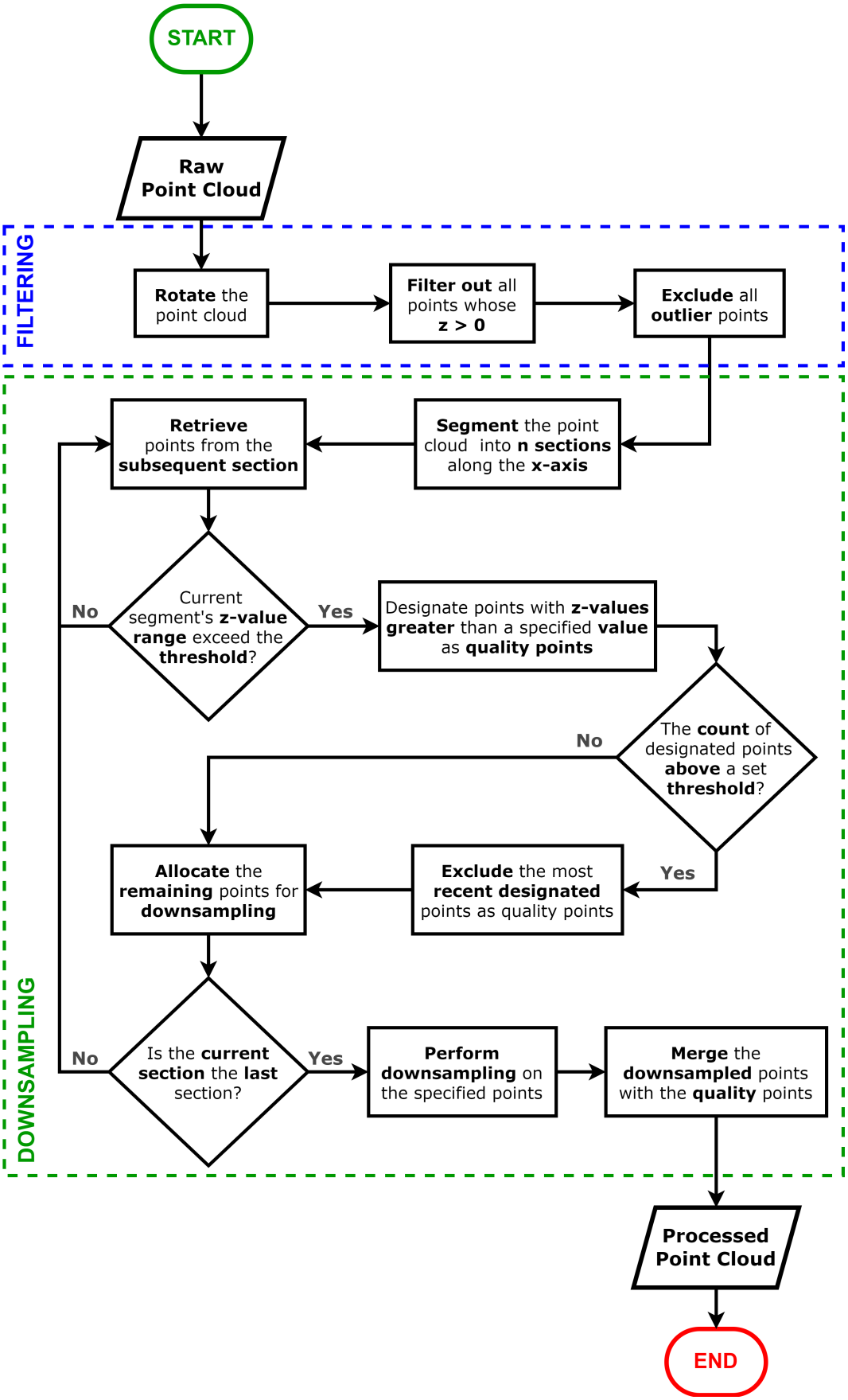


Figura 4.1: Diagrama que ilustra o processo lógico de implementação das duas primeiras subetapas: *Filtering and Rotating*, *Downsampling*

4.1.1 *Filtering and Rotating*

Na análise da secção azul da Figura 4.1, que se dedica à implementação de *Filtering and Rotating*, é verificado que o primeiro passo consiste em aplicar rotações na *point cloud*. O objetivo é realizar rotações na nuvem de pontos de forma a garantir a concordância entre o referencial desta e o referencial do UAV. Esta ação não apenas facilita a interpretação da nuvem de pontos, mas também leva em consideração a limitação do *object detector* 3D de prever apenas o ângulo de rotação *yaw*. Através desta transformação, é assegurado que o eixo z representa a altura do cenário, o que possibilita o *object detector* 3D extrair a orientação das linhas com precisão.

Os dois blocos subsequentes visam remover ruído e *outliars*. Resultante da transformação anterior, é estabelecido que pontos cujo valor no eixo z é igual a zero estão situados no mesmo plano do LiDAR, enquanto que valores negativos nesse eixo são interpretados como pontos abaixo do sensor, e, por sua vez, valores positivos são considerados acima do sensor.

Neste sentido, considerando o hardware alvo desta implementação, uma vez que o sensor LiDAR está posicionado na parte inferior da UAV a um ângulo de 90^{o}, os feixes que geram pontos acima do plano do LiDAR ou colidem imediatamente com o UAV ou geralmente não colidem com nenhum objeto. Inevitavelmente, esta situação propicia a ocorrência de ruído. Portanto, considerando esta constatação e o contexto de aplicação do sistema, todos os valores de z superior que zero são descartados.

No passo seguinte, são identificados *outliers* para posteriormente serem eliminados. Esta função é desempenhada através da análise dos valores máximos e mínimos de cada eixo. Especificamente, para cada eixo, é calculada a média dos n valores máximos, e todos os pontos que se encontram significativamente acima dessa média são descartados. Com naturalidade, o mesmo procedimento é aplicado de forma inversa aos valores mínimos de cada eixo, em que todos os pontos consideravelmente abaixo da média correspondente são eliminados. Essa prática é notoriamente importante, uma vez que na etapa subsequente são executados processos que são dependentes da análise dos valores máximos e mínimos em diferentes eixos.

4.1.2 *Downsampling*

Destacado a verde na Figura 4.1, podemos observar os blocos que correspondem à implementação do processo de *Downsample*. A premissa fundamental desta etapa é identificar potenciais pontos que correspondem a linhas , a fim de posteriormente reduzir a densidade dos demais pontos não identificados. Essa identificação é realizada com base nos valores máximos e mínimos de altura da nuvem de pontos, além do número de pontos presentes.

Ao analisar os componentes lógicos subjacentes a este processo 4.1, a primeira etapa consiste em segmentar a nuvem de pontos em n secções ao longo do eixo x. O propósito deste procedimento reside na atenuação do efeito adverso provocado pela vegetação e pela inclinação do terreno ou do UAV na identificação de possíveis pontos que representem linhas, assim como na otimização da qualidade do *downsampling*. Para uma melhor compreensão do motivo subjacente a esse efeito, é fundamental analisar o método de identificação.

A condição chave para a identificação é verificar se a discrepância entre a altura máxima e a altura mínima da secção excede um determinado *threshould*, o qual é definido conforme a altura do poste. A justificativa para tal afirmação reside no facto de que, quando existem pontos correspondentes a linhas presentes na secção a ser analisada, ocorre sempre uma diferença considerável entre a altura máxima e a altura mínima, sendo esta diferença estabelecida pela altura das linhas e postes. Para o *dataset* estudado, esta altura foi definida para 14 metros.

De forma sucessiva, identificam-se pontos que se localizam a uma altitude inferior a 5 metros da elevação máxima (*quality hight*) e são designados como *quality points* (potenciais pontos representativos de linhas). Naturalmente, a condição inicial é insuficiente para filtrar exclusivamente pontos que potencialmente se traduzem em linhas, visto que, por exemplo, na presença de vegetação alta, parte desses pontos é incorretamente classificada como *quality points*. Então, caso o número de pontos adicionados à última instância de *quality points* exceder um limite definido, esses pontos serão excluídos dessa designação. Este limite dependerá não apenas do número de *scans* do LiDAR presentes numa nuvem de pontos, mas também do número de segmentações aplicadas à mesma. Uma linha de referência apropriada para este limite é considerar 50 pontos por cada *scan* do LiDAR presente na *point cloud*, num cenário em que a mesma é dividida em 10 secções. Em conclusão, o ciclo de identificação termina, com o alocamento de todos os pontos não classificados como *quality points* para o *downsampling*.

Esta sequência é repetida para cada uma das secções, a fim de efetuar uma análise completa da nuvem de pontos. Esta abordagem, semelhante à técnica de *sliding window*, possibilita isolar vários casos numa nuvem de pontos que satisfazem a condição inicial. Desta forma, é possível aprimorar a viabilidade dos *quality points*, uma vez que isto facilita o reconhecimento de instâncias de pontos que foram erroneamente classificados como tal. Além disso, este método atenua o efeito da inclinação do terreno ou do UAV, visto que, num espaço delimitado, a altura em relação ao solo é mais constante, o que resulta numa condição inicial mais robusta e consistente.

Na Listagem 4.1, é possível observar o excerto de código em *Python* responsável por selecionar os *quality points* de cada secção.

```
for i in range(n_sections):
    if not(len(section[i])) : continue

    if (max(section[i][:,2])-min(section[i][:,2])) > 14 :
        quality_points.append(section[i][section[i][:,2]>
            max(section[i][:,2])- quality_height])

        if len(quality_points[-1]) > n_merged * n_p and (
           max(section[i][:,2])-min(section[i][:,2])) <
           18 :
            rest.append(quality_points[-1])
            quality_points.pop(-1)

        rest.append(section[i][section[i][:,2]< max(
           section[i][:,2])- quality_height])
    else:
        rest.append(section[i])
```

Listagem 4.1: Python code snippet para identificar *quality points*

Posteriormente, todos os pontos alocados para *downsampling* são submetidos à técnica de *voxel downsampling*. O processo de *voxel downsampling* utiliza uma *voxel grid* regular para reduzir uniformemente a densidade de pontos da nuvem de pontos de entrada. Essencialmente, o algoritmo funciona em duas etapas: Primeiramente, os pontos são agrupados em *voxels* e, em seguida, cada *voxel* ocupado gera um ponto único resultante da média de todos os pontos nele contidos. Nesta ótica, a magnitude do *downsampling* é regulada em função do tamanho do *voxel*.

Por fim, os *quality points* são integrados na nuvem de pontos resultante da etapa de *voxel downsampling*. Através disto, é viável reduzir substancialmente o número de pontos da *point cloud* , sem comprometer informações relevantes relacionadas às linhas de transmissão.

Dado o leque de valores de referência e *threshould* presentes no processo completo de *Downsampling*, é imprescindível encontrar valores que maximizem o efeito de downsampling sem comprometer informação relevante das linhas Esta solução algorítmica é capaz de reduzir o tamanho de uma *point cloud* para apenas $0,4\%$ a 6% do seu tamanho original, ao utilizar um *voxel* com tamanho igual a 12 unidades.

Centering

Finalmente, na etapa final de pré-processamento, a nuvem de pontos é centralizada em todos os eixos, com o intuito de normalizar os dados de entrada da rede e garantir uma escala e distribuição mais uniforme. Um exemplo concreto do impacto deste

processo é que confere ao sistema invariância em relação à altitude do UAV, uma vez que a centralização do cenário capturado normaliza as altitudes iniciais.

Relativamente à aplicação deste processo, é adotada uma abordagem adaptada do sistema utilizado pelo Supervisely para centralizar nuvens de pontos durante a inferência. Conforme será discutido mais adiante,o treino será conduzido por meio da ferramenta Supervisely. Neste sentido, é crucial empregar a mesma preparação de dados utilizada em seu processo.

A cerne deste algoritmo está na definição de uma caixa delimitadora, com base nas dimensões máximas registadas em cada eixo ao longo de todas as nuvens de pontos do *dataset* utilizado para treino.

Inicialmente, são filtrados os pontos que ultrapassam os limites impostos pela dimensão da caixa delimitadora. Seguidamente, é calculado o *offset* a ser aplicado em cada eixo, obtido através da soma do valor mínimo com metade da diferença entre o valor mínimo e máximo do respetivo eixo. Por fim, os *offsets* estipulados são empregados a cada eixo correspondente.

4.2 Treino

Nesta etapa, efetua-se o procedimento central do projeto, onde é empregado um 3D *object detector* para estimar a localização e orientação da trajetória das linhas, a partir da previsão *3D bounding boxes*. Naturalmente, a incorporação do detetor de objetos na sequência de procedimentos do projeto, exige o treino de um modelo adequado, capaz de desempenhar a tarefa mencionada anteriormente. Neste sentido, o processo de treino de um modelo é decomposto em três fases: *Data preparation*, *Labeling* e *Training*.

4.2.1 *Data preparation*

O procedimento *data preparation* engloba a aquisição, limpeza, transformação e estruturação dos dados de entrada de forma adequada, antecedendo a sua alimentação ao processo de treino do modelo. Dentro dessa ótica, é evidente que o passo inicial a ser executado compreende a recolha de um conjunto de dados apropriado. O *dataset* eleito para o estudo foi adquirido através do drone designado STORK II, durante uma missão efetuada sobre as regiões de Santo Tirso, no dia 10 de novembro de 2022.

O UAV STORK II, representado na Figura 4.2, constitui uma evolução do UAV STORK I, cujo desenvolvimento foi efetuado em prol da empresa nacional de energia elétrica EDP, mais especificamente para a EDP Labelec.

O UAV STORK I demonstra ser eficaz em ensaios de campo em várias situações de aplicação, incluindo operações de busca e salvamento, monitorização ambiental, mapeamento 3D, inspeção, vigilância e patrulhamento.

Dado que esta plataforma foi concebida com um caráter modular com capacidade de incorporar novos sensores, a principal distinção entre o STORK I e II reside na melhoria da precisão dos sensores de navegação, alcançada através da integração do STIM300 e do GPS RTK, além da substituição do LiDAR 2D Hokuyo por um LiDAR 3D da Velodyne (VLP-16). Na Tabela 4.1, estão evidenciadas as principais características do STORK II [37].

O *dataset* em análise é composto por 3212 *point clouds* capturadas pelo VLP16 a 10 HZ, ao longo de aproximadamente 5 minutos e meio. Este conjunto de dados apresenta uma variedade de cenários, sobretudo no contexto da vegetação, o que estabelece uma base robusta para o treino do modelo.

No decurso deste estudo, foram estabelecidos três *datasets* distintos. O primeiro conjunto de dados é constituído mediante a seleção, a cada segundo, de uma única nuvem de pontos, composta por um total de 312 elementos. O segundo e o terceiro *datasets* seguem um procedimento semelhante, contudo, em vez de selecionar apenas uma única nuvem de pontos, são combinadas as últimas quatro e as últimas dez nuvens de pontos, respetivamente, numa única entidade. Inerentemente, a fusão

Tabela 4.1: Características fundamentais do STORK II [37]

Features	**STORK II**
Size open (DxH)	1060 x 570 mm
Size close (DxH)	670 x 570 mm
Motors	6 x DJI E2000 Pro
Weight (with batteries)	11 kg
Batteries	22.2 V - 22000 mAh
Flight time	20 - 25 min
Navigation	
GNSS	RTK GPS Unicore UB482 dual antenna
Autopilot	INESC TEC autopilot
IMU	STIM300
Perception	
Cameras	Teledyne Dalsa Genie Nano GiE
IR Camera	Teledyne Dalsa Calibir DXM640
LiDAR	Velodyne VLP-16
CPU	i7-6822EQ with 16 GB of RAM DDR4

Figura 4.2: STORK II: UAV usado para recolha do *dataset*

das *point clouds* decorre em conformidade com a pose de aquisição de cada uma, contudo, a pose gerada pelo STORK II não está sincronizada com o *scan* do VLP16, o que acarreta um ligeiro desfasamento. De todo modo, é válido assinalar que, na maioria significativa das situações, tal discrepância não exerce influência substancial na qualidade das nuvens de pontos. A motivação subjacente à conceção de três *datasets* reside na análise do impacto decorrente da combinação de múltiplos *scans* lidar para a *point cloud* de entrada do *object detector*, com foco na avaliação da sua

eficácia e eficiência.

Prosseguindo para as tarefas subsequentes desta fase, todas elas serão realizadas pelo intermédio da implementação do pré-processamento descrito no Capítulo 4.1. Tal como referido, este processo em questão terá como objetivo a adequada preparação dos dados, permitindo, assim, que o modelo seja capaz de aprender e generalizar de forma eficaz.

No que concerne às configurações definidas para o processamento dos conjuntos de dados, verifica-se uma coerência entre as mesmas, com a única discrepância a residir no *voxel size*. Especificamente, foram adotados os valores de referência mencionados ao longo do subcapítulo 4.1. No qual no primeiro *dataset* é utilizado um *voxel size* de 9 unidades, enquanto nos conjuntos restantes, o valor adotado é de 12 unidades. Como resultado, são gerados três conjuntos de dados devidamente processados, os quais se encontram prontos para as etapas posteriores de *labeling* e *training*.

4.2.2 Labeling

Um bom processo de *labeling* é essencial na criação de dados de treino de alta qualidade, com o efeito de garantir a precisão e eficácia do modelo treinado para a deteção de objetos em nuvens de pontos 3D. Neste processo, são atribuídas anotações precisas de caixas delimitadoras 3D a pontos individuais ou grupos de pontos que descrevem a localização exata, a extensão dos objetos no espaço tridimensional e a classe de cada anotação. Desta forma, os dados rotulados são utilizados como referência para então treinar e validar os modelos.

Naturalmente a precisão do *labeling* afeta diretamente o desempenho e as capacidades de generalização dos modelos. Portanto, é essencial garantir que as anotações sejam devidamente atribuídas, e assim assegura a uniformidade na distribuição e mantendo uma qualidade constante das anotações da mesma classe.

Considerando o conjunto de soluções disponíveis para o *labeling* de imagens, a variedade de ferramentas para *labeling* de *point clouds* é significativamente mais limitada, sendo o Supervisily a opção mais proeminente. O Supervisely é uma plataforma de visão computacional que fornece ferramentas e infraestrutura para anotar, gerir e treinar modelos Inteligência Artificial (IA), não só para imagens, mas também para nuvens de pontos. Esta é projetada para auxiliar na criação de *datasets* de treino de alta qualidade de forma eficaz e eficiente.

Primeiramente, para estabelecer quais objetos devem ser rotulados e de que modo, é importante ter em consideração qual é a previsão pretendida pelo modelo.

Conforme mencionado, o objetivo do modelo é realizar a previsão de *oriented bounding boxes* para todas as instâncias que correspondam a linhas de transmissão presentes numa nuvem de pontos. Especificamente, cada *oriented bounding boxes* terá de incluir todos os pontos pertencentes às linhas de transmissão nessa instância,

e a sua orientação deverá estar em concordância com a trajetória percorrida pelas linhas. Neste sentido, o *labeling* terá de se traduzir numa previsão ideal, para então modelar o modelo de forma pretendida.

Na Figura 4.3, podemos observar uma secção da interface gráfica da *3D LABELING TOOLBOX* do Supervisely. Esta solução apresenta uma forma simples de navegar e anotar, utilizando os controlos WASD do teclado para deslocar-se na nuvem de pontos e o rato para controlar o ângulo da câmara. Adicionalmente, existem três painéis complementares que exibem vistas do topo, lateral e frontal, por meio de projeções ortográficas. Isto não apenas oferece uma representação precisa do cenário em questão, mas também proporciona ao utilizador uma metodologia precisa de estipular as dimensões e orientações do ortoedro.

Na analise da anotação representada na Figura 4.3 do primeiro *dataset* de apenas um *scan*, é possível então observar o que é considerado uma previsão ideal. A caixa delimitadora é dimensionada de modo a abranger todos os pontos, garantindo sempre uma margem adequada em todos os eixos. Por outro lado, o processo de recuperar a orientação da trajetória das linhas que não é tão direto é alcançado através da variação do *yaw* da *bounding box* até que os pontos da mesma linha se alinhem na vista frontal. Na Figura 4.4 está representado um exemplo deste processo.

Certamente, é imperativo que este procedimento seja reiterado para todas as *point clouds* dos três conjuntos de dados, uma vez que, apesar da existência de uma correspondência temporal entre estes, nem sempre existe uma correspondência de linhas entre os *datasets*.

Notoriamente, o processo de *labeling* nem sempre é tão explícito no dataset de apenas um *scan* LiDAR, como o exemplo ilustrado nas Figuras 4.3 e 4.4, onde são capturados múltiplos pontos das linhas. Dado que, em instâncias de linhas onde o numero de pontos é escasso, é impraticável determinar a orientação das linhas. Teoricamente, numa transmissão padrão com um agrupamento de três linhas, são necessários pelo menos quatro pontos para garantir a possibilidade de determinar a orientação da trajetória das linhas. No entanto, é possível fazê-lo apenas com dois pontos, desde que esses dois pontos pertençam à mesma linha.

4.2.3 Training

Após a aquisição de dados devidamente rotulados, os quais desempenham o papel de *ground truth*, o subsequente processo de treino incorpora múltiplas fases essenciais com o propósito de dotar a rede neural da capacidade de assimilar e extrapolar padrões a partir dos dados disponibilizados.

Para a concretização desse processo, é utilizada a ferramenta MMdetection3D [38]. O MMdetection3D é uma plataforma *open-source* integrada no projeto OpenMMLab, com foco em tarefas de deteção e segmentação de objetos em espaços tridimensionais. Esta framework é desenvolvida sobre a plataforma MMdetection [39],

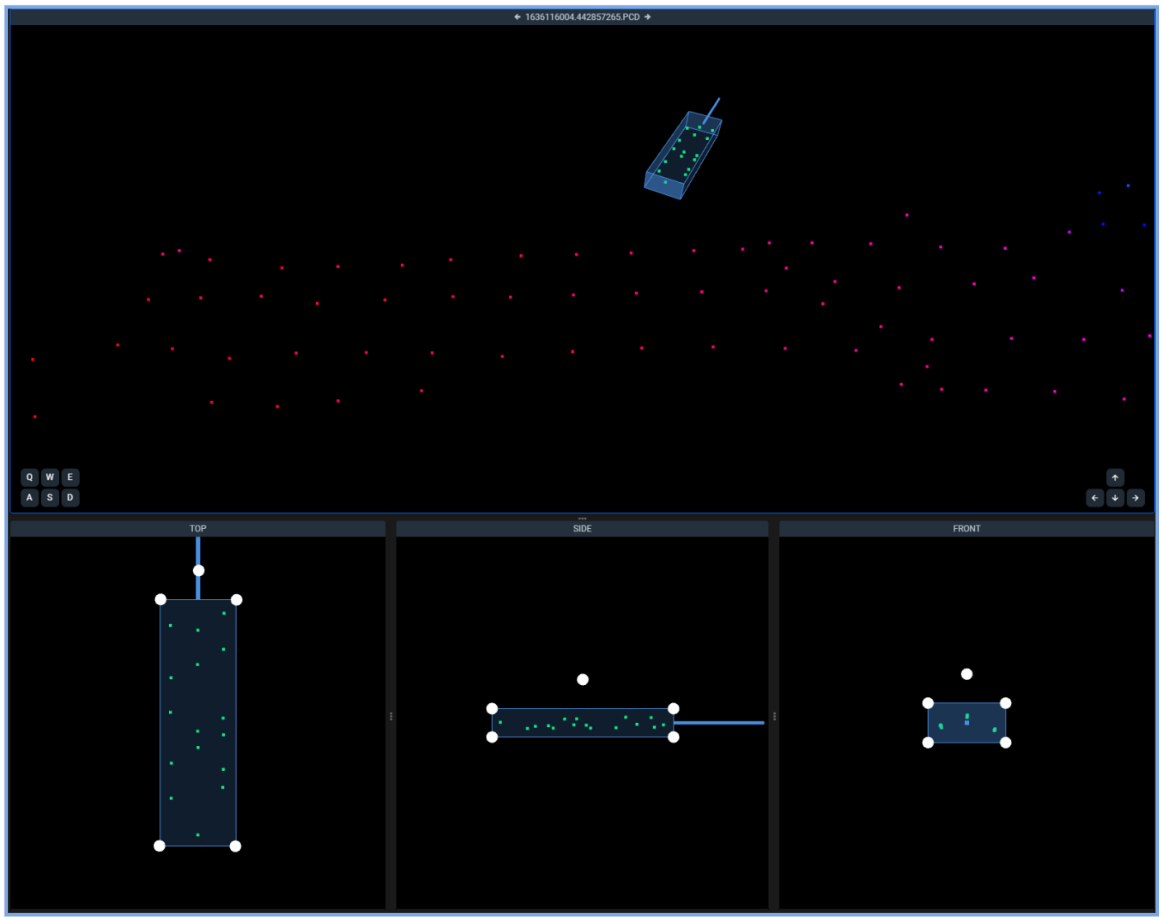


Figura 4.3: Secção da interface gráfica da 3D LABELING TOOLBOX do Supervisely.

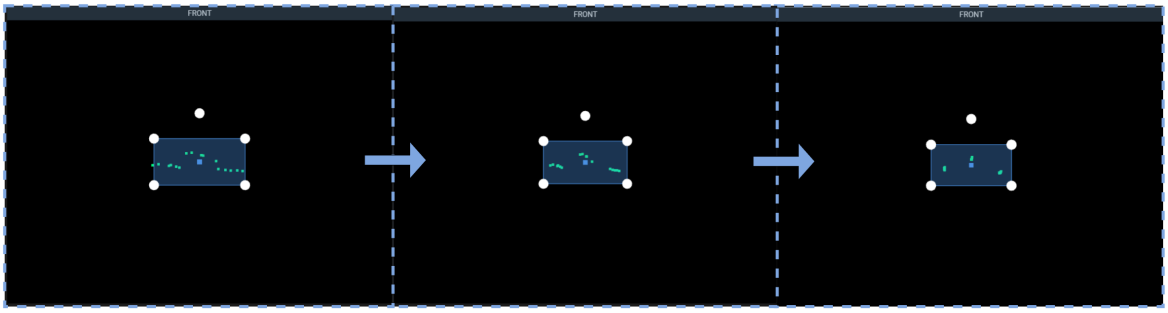


Figura 4.4: Representação do processo de recuperar a orientação da trajetória das linhas.

originariamente criada para deteção de objetos em 2D e segmentação de instâncias. O MMdetection3D expande esta funcionalidade para o domínio tridimensional, com um enfoque particular nas tarefas de deteção de objetos e segmentação instâncias em *point clouds* ou 3D *voxel grids*.

Desenvolvida em PyTorch, esta plataforma disponibiliza uma coleção diversificada de modelos *state-of-the-art* de *3D object detection* de arquiteturas distintas, nomeadamente *LiDAR-based 3D Object Detection*, *Camera-based 3D Object Detection*, *Multi-modal 3D Object Detection* e *3D Semantic Segmentation*. Sendo que, para cada arquiteturas são fornecidas várias alternativas especializadas para cenários *outdoor* e *indoor*.

De notar, o CenterPoint [40], juntamente de outras arquiteturas previamente abordadas, tais como o SECOND [22] , PointPillars [25], PointNet++ [23] e Cylinder3D [41], é incorporado na lista de arquiteturas suportadas. Além destas abordagens estabelecidas, há também alternativas mais recentes, a exemplo o TPVFormer [42] e FCAF3D [43].

Como complemento a esta *framework*, é empregada a aplicação do *Train* MMDetection3D do ecossistema do Supervisily para o processo de treino. A principal fundamentação para a incorporação desta aplicação reside na sua capacidade de simplificar todo o processo de treino do MMDetection3D, especialmente quando o treino é realizado sobre um *custom dataset*.

O software abordado incorpora múltiplas funcionalidades essenciais, incluindo a conversão de anotações no formato Supervisely para o formato requerido pelo MMDetection3D para o treino de datasets personalizados. Além disso, efetua a divisão do *dataset* em *train/validation splits* e também faculta *training augmentations* e a otimização dos hiperparâmetros de treino.

Neste contexto, os três conjuntos de dados serão sujeitos aos mesmos parâmetros de treino, de modo a viabilizar uma comparação válida. Estes parâmetros foram fundamentalmente estipulados de forma empírica, através da realização de inúmeros treinos com os diferentes *datasets* e parâmetros variados.

Training Configurations

Como ponto de partida, é adotado um modelo pré-treinado CenterPoint, o qual foi previamente treinado utilizando o *dataset* nuScenes. Em especifico, o modelo disponibilizado pelo OpenMMLab é designado como centerpoint_0075voxel_second_secfpn_dcn_circlenms_4x8_cyclic_20e_nus, sendo este nome representativo das configurações do referido modelo. No contexto em questão, cada campo representa, de forma respetiva, o tipo de modelo, o tamanho do *voxel*, o *backbone*, o *neck*, caso seja aplicado *deformable convolution*, a decisão em utilizar *circular nms* (non-maximum suppression), a proporção de GPUs e as amostras por GPU, o *training schedule* e o *dataset*.

A proporção padrão para a divisão entre conjuntos de *train/validation* foi de 80%/20%, sendo relevante notar que esta distribuição foi aleatoriamente efetivada.

Prosseguindo com as *training augmentations* estipuladas, nesta etapa objetivou-se expandir cada *dataset* ao incorporar maior diversidade nas variações angulares das

linhas. Para este propósito, efetua-se uma rotação global da cena, com uma variação angular mínima de $-\frac{\pi}{5}$ e uma variação angular máxima de $\frac{\pi}{5}$. Adicionalmente, é definida uma probabilidade base de 20% de aplicar um *flip* horizontal do cenário.

No que concerne aos hiperparâmetros de treino, é definido um total de 600 *epochs* para cada conjunto de dados, com um *batch size* de 8, com 8 *workers*. Ademais, é estabelecido que a validação ocorre após o término de cada *epoch*.

Continuando com as escolhas referentes ao *optimizer*, foi selecionado o algoritmo ADAM nas suas definições padrão do Pytorch [44], exceto pelo ajuste na *learning rate* (lr) padrão para um valor base de 3e−3. O incremento do valor padrão da lr é fundamentado devido à utilização de um *step lr scheduler*, configurado com um intervalo de decaimento da lr de 200 *epochs* com *ratio* de decaimento de $0,333$. Isto significa que, ao longo dos primeiros 200 *epochs*, a lr é de 3e−3. Para os 200 *epochs* seguintes, a lr diminui para 1e−3. Nos últimos 200 *epochs*, a lr é estabelecida em 3.33e−4.

Métricas de avaliação

Na implementação de algoritmos de *deep learning*, a utilização de métricas específicas assume um papel fundamental na avaliação meticulosa do desempenho destes algoritmos. No contexto de *3D Object Detection*, estas métricas, derivadas das suas equivalentes bidimensionais, são adaptadas à complexidade do espaço tridimensional para avaliar a capacidade dos modelos em estimar com exatidão as posições e dimensões dos objetos num espaço tridimensional. Para o efeito, são empregues duas métricas para a avaliação dos modelos treinados: *mean Average Precision* (mAP) e *mean Average Recall* (mAR).

Neste âmbito, destacam-se dois conceitos essenciais:*precision* e *recall*. A *precision* mede a exatidão das previsões, ou seja, a percentagem de previsões corretas. Isto corresponde à proporção de instâncias previstas como positivas que foram corretamente classificadas como tal. O *recall* reflete a eficácia na identificação de todos os exemplos positivos. Ou seja, calcula a proporção de instâncias corretamente previstas como positivas em relação ao número total de instâncias reais.

Deste modo, o mAP, apesar da sua designação, funde estas duas noções. Do ponto de vista matemático este corresponde ao integral definido pela *precision-recall curve*. Ademais, esta métrica encontra-se invariavelmente associado a um valor específico, o qual representa o *theshould* da *Intersection over Union* (IoU) utilizado para estabelecer se uma previsão positiva é devidamente categorizada como tal. No qual, IoU é um valor que procura quantificar a percentagem de *overlap* entre uma *predicted bounding box* e a *ground-truth*. Então, um valor elevado do *theshould* da IoU requer uma maior concordância entre as previsões do modelo e as instâncias reais.

Enquanto o mAP leva em consideração os conceitos de *precision* e *recall*, o mAR concentra-se exclusivamente na avaliação do *recall*. A sua computação envolve a determinação da média do *recall* em várias faixas de *precision*. Em outras palavras, o mAR avalia a taxa de deteção de objetos em diferentes níveis de precisão. E assim como mAP, está vinculado a um valor de IoU [45].

Tanto a mAP quanto a mAR desempenham papéis cruciais na avaliação abrangente de algoritmos de deteção de objetos a três dimensões. Enquanto a mAP proporciona um equilíbrio entre *precision* e *recall*, a mAR enfatiza diretamente a capacidade do modelo em identificar todos os objetos relevantes.

4.2.4 Resultados do Treino

Na sequência dos tópicos abordados, efetuou-se um treino idêntico para cada *dataset*, com o devido registo das métricas de avaliação ao longo do processo de treino.

Deste modo, para os respetivos *datasets*, as Figuras 4.5, 4.8 e 4.11 apresentam a evolução das *losses* e da *lr* ao longo de cada treino. Por sua vez, as Figuras 4.6, 4.9 e 4.12 realçam as métricas mAR0.25 e mAR0.50, enquanto as Figuras 4.7,4.10 e 4.13 apresentam os valores de mAP0.25 e mAP0.50.

Dataset composto por *point clouds* derivadas de 1 *scan* LiDAR

Ao analisarmos a *loss trend* retratada na Figura 4.5, podemos deduzir que esta exibe um perfil característico de uma lr adequada, estabilizando-se por volta de 0.6.

Já a análise dos gráficos apresentados na Figura 4.6 demonstra que, quando o *threshold* de IoU é definido como 0.25, é possível concluir que este é suficientemente competente para detetar a maioria das instâncias. No entanto, naturalmente, quando o *threshold* de IoU é aumentado para 0.5, observa-se uma deterioração significativa na eficácia do sistema, uma vez que este apenas consegue identificar uma fração das linhas. Finalmente, no que concerne à métrica mAP representada na Figura 4.7, constata-se uma situação semelhante à métrica mAR, na qual o sistema perde uma parte significativa da eficácia quando o IoU é incrementado de 0.25 para 0.5.

No que diz respeito ao melhor *checkpoint* do treino, este ocorreu no *epoch* 358 e alcança uma mAR0.25 de 0.807692, uma mAR0.5 de 0.307692, uma mAP0.25 de 0.736296 e uma mAP0.5 de 0.180897. No caso, esta seleção foi efetuada com base no valor mais elevado de mAP0.25

Para concluir, após a análise de todas as métricas, pode-se afirmar que o sistema demonstra eficácia na deteção da maioria das instâncias de linhas. Contudo, a sua capacidade de estimar a localização e orientação revela-se pouco precisa, devido à sua ineficiência em IoUs mais elevados.

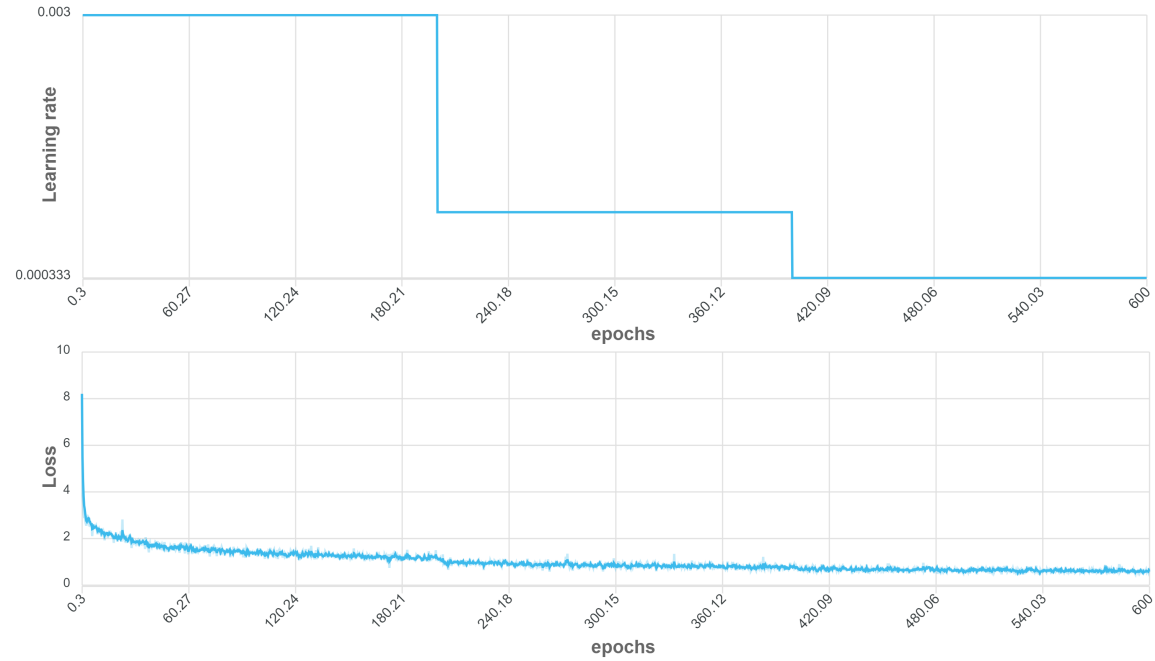


Figura 4.5: Gráfico ilustrativo da lr e das *losses* ao longo do processo de treino primeiro modelo

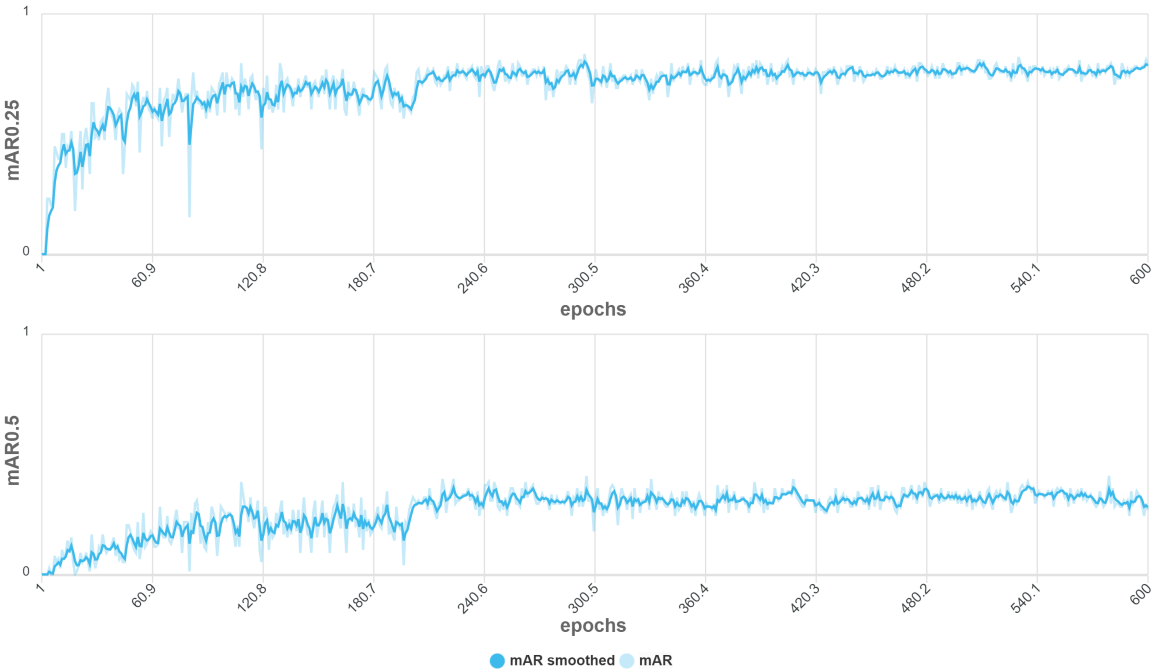


Figura 4.6: Representação gráfica das métricas mAR0.25 e mAR0.50 durante o processo de treino do primeiro modelo

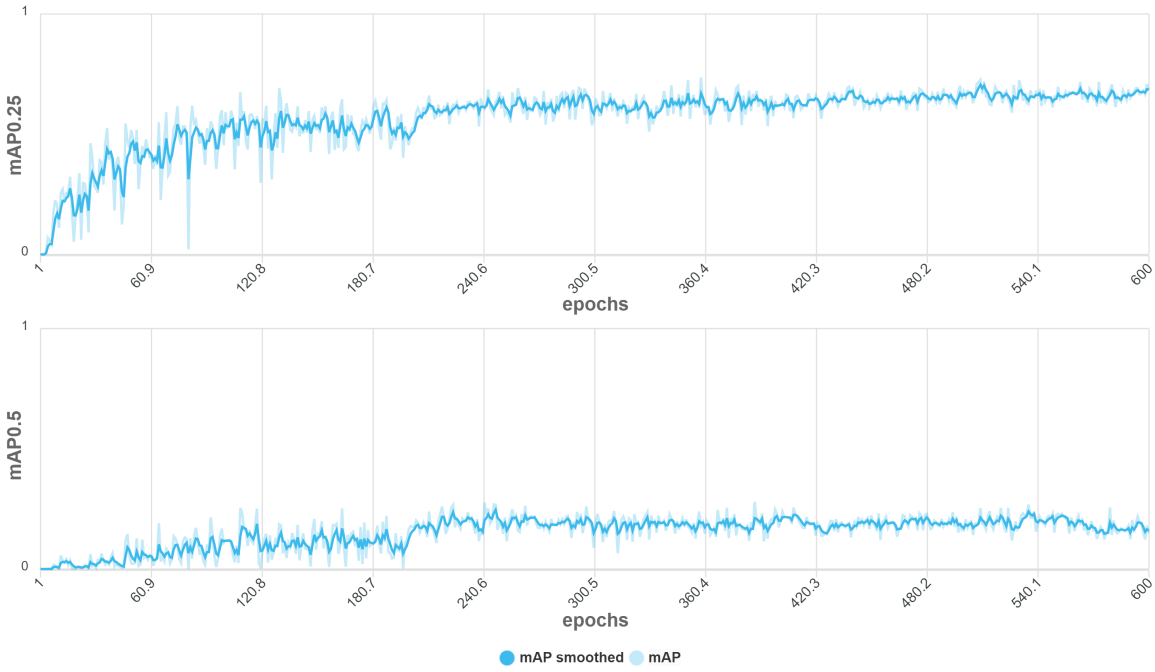


Figura 4.7: Representação gráfica das métricas mAP0.25 e mAP0.50 durante o processo de treino do primeiro modelo

Dataset composto por *point clouds* derivadas de 4 *scans* LiDAR

Conforme o esperado, a *loss trend*, ilustrada na Figura 4.8, apresenta um perfil típico de uma lr apropriada, no qual se estabiliza por volta de 0.2.

Tal como o modelo anterior, o sistema demonstra uma redução de eficácia em ambas as métricas representadas nas Figuras 4.9 e 4.10 à medida que o valor do *threshold* IoU é aumentado para 0.5. Contudo, é relevante notar que esta redução de eficácia é menos acentuada do que aquela observada no modelo treinado com o *dataset* com apenas 1 *scan* LiDAR, para ambas as métricas.

Em concreto o melhor *checkpoint* do treino, dá-se no *epoch* 569, com os seguintes valores métricos: uma mAR0.25 de 0.911765, uma mAR0.5 de 0.573529, uma mAP0.25 de 0.899989 e uma mAP0.5 de 0.457373.

Com base nestas métricas, é seguro afirmar que o segundo sistema evidencia um desempenho notavelmente superior ao do primeiro modelo, não apenas na capacidade de deteção das instâncias de linhas, mas também na precisão da previsão da sua orientação e localização.

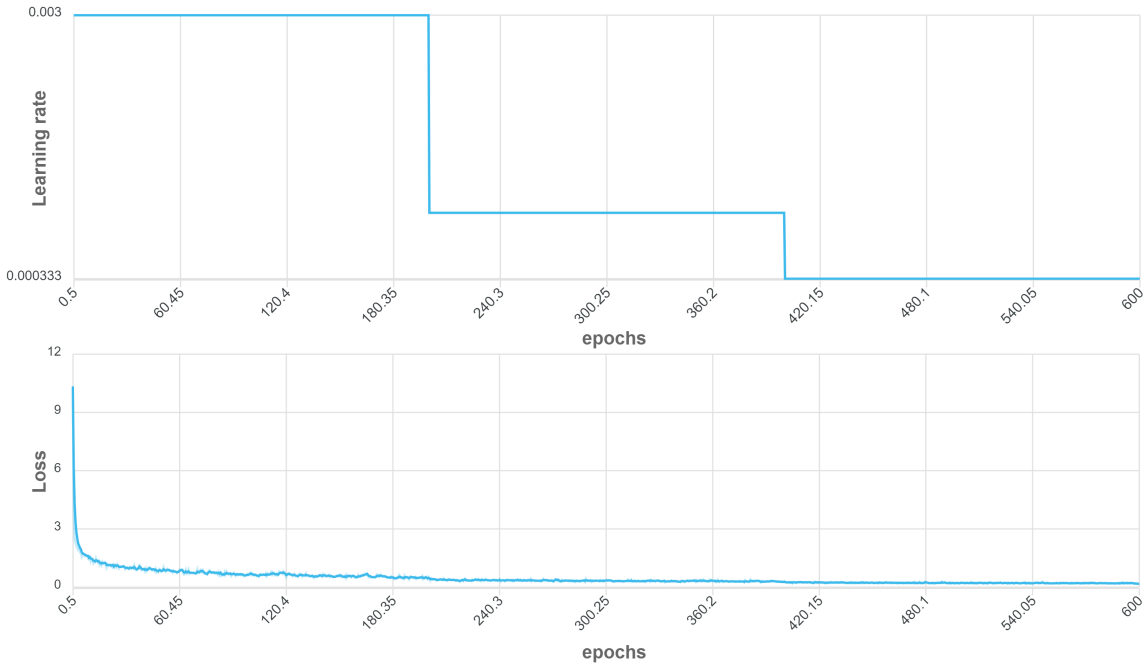


Figura 4.8: Gráfico ilustrativo da lr e das *losses* ao longo do processo de treino do segundo modelo

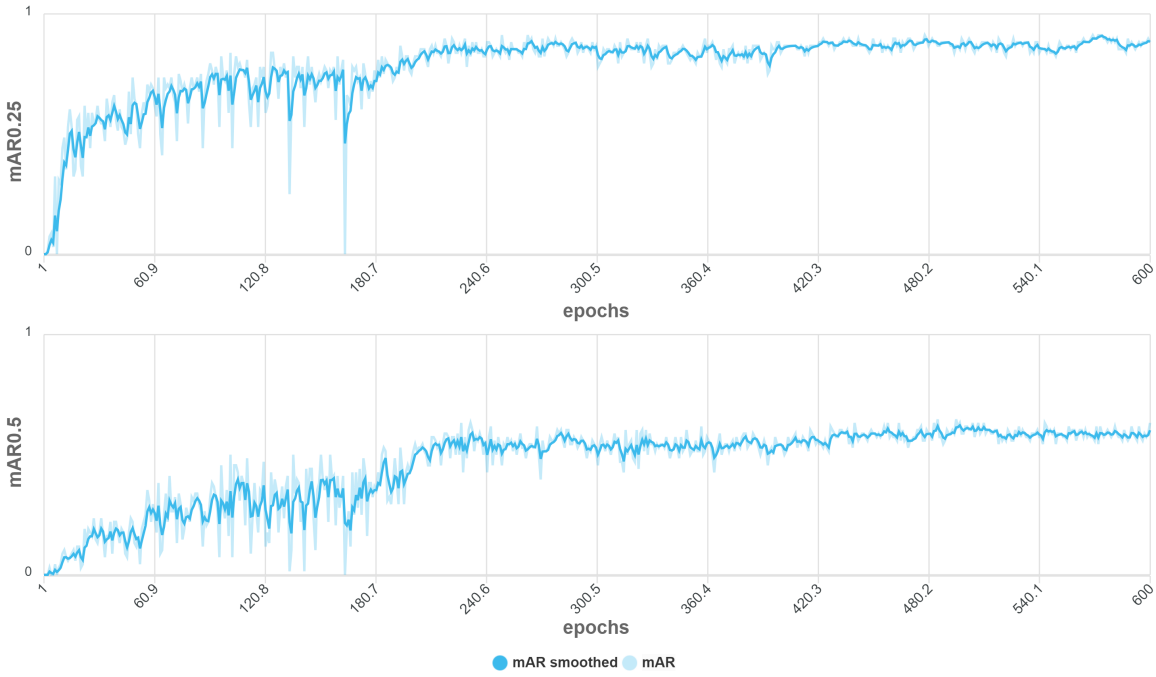


Figura 4.9: Representação gráfica das métricas mAR0.25 e mAR0.50 durante o processo de treino do segundo modelo

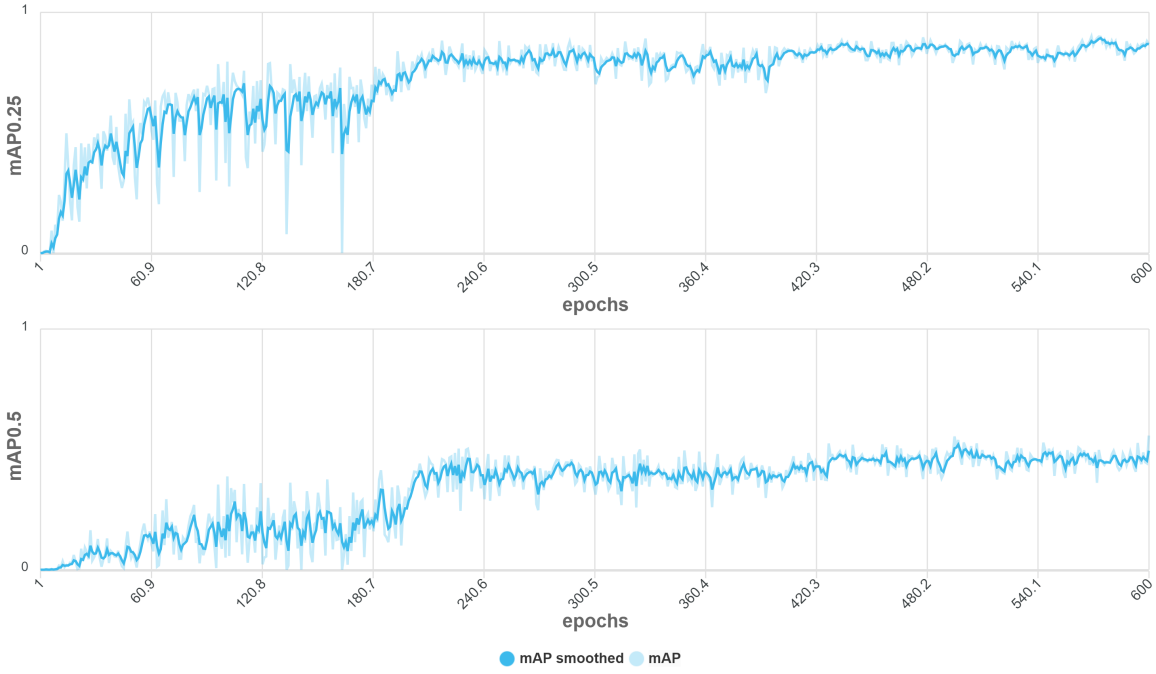


Figura 4.10: Representação gráfica das métricas mAP0.25 e mAP0.50 durante o processo de treino do segundo modelo

Dataset composto por *point clouds* derivadas de 10 *scans* LiDAR

Finalmente, no terceiro e último treino, é examinado um cenário bastante análogo ao segundo modelo, no que concerne a todas as métricas apresentadas nos gráficos representados nas Figuras 4.11, 4.12 e 4.13. Por comparação, é notável uma melhoria nas várias métricas, embora esta discrepância seja de menor magnitude quando comparada com a diferença observada entre o primeiro e o segundo treino.

De forma específica, o melhor *checkpoint* do treino é registado no *epoch* 309, com as seguintes métricas: uma mAR0.25 de 0.947368, uma mAR0.5 de 0.640351, uma mAP0.25 de 0.927441 e uma mAP0.5 de 0.513563.

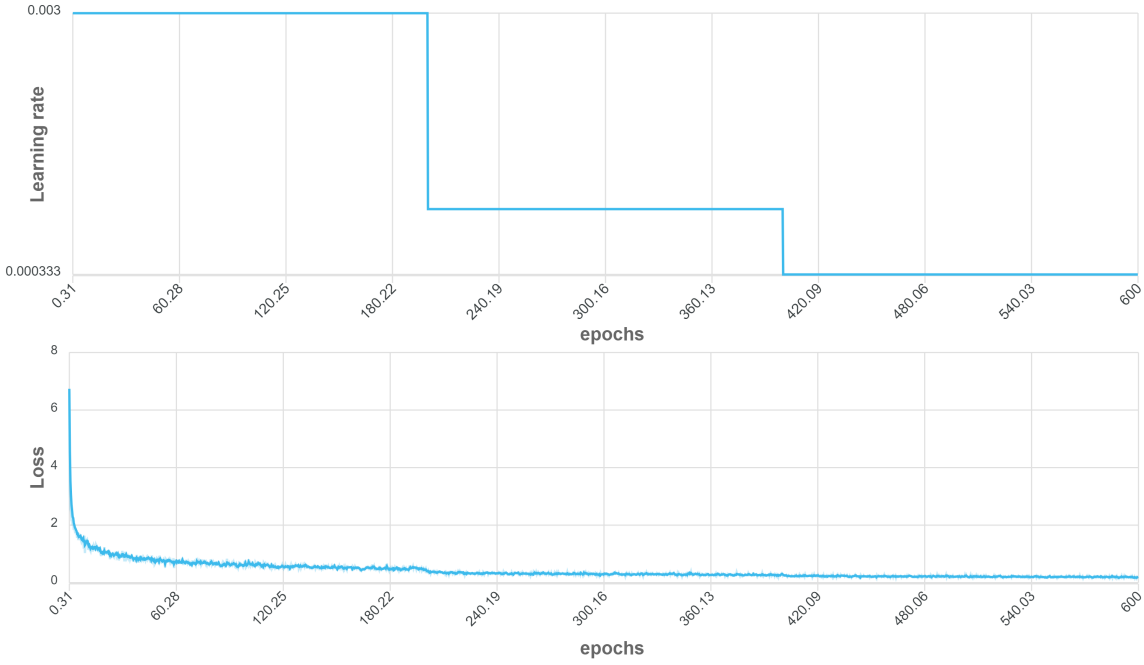


Figura 4.11: Gráfico ilustrativo da lr e das *losses* ao longo do processo de treino do terceiro modelo

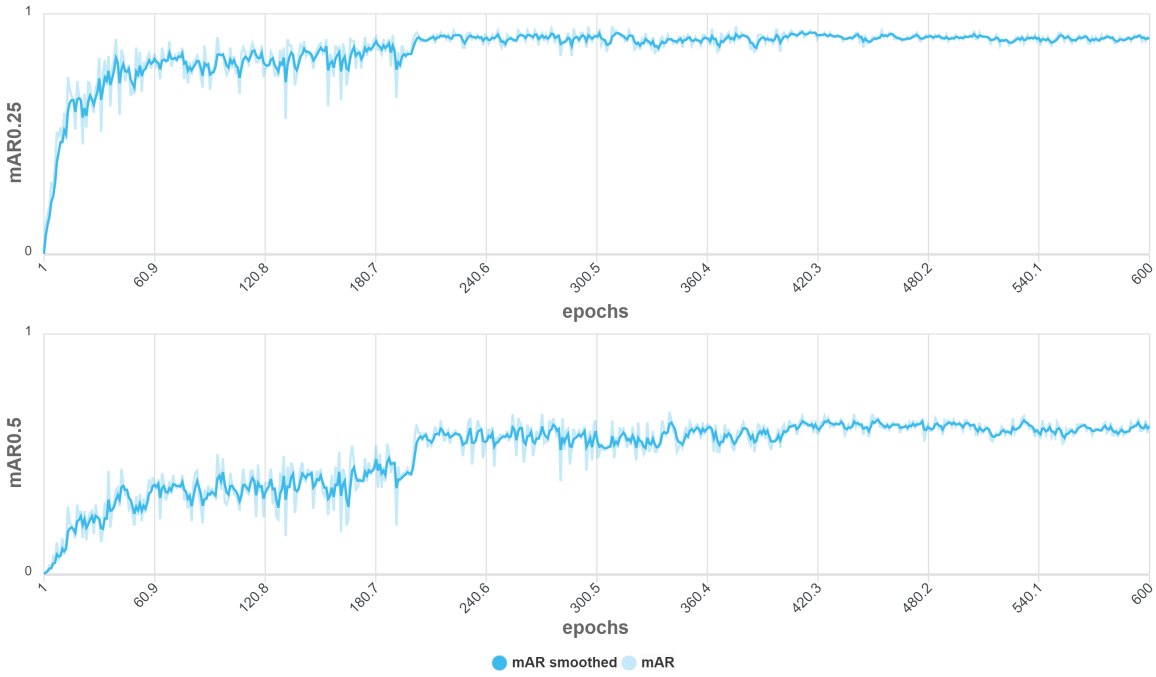


Figura 4.12: Representação gráfica das métricas mAR0.25 e mAR0.50 durante o processo de treino do terceiro modelo

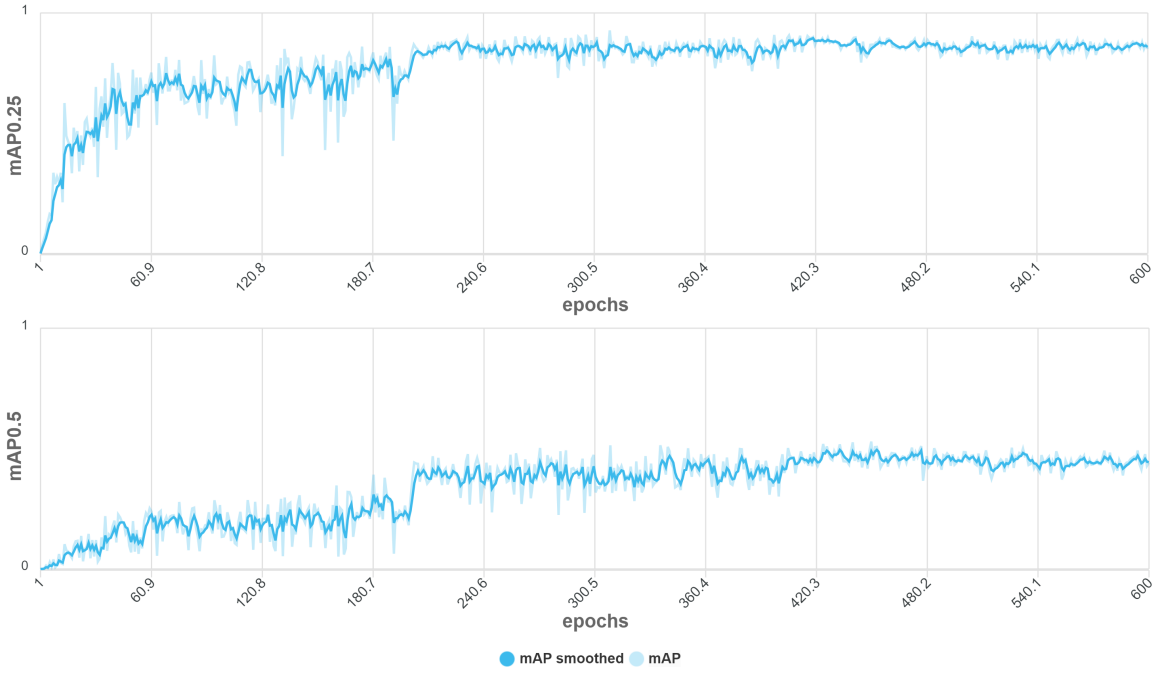


Figura 4.13: Representação gráfica das métricas mAP0.25 e mAP0.50 durante o processo de treino do terceiro modelo

Como evidenciado na Tabela 4.2, é notório que à medida que se aumenta o número de *scans* LiDAR utilizadas na construção da nuvens de pontos, o desempenho do sistema melhora substancialmente. Esta melhoria é particularmente significativa no que diz respeito à precisão na previsão da orientação e localização das instâncias de linhas.

No entanto, a utilização de múltiplos *scans* LiDAR acarreta benefícios que não podem ser quantificados por métricas. Uma vantagem reside no facto de que uma *point cloud* com múltiplos *scans* tem maior probabilidade de conter informações sobre um número superior de instâncias de linhas do que uma *point cloud* com apenas um ou poucos *scans*.

Numa outra perspetiva, as instâncias de linhas numa *point cloud* com múltiplos *scans* contêm um número de pontos substancialmente maior, o que permite que o sistema possa estimar viavelmente a orientação das linhas. Em contraste, numa nuvem de pontos resultante de apenas uma *scan* , é frequente que essas instâncias contenham apenas entre 1 a 3 pontos, o que inviabiliza a estimativa de orientação.

Naturalmente, a utilização de um maior número de *scans* acarreta as suas desvantagens, que neste contexto se manifestam sobretudo no aumento do tempo de pré-processamento, de inferência e de pós-processamento, dado o maior número de pontos envolvidos.

Tabela 4.2: Métricas dos melhores *checkpoints* de cada modelo treinado.

Datasets	**mAR0.25**	**mAR0.50**	**mAP0.25**	**mAP0.50**
1 *scans* LiDAR	0.807692	0.307692	0.736296	0.180897
4 *scans* LiDAR	0.911765	0.573529	0.899989	0.457373
10 *scans* LiDAR	0.947368	0.640351	0.927441	0.513563

4.3 Pós-processamento

O pós-processamento desempenha um papel crucial na obtenção de resultados refinados e fiáveis, tornando-se uma etapa essencial no fluxo de trabalho da análise de dados e aplicações de redes neurais. A sua aplicação adequada pode aumentar significativamente a utilidade e a precisão dos dados obtidos, contribuindo assim para uma tomada de decisão mais informada e eficaz. De acordo com o exposto no capítulo 3, este processo engloba três fases distintas: *Eliminate/Merge predictions*, *Refine predictions* e *Estimate the trajectory of the lines*.

A Figura 4.14 ilustra o fluxograma que descreve o processo lógico da implementação das duas primeiras subetapas: *Eliminate/Merge predictions*, *Refine predictions*

4.3.1 *Eliminate/Merge predictions*

A etapa inaugural, destacada a azul na Figura 4.14, visa eliminar e conjugar previsões, como o nome sugere. Numa primeira fase, procede-se à filtragem de todas as previsões com base no seu *score* associado, ou seja, aquelas que apresentarem um *score* inferior ao limite estabelecido serão descartadas. Deste modo, grande parte das más previsões feitas pela rede são excluídas.

Em sequência, desencadeia-se um procedimento que visa combinar previsões referentes à mesma instância de linhas. Com esse propósito, as previsões que retratam a mesma instância de linhas são determinadas com base na sobreposição das *bounding boxes* e pela semelhança de seus ângulos.

Tendo em conta esta premissa, o primeiro passo é identificar pares de *bounding boxes* que se sobrepõem. Seguidamente, são eliminados todos os pares cujos ângulos não são semelhantes numa perspetiva geométrica.

Através destas etapas, são estabelecidos os diversos pares que representam a mesma instância. Contudo, é possível que uma mesma instância esteja associada a mais do que duas previsões. Então, todos os pares com elementos em comum são consolidados num só grupo. Posteriormente à reorganização dos pares em grupos, as previsões de cada grupo são sintetizadas numa única previsão. Em específico, as *bounding boxes* são combinadas com base nos respetivos *scores*, deste modo, as previsões com pontuações mais elevadas exercerão um maior impacto na formação da caixa delimitadora final.

Finalmente, após a reconstrução de uma nova previsão proveniente de um grupo, procede-se à eliminação das previsões de origem.

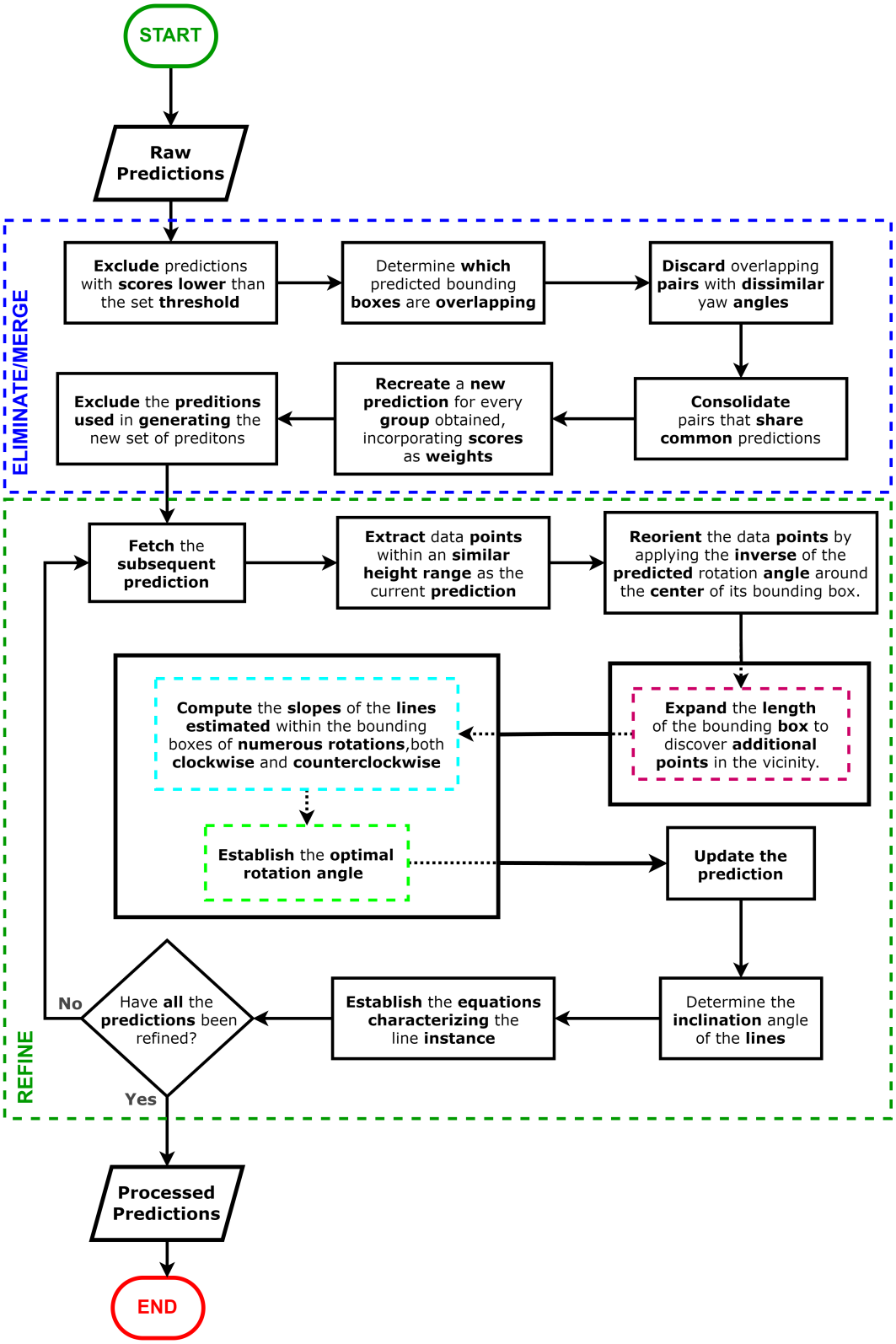


Figura 4.14: Diagrama de fluxo que ilustra a sequência lógica da implementação das duas primeiras subetapas do pós-processamento: *Eliminate/Merge predictions*, *Refine predictions*

4.3.2 *Refine predictions*

A etapa subsequente, realçada a verde na Figura 4.14, procura otimizar as previsões derivadas da etapa anterior.

O processo de refinamento decorre em dois níveis. Primeiramente, busca-se ampliar a extensão da caixa delimitadora, de modo a abranger um maior número de pontos da instância. Em seguida, é efetuado um ajuste do ângulo da caixa para que se alinhe adequadamente com a trajetória das linhas.

Como ponto de partida, são extraídos todos os pontos que se encontram numa faixa de altura semelhante à previsão em questão. De forma sucessiva, estes pontos são reorientados mediante a aplicação do ângulo inverso da previsão, em torno do centro da respetiva *bounding box*. A execução deste procedimento simplifica tanto a extração dos pontos dentro da caixa delimitadora quanto os processos de aperfeiçoamento. Considerando que, ao realizar a rotação inversa nos pontos, a orientação da caixa delimitadora pode ser efetivamente anulada, e, desta forma, esta pode ser representada apenas através de uma delimitação nos três eixos, sem necessidade de considerar ângulos.

Por conseguinte, dá-se início à primeira fase de refinação, incumbida de expandir o comprimento da *bounding box*. O procedimento em questão, representado na Figura 4.15, tem como conceito base o gradual incremento do comprimento da caixa, seguido da análise do número de pontos registados em cada incremento, a fim de determinar o comprimento ideal.

Conforme esquematizado na Figura 4.15, a expansão gradual da *bounding box* não se limita apenas ao aumento do seu comprimento. Em vez disso, esse processo é dividido em duas sequências: o incremento da *front-side* e do *back-side* da caixa. No caso, esta justificação é fundamentada no facto de que esta abordagem permite concentrar a análise em um lado de cada vez, viabilizando a expansão de um lado sem comprometer o outro, o que confere eficácia a este método.

Adicionalmente, a largura da caixa é aumentada para expandir a área de pesquisa, de modo que o algoritmo considera pontos localizados além da trajetória prevista. Desta forma, em cenários nos quais o ângulo previsto encontra-se desfasado em relação à trajetória das linhas, o algoritmo consegue ser eficaz.

Prosseguindo, o processo de expansão mencionado é iterado até alcançar o limite de expansão pré-definido. Sendo que, após cada ciclo de expansão, o número de pontos é registado numa lista específica.

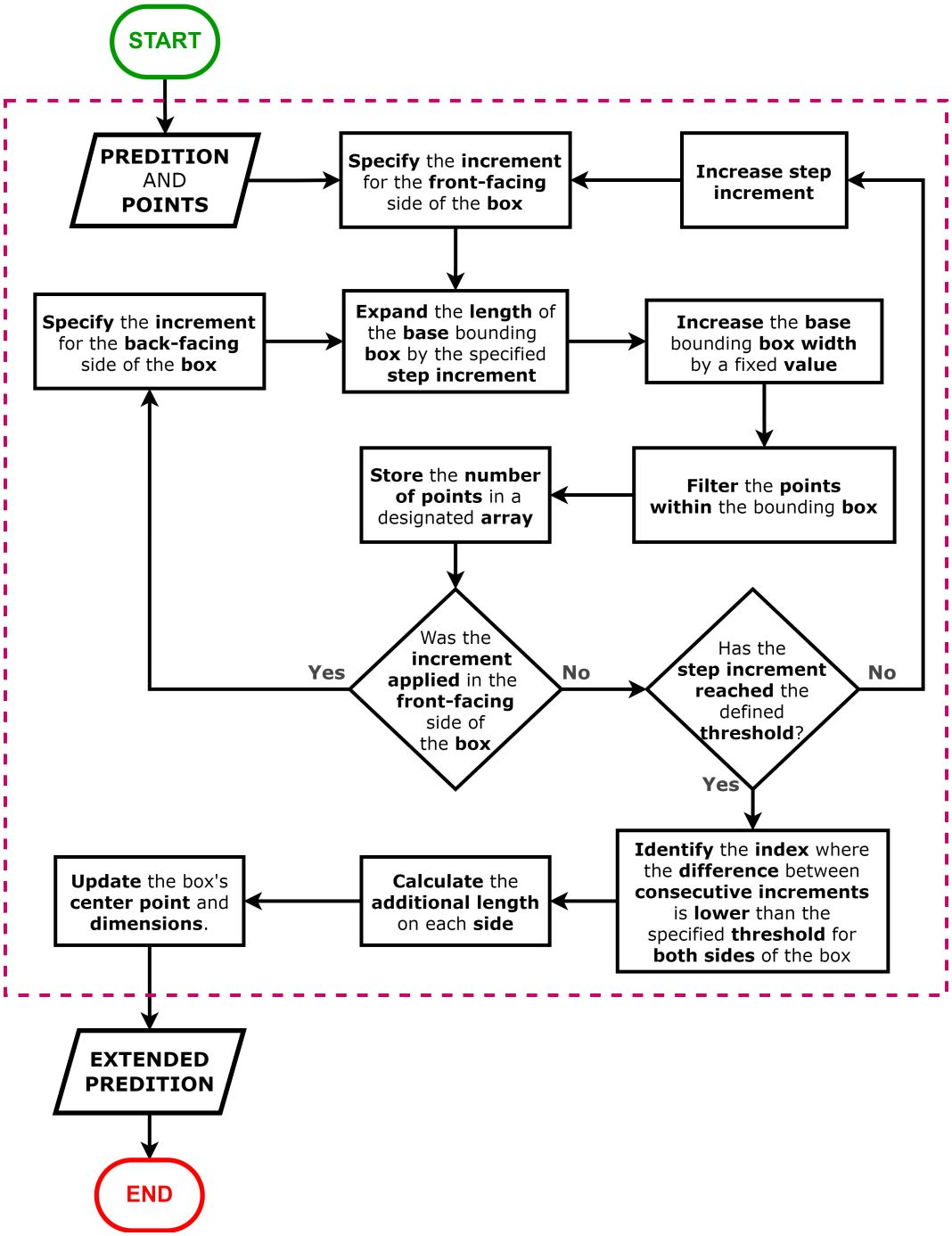


Figura 4.15: Esquema gráfico representativo da primeira camada de refinamento da etapa *Refine predictions*

Subsequentemente, com ambas as listas completas, estas são analisadas a fim de determinar o último índice em que a diferença entre os incrementos consecutivos é inferior ao *threshold* estabelecido. A premissa subjacente a esta análise fundamenta-se na determinação do último incremento em que foram identificados alguns pontos, sem ultrapassar o limite.

Nesta perspetiva, caso o incremento contenha um número significativamente elevado de pontos, considera-se que a extensão engloba uma estrutura atípica de uma instância com linhas, e, como resultado, é considerada inválida. Um valor de referência para este *threshold* é de 3 pontos, um *step* de 0.10 metros e um incremento máximo de 5 metros. Então, através dos índices de ambas as listas e do *step* escolhido, é possível calcular o acréscimo estipulado para ambas as extremidades.

Na Listagem 4.2, está representada a função responsável por calcular o incremento para ambas as extremidades.

```
def inc_boxes(box,boxd,angle, max_len=5, step = 0.1):
    neg_nlines = []
    pos_nlines =  []

    last = 1000
    for i in range(1,int(max_len/step)):
        nboxd = [boxd[0]-1,boxd[1]+1,
                 boxd[2],boxd[3]+i*step,
                 boxd[4]-1,boxd[5]+1]

        linesbox = filter_box(box,nboxd)
        pos_nlines.append(len(linesbox))

        if ( pos_nlines[-1] - last) > 5:
            break
        last = pos_nlines[-1]

    last = 1000
    for i in range(1,int(max_len/step)):
        nboxd = [boxd[0]-1,boxd[1]+1,
                 boxd[2]-i*step,boxd[3],
                 boxd[4]-1,boxd[5]+1]

        linesbox = filter_box(box,nboxd)
        neg_nlines.append(len(linesbox))

        if ( neg_nlines[-1] - last) > 5:
            break
        last = neg_nlines[-1]

    if angle <0:
```

```
            return [(last_true_index([0 < x < 3 for x in np.
                diff(neg_nlines)]) + 2)*step,(last_true_index
                ([0 < x < 3 for x in np.diff(pos_nlines)]) +
                2)*step]
        else:
            return [-(last_true_index([0 < x < 3 for x in np.
                diff(neg_nlines)]) + 2)*step,-(last_true_index
                ([0 < x < 3 for x in np.diff(pos_nlines)]) +
                2)*step]
```

Listagem 4.2: Python code snippet da função encarregada de calcular o acréscimo estipulado em ambas as extremidades

Por último, considerando que este aumento não é uniformemente distribuído em ambas as extremidades, é imperativo redefinir não apenas as dimensões da caixa, mas também o seu ponto central.

Na fase subsequente de refinamento, como referido, é atribuída a tarefa de efetuar o ajuste do ângulo da caixa, com o objetivo de garantir que esta se encontre devidamente alinhada com a trajetória das linhas.

A ideia central deste processo, representado na Figura 4.16, consiste em calcular os declives das linhas para várias rotações, com o propósito de posteriormente, com base na lista de declives obtida, encontrar a rotação ideal. Como tal, este processo encontra-se dividido em duas etapas: o cálculo dos declives das linhas para várias rotações e a determinação da rotação ideal.

Na região destacada a azul na Figura 4.16, é possível observar a sequência lógica da primeira etapa, na qual se evidencia de imediato que, tal como nas etapas primordiais do processo de *Refine predictions*, a rotação é aplicada inversamente nos pontos em vez de ser aplicada à *bounding box*. Além disso, importa salientar que este processo não se limita a uma única direção, uma vez que, para cada incremento angular, este é aplicado sequencialmente em ambas as direções, *clockwise* e *counterclockwise*.

No que concerne ao *loop* de cálculo de declive, o procedimento, como referido, inicia-se pela reorientação dos pontos com base no ângulo de incremento e sentido estabelecidos. Na sequência, procede-se à filtragem dos pontos contidos na caixa delimitadora, seguida da discriminação dos pontos correspondentes a cada linha individual.

Esta atribuição é alcançada pela subdivisão do intervalo do eixo x dos pontos filtrados. Concretamente, num cenário em que a caixa delimitadora encontra-se alinhada com a trajetória das linhas, todos os pontos pertencentes a uma linha individual apresentam baixa variação no eixo x. Neste contexto, é viável proceder à separação dos pontos pela divisão da caixa delimitadora em três secções, considerando que a *bounding box* deve estar praticamente centralizada na instância e sua

largura deve ser próxima da realidade da instância, conforme ilustrado na Figura 4.17.

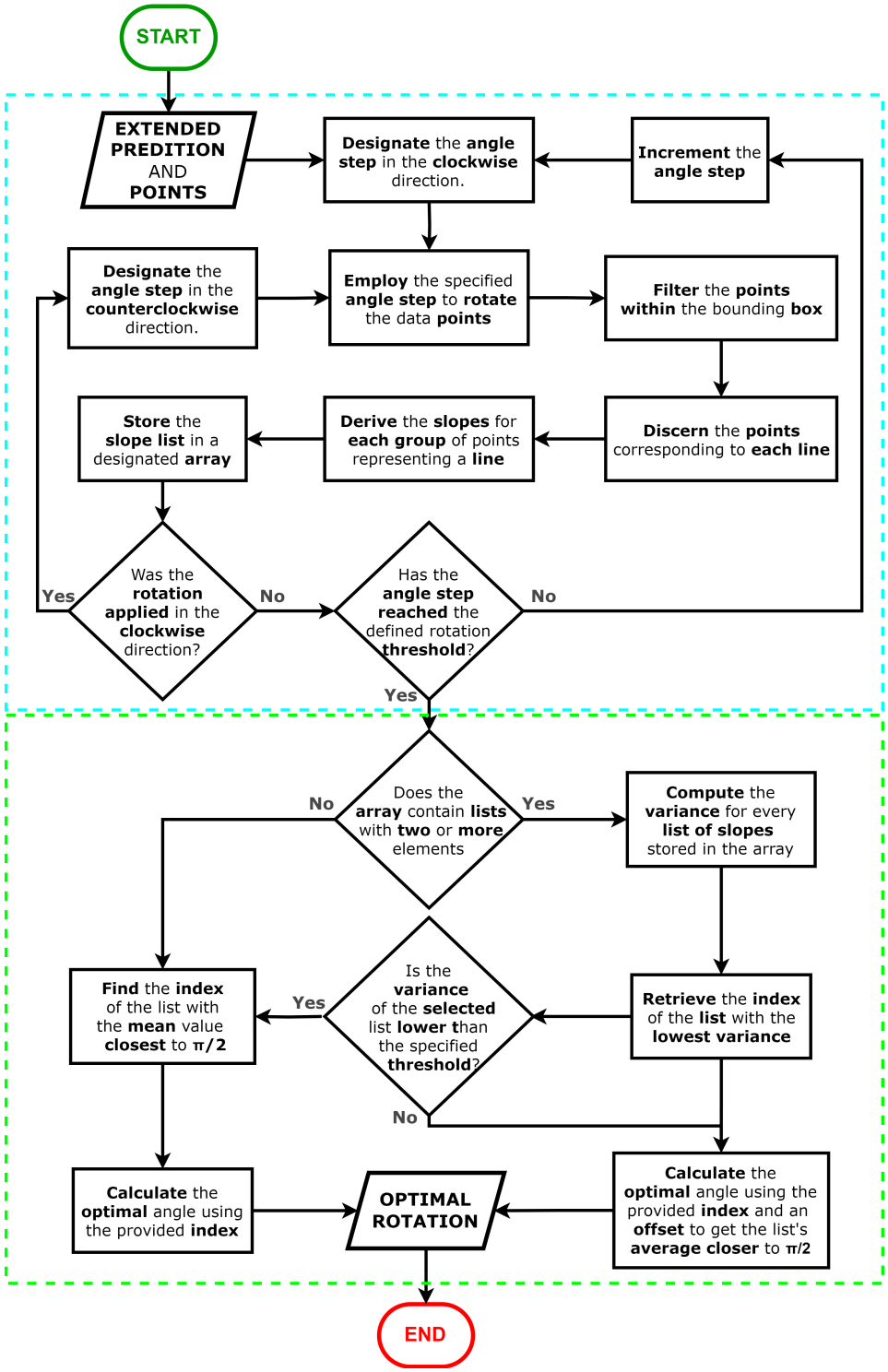


Figura 4.16: Esquema gráfico representativo da segunda camada de refinamento da etapa *Refine predictions*

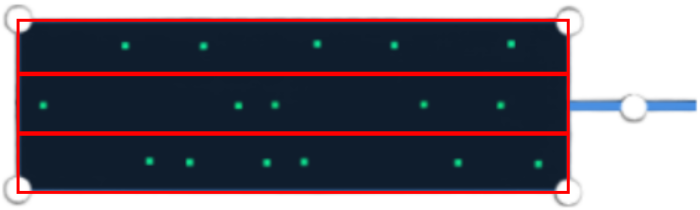


Figura 4.17: Ilustração da subdivisão da caixa delimitadora em três secções

Num estágio posterior, estima-se o declive gerado por cada conjunto de pontos, o qual é determinado através da divisão da covariância de x e y pela variância de x. Este cálculo fornece o declive da linha de regressão linear, que é depois convertido num ângulo angular responsável por avaliar a qualidade do incremento angular de origem. Este cálculo fornece o declive da linha de regressão linear, que é depois convertido num ângulo angular e armazenado em um *array*, em conjunto com os restantes ângulos de declive de cada grupo. Um aspeto crucial neste processo reside no facto de que o ângulo ideal formado pelos pontos é de 90º. Como consequência, o declive da linha correspondente é infinito, o que pode tornar o cálculo inconsistente quando tratam de ângulos próximos ao ideal. Para solucionar esta questão, procedeu-se à permuta dos valores de x e y, com o intuito de estabelecer o ângulo ideal das linhas formadas como 0 rad, o que possibilitou um cálculo mais intuitivo. Por fim, para compensar esta adaptação, foi adicionado um acréscimo de $\frac{\pi}{2}$ rad ao ângulo estimado. Em suma, a cada iteração, é registada uma lista que contém as várias inclinações de cada grupo de pontos numa *array*, na qual a qualidade do incremento será avaliada.

No desfecho do ciclo, no final de uma iteração para ambos os sentidos, o ângulo de incremento é progressivamente aumentado até que o limite estabelecido seja alcançado

Na segunda etapa, realçada em verde na Figura 4.16, o procedimento lógico é bifurcado em duas abordagens distintas, ambas com o propósito de determinar o ângulo ideal. A primeira abordagem, considerada a mais favorecida, consiste em selecionar o índice do *step* no qual se verifica a maior congruência entre os ângulos das retas de regressão linear de cada grupo, registados numa iteração. Isso é justificado pelo facto de que, caso os ângulos das diversas linhas sejam semelhantes, fica explícito que a divisão da instância em grupos de pontos está adequadamente efetuada. Nesta vertente, constata-se que cada agrupamento de pontos corresponde corretamente a uma linha individual, o que possibilita o cálculo de um ajuste viável com base nos ângulos registados.

De forma específica, o índice é determinado com base no cálculo da variância de cada lista de declives, sendo escolhida aquela que apresenta a menor variância. Se a lista selecionada apresentar uma variância inferior ao valor limite instituído, então o ângulo ideal será determinado com base no índice da lista, no tamanho *step* e na discrepância entre a média dos ângulos da lista e o ângulo ideal, que neste contexto é consolidado em $\frac{\pi}{2}$ rad. Caso contrario o processo de seleção é submetido à outra abordagem.

Na abordagem secundária, o índice é estabelecido pela seleção da lista cuja média dos ângulos das linhas se encontre mais próxima do ângulo de referência. Esta abordagem é adotada quando a primeira estratégia se revela ineficaz ou impraticável. Tal cenário pode ocorrer quando nenhuma lista do conjunto de ângulos possui pelo menos dois elementos, o que impossibilita o cálculo da variância. Além disso, conforme referido, a opção também pode ser tomada quando a variância da lista selecionada excede o limite definido. Nesta ótica, o ângulo ideal é estimado mediante a exclusiva consideração do índice da lista e a dimensão base do *step*.

Na Listagem 4.3, estão representadas as funções encarregadas de encontrar o incremento angular ótimo para o calculo do ângulo ideal.

```
def find_best_fit(arrays):
    if any(len(arr) > 2 for arr in arrays):
        arr,idx = find_array_with_min_variance(arrays)
        if max(abs(np.diff(arr))) < 0.1:
            return arr,idx, True
        else:
            return find_closest_single_array(arrays,np.pi
                /2)

    else:
        return find_closest_single_array(arrays,np.pi/2)

def find_closest_single_array(arrays, value):
    closest_array = None
    closest_index = -1
    min_diff= float('inf')

    for index, array in enumerate(arrays):
        if len(array) == 1:
            diff = abs(array[0] - value)
            if diff < min_diff:
                min_diff = diff
                closest_array = array
                closest_index = index
```

```
        return closest_array, closest_index,False

    def find_array_with_min_variance(arrays):

        variances = [np.var(array) if len(array) > 1 else 10
            for array in arrays]
        min_variance_index = np.argmin(variances)
        return arrays[min_variance_index],min_variance_index
```

Listagem 4.3: Python code snippet das funções utilizadas para determinar o incremento angular ótimo para o calculo do ângulo ideal

Os parâmetros de referência definidos para este procedimento consistem em um *step* angular de 0.10 rad, com um limite máximo de incremento de $\frac{\pi}{8}$.

Após a conclusão do último processo de refinação, a última ação a ser realizada é a caracterização da instância das linhas, conforme ilustrado na Figura 4.14. Esta caracterização é representada por três equações, uma para cada um dos eixos que descrevem a linha central da instância. Fundamentalmente, estas equações são concebidas a partir dos ângulos de *yaw* (ψ) e *pitch* (θ), assim como o ponto central inicial da instância (x_0, y_0, z_0).

Portanto, o ângulo *yaw* corresponde ao ângulo da caixa delimitadora, ao passo que o ângulo *pitch* é determinado através da análise do plano YZ da caixa delimitadora. Concretamente, o ângulo de *pitch* é obtido mediante o cálculo da média dos *slopes* individuais de cada linha no plano YZ da *bounding box*.

Assim, a partir destes elementos, é factível descrever a localização da instância das linhas, em função do comprimento (l) desta, conforme é demonstrado no grupo de Equações 4.1, 4.2 e 4.3.

$$x(l) = x_0 + l \cdot \sin(\psi) \cdot \cos(\theta), \tag{4.1}$$

$$y(l) = y_0 + l \cdot \cos(\psi) \cdot \cos(\theta), \tag{4.2}$$

$$z(l) = z_0 + l \cdot \sin(\theta) \tag{4.3}$$

Capítulo 5

Resultados

Posterior à implementação completa do sistema, procede-se à última etapa, que consiste no registo e análise do desempenho do modelo desenvolvido para abordar o problema central do projeto em questão, É, portanto, imperativo validar o desempenho do algoritmo em vários cenários, reforçando, assim, a confiabilidade do sistema.

Neste sentido, a totalidade do *dataset* abordado no subcapítulo 4.2.1 é alimentado ao sistema desenvolvido, com o intuito de avaliar os efeitos do pré-processamento e pós-processamento, bem como a eficiência e eficácia dos três modelos treinados.

Deste modo, neste capítulo é analisada a performance das etapas de processamento e os resultados obtidos pelos três sistemas com um conjunto de dados reais.

5.1 Pré-processamento

O principal propósito deste procedimento reside na eliminação de ruídos, na redução do volume de dados e na sua devida preparação, com o intuito de garantir que o modelo possa aprender e generalizar com eficácia.

Neste sentido, a maior parte do impacto do pré-processamento ocorre na fase de treino. No entanto, é imperativo que o pré-processamento seja consistentemente aplicado para assegurar a eficácia da inferência do modelo. Consequentemente, as configurações de pré-processamento estipuladas são idênticas às definidas para o treino, nos quais foram utilizados os valores de referência mencionados ao longo do subcapítulo 4.1.

Relativamente ao repercussão concreta do pré-processamento, a Figura 5.1 ilustra o resultado característico da transformação de uma *point cloud* derivada de 10 *scans* LiDAR, para um *voxel size* de 12 unidades.

Na *raw point cloud* da primeira secção da Figura 5.1, é evidente que o solo do cenário apresenta uma densidade elevada de pontos, sendo também observável algum ruído proveniente do LiDAR.

Conforme é esperado, o pré-processamento atua nestes aspetos, como pode ser observado na *processed point cloud* ilustrada na Figura 5.1. O solo, cuja densidade de pontos era inicialmente elevada e constituía uma parte considerável do conjunto total de pontos, foi subsequentemente e eficazmente reduzido a um número limitado de pontos, e o ruído previamente presente foi completamente erradicado, sem comprometer a integridade dos pontos que representam as linhas de transmissão.

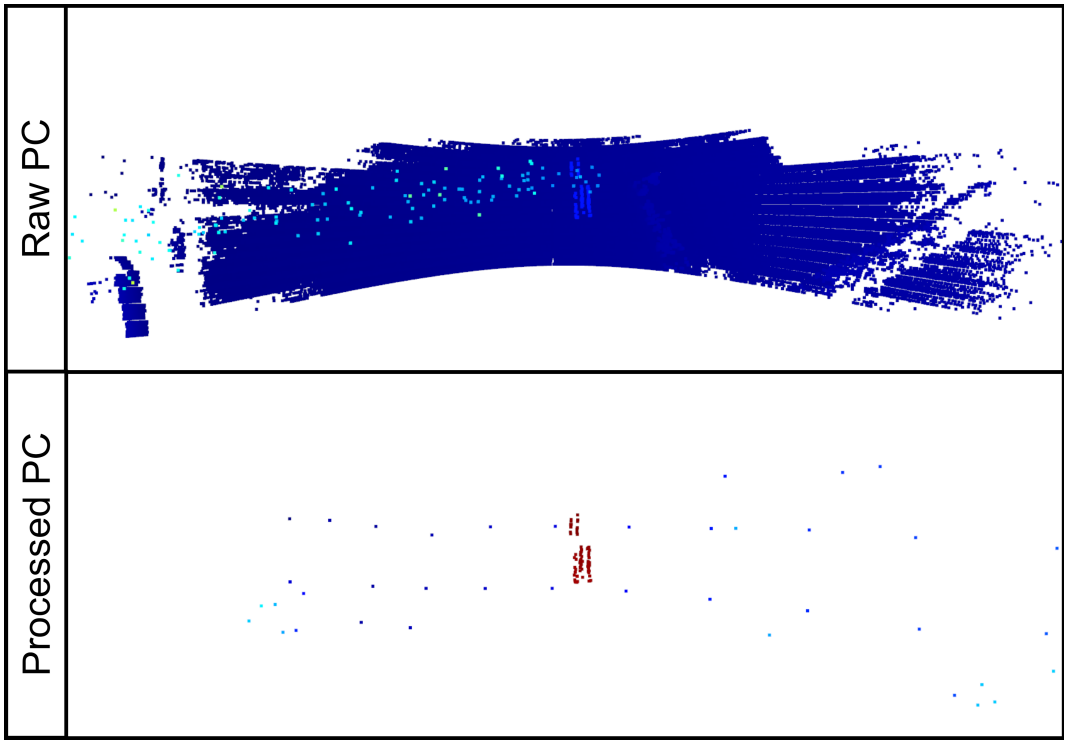


Figura 5.1: Representação visual do efeito do pré-processamento, onde a imagem inferior ilustra o resultado obtido após a aplicação do pré-processamento na *point cloud* da imagem superior

Efetivamente, as nuvens de pontos derivadas de apenas um *scan* possuem, em média, 7652 pontos, os quais são reduzidos para apenas 80 após o processamento. No caso das *point clouds* provenientes de quatro *scans*, a média é de 30608 pontos, que são posteriormente reduzidos para 140. Por último, as nuvens de pontos constituídas por dez *scans* apresentam em média 76515 pontos, dos quais, após o pré-processamento, são transformados em 296 pontos.

5.2 Pós-processamento

O pós-processamento objetiva otimizar o *output* do *object detector*, o que implica a eliminação e fusão preliminar de previsões, seguida por um refinamento direcionado especialmente à sua orientação.

Naturalmente, para um pós-processamento eficaz, como mencionado na secção 4.3, é fundamental selecionar valores de referência apropriados para garantir o correto funcionamento do mesmo. Neste contexto, na etapa de *eliminate/merge predictions*, foram descartadas todas as previsões com um *score* inferior a 0.20.

Na etapa de *refine predictions*, foram utilizados os valores de referência mencionados ao longo do subcapítulo 4.3.2. Isto é, no primeiro nível de refinação, foi aplicado um *threshould* de 3 pontos, um *step* de 0.10 metros e um incremento máximo de 5 metros. No segundo nível, adotou-se um incremento angular de $0,10\ rad$, com um limite máximo de incremento de $\frac{\pi}{8}\ rad$.

As Figuras 5.2 e 5.3, representam dois cenários distintos típicos observados ao longo da execução do sistema.

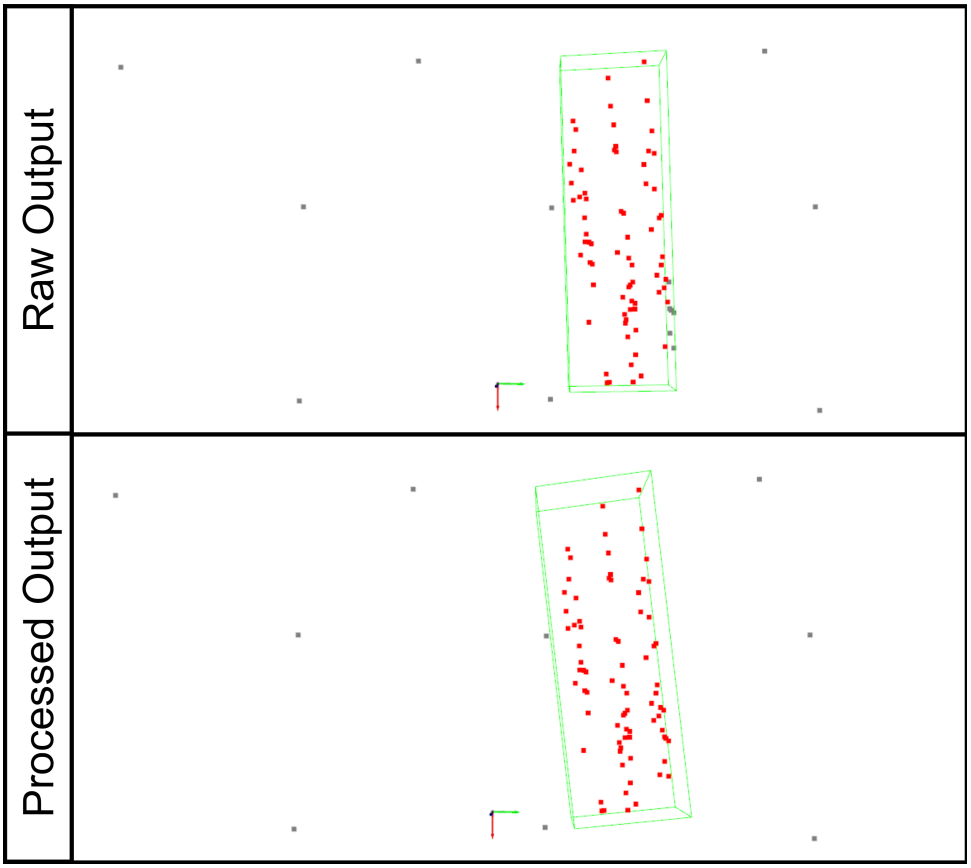


Figura 5.2: Representação visual do efeito do pós-processamento, na qual a imagem inferior ilustra o resultado obtido após a aplicação do pós-processamento ao *output do modelo* representado na imagem superior

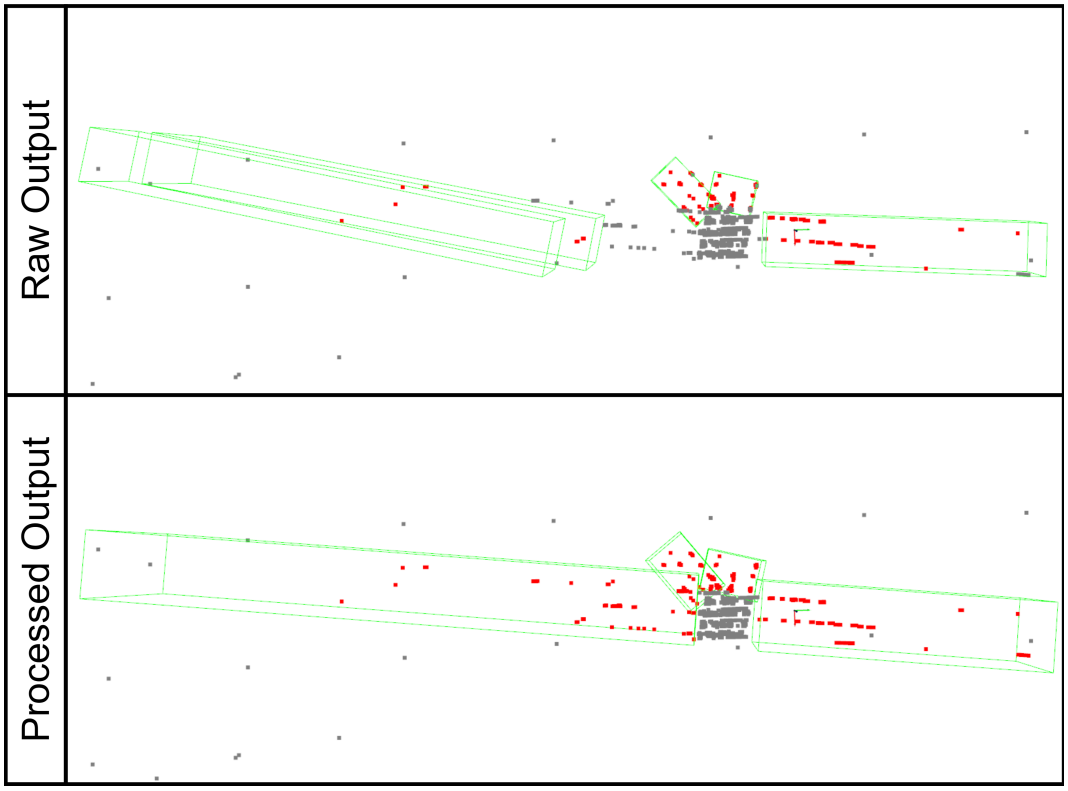


Figura 5.3: Representação visual do efeito do pós-processamento num cenário mais complexo, na qual a imagem inferior ilustra o resultado obtido após a aplicação do pós-processamento ao *output do modelo* representado na imagem superior

No primeiro exemplo, ilustrado na 5.2, aborda a situação mais recorrente no contexto de aplicação, na qual o UAV está a efetuar o acompanhamento da linha de transmissão.

Neste cenário, o output do modelo demonstra eficácia na localização das instâncias de linhas, a previsão da orientação destas é relativamente próxima da realidade, e à medida que a nuvem de pontos incorpora mais *scans*, observa-se uma tendência de maior convergência com a realidade. Nesta perspetiva, o pós-processamento atua eficazmente, em praticamente todas as situações desta natureza, na correção da orientação das previsões, como pode ser observado na Figura 5.2, onde a *bounding box* no *processed output* está em conformidade com a direção das linhas, em contraste com a do *raw output*.

O segundo cenário, retratado na Figura 5.3, é notável pela inclusão de um poste, o que acrescenta um grau significativo de complexidade ao sistema, uma vez que implica a presença de múltiplas instâncias de linhas com orientações distintas. No caso, o sistema enfrenta dificuldades na tarefa de determinar com precisão a orientação e a localização das linhas horizontais do poste, como claramente evidenciado na primeira secção da Figura 5.3.

Na situação destacada é comum que estas previsões do modelo apresentem uma ou várias imperfeições, tais como a deteção de uma única instância de linhas com múltiplas *bounding boxes*, um evidente desfasamento na orientação em relação às linhas e a não abrangência de todos os pontos da instância pela caixa delimitadora.

No caso, como já referido, todas estas imperfeições são aspetos que o pós-processamento procura retificar. Do ponto de vista prático, apesar desta correção nem sempre ser perfeita, na maioria das situações, resulta em uma melhoria substancial. Concretamente, na Figura 5.3, está exemplificado um cenário em que o pós-processamento é eficaz na retificação de todas as imperfeições.

5.3 Resultados Obtidos pelos Diferentes Modelos

Como premissa para estudar o efeito da acumulação de *scans*, conforme exposto no subcapítulo 4.2.3, foram treinados três modelos com base em três conjuntos de dados equivalentes, porém com diferentes níveis de informação. Em particular, foram treinados modelos para a agregação de 1 *scan*, 4 *scans* e 10 *scans*.

Tal como analisado no subcapítulo 4.2.4, com base nas diversas métricas registadas durante o treino, é possível deduzir algumas suposições relativas a cada modelo. É notório que à medida que o número de *scans* LiDAR utilizadas na construção das *point cloud* aumenta, o desempenho do sistema melhora de forma substancial. Esta melhoria revela-se particularmente significativa no que concerne à precisão na previsão da orientação e localização das instâncias de linhas.

É ainda relevante salientar que este aprimoramento não é linear, sendo significativamente menos acentuado na transição de 4 para 10 *scans*, em comparação com a mudança de 1 para 4 *scans*.

Naturalmente, a análise comparativa de métricas é insuficiente para a avaliação concreta das discrepâncias e melhorias de cada sistema.

Deste modo, através dos testes efetuados, foi possível identificar e isolar três cenários distintos, representados nas Figuras 5.4, 5.5 e 5.6, que claramente demonstram as diferenças e limitações de cada modelo.

A Figura 5.4 ilustra o cenário ideal no qual o UAV está a seguir as linhas de transmissão e o LiDAR é capaz de capturar múltiplos pontos das linhas.

Nesta circunstância, com o auxílio do pós-processamento, todos os modelos demonstram a capacidade de prever com precisão a localização e orientação das instâncias de linhas, tal como evidenciado na Figura 5.4

No entanto, noutra situação semelhante, em que o LiDAR, neste caso, é inapto de registar inúmeros pontos, como representado na Figura 5.5, a limitação do primeiro modelo torna-se evidente. Devido à limitada quantidade de informação disponível (apenas dois pontos), o primeiro sistema encontra-se incapacitado de prever a orientação da instância de linhas.

Por outro lado, nos demais modelos, uma vez que as *point cloud* de entrada são compostas por múltiplos *scans*, as instâncias de linhas continuam a ser caracterizadas por vários pontos, o que faculta que ambos os sistemas prevejam com exatidão a sua orientação.

Finalmente, no último exemplo apresentado na Figura 5.6, observa-se uma situação de notável complexidade, na qual se tornam evidentes as limitações do segundo modelo. Neste contexto, torna-se claro que a presença de um maior volume de informação no terceiro modelo, em comparação com o primeiro e o segundo modelos, resulta numa identificação mais precisa de um maior número de instâncias.

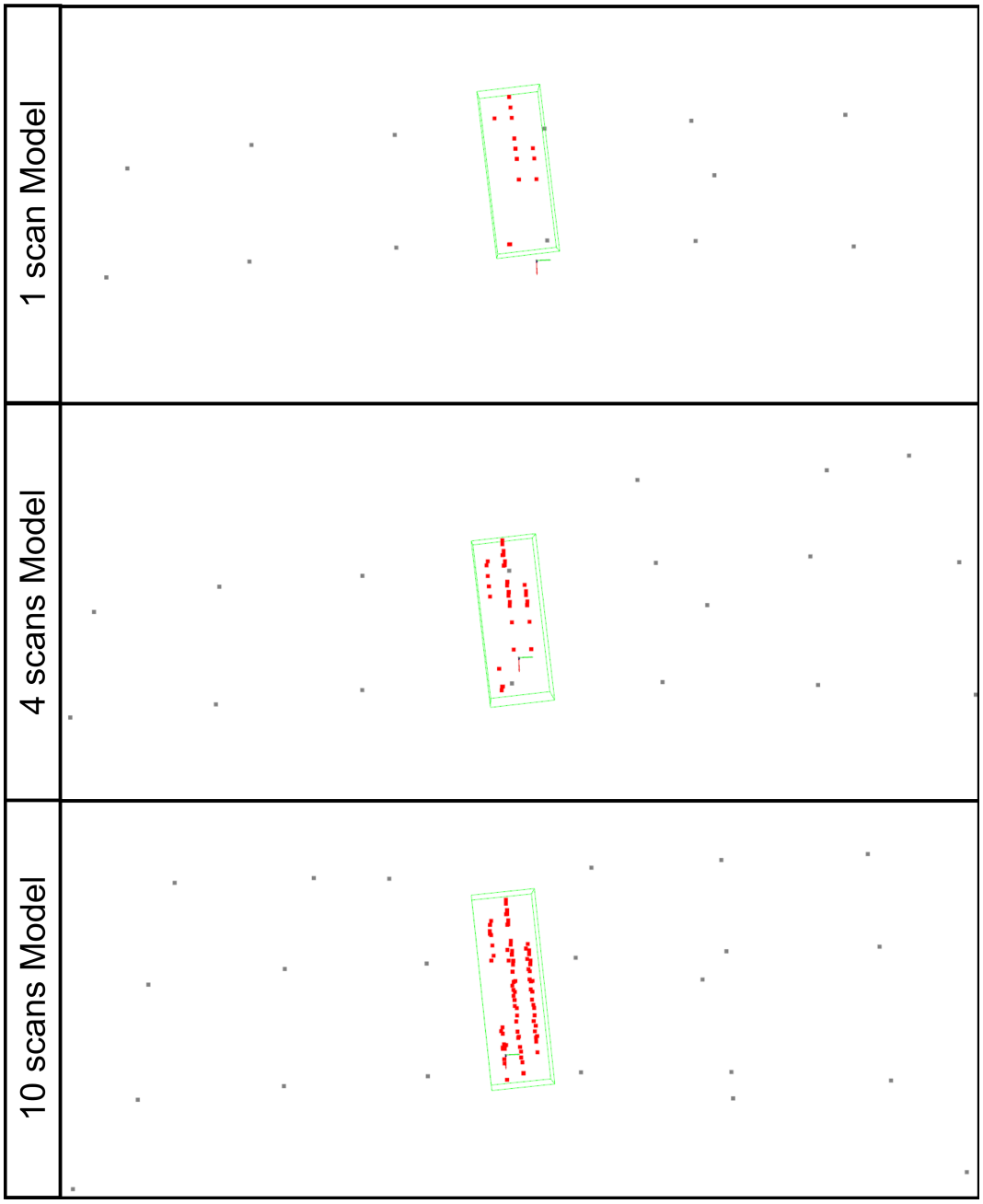


Figura 5.4: Representação visual da comparação dos resultados de cada modelo no primeiro cenário

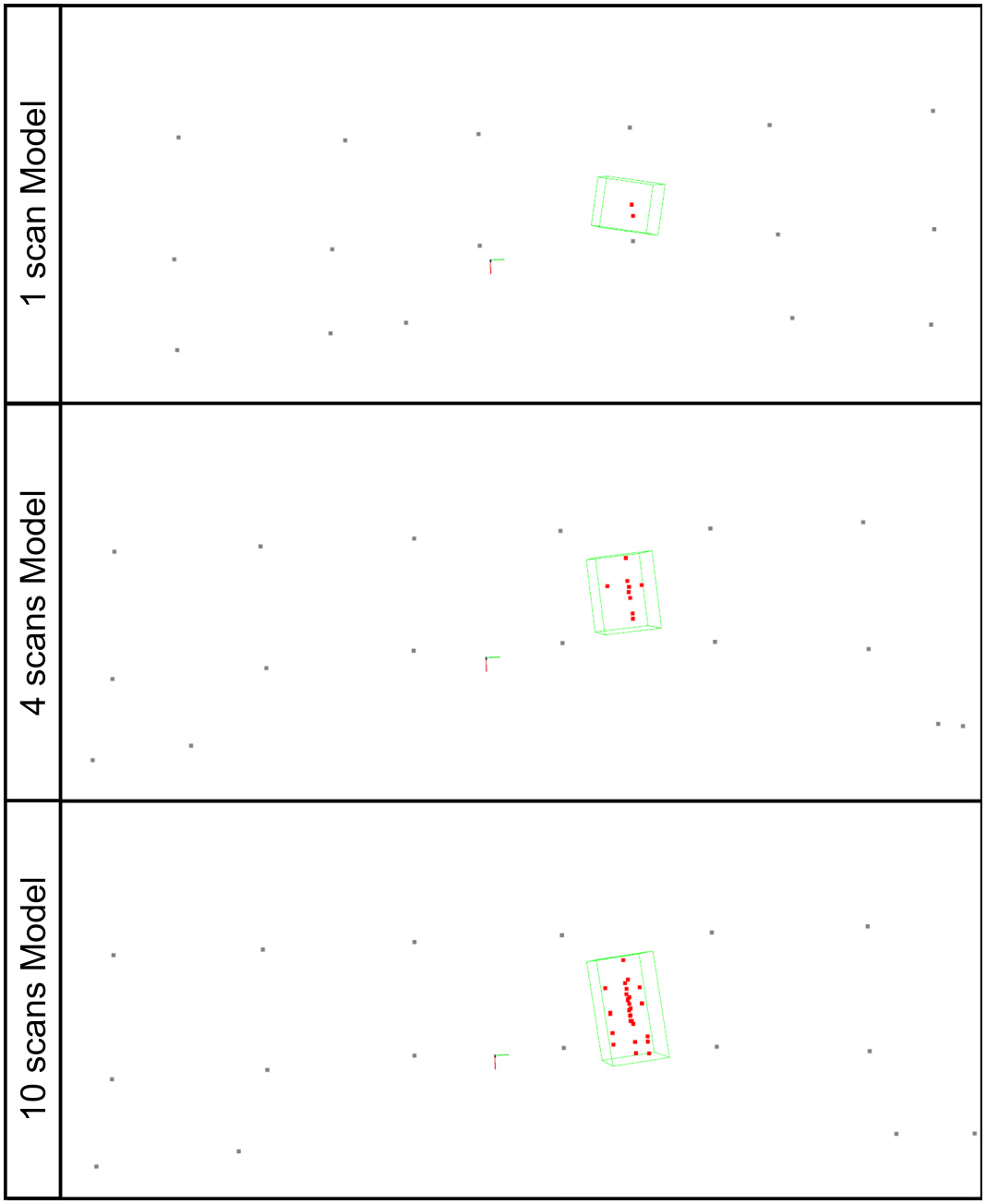


Figura 5.5
Representação visual da comparação dos resultados de cada modelo no segundo cenário

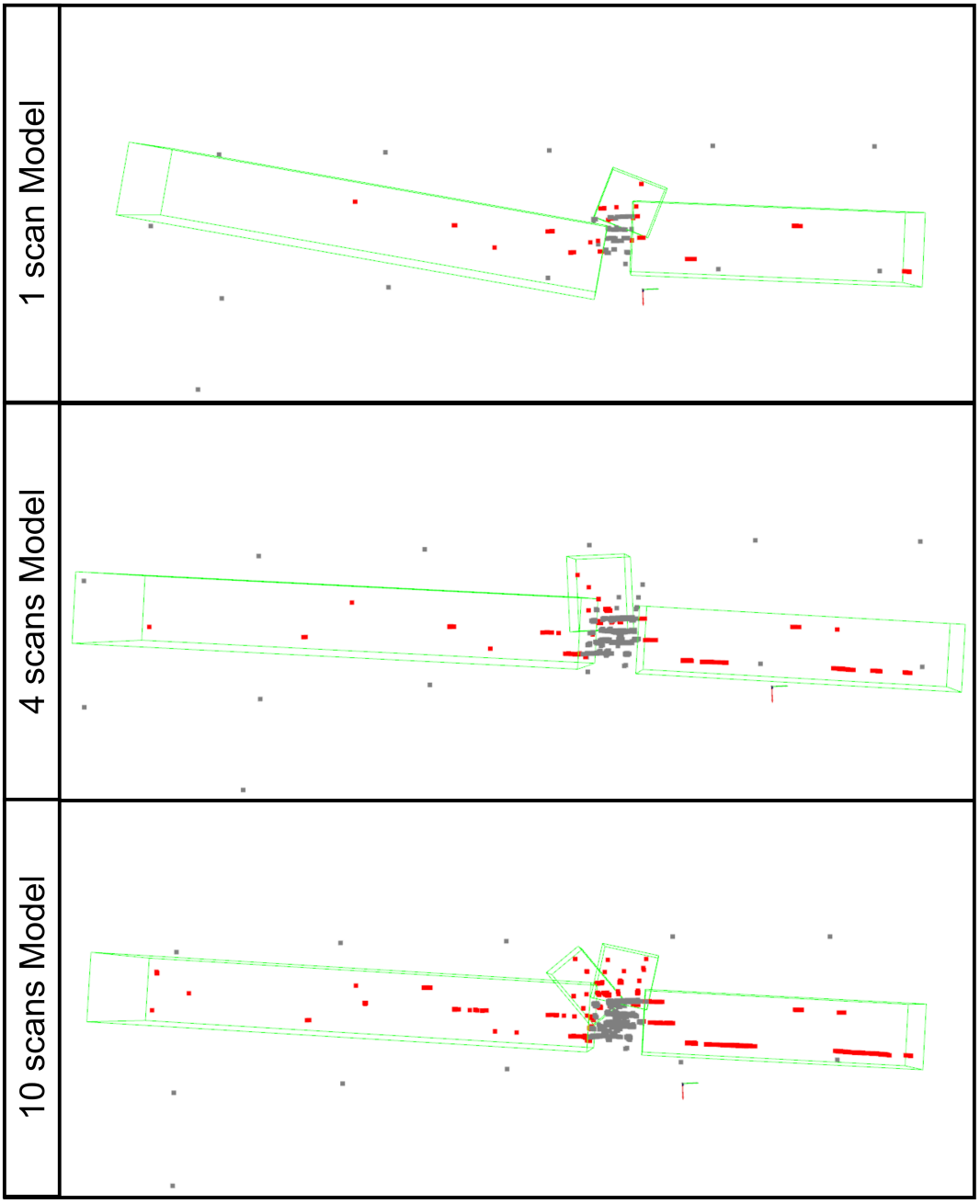


Figura 5.6: Representação visual da comparação dos resultados de cada modelo no terceiro cenário

Naturalmente, a utilização de múltiplos *scans* acarreta desvantagens, as quais, neste contexto, se evidenciam principalmente pelo prolongamento do tempo de pré-processamento, inferência e pós-processamento, devido ao aumento no número de pontos envolvidos.

Neste sentido, foram registados os tempos médios de execução de cada fase do sistema para cada modelo. Os tempos de execução das diferentes fases do sistema

podem ser visualizados na Tabela 5.1, sendo que o hardware utilizado para estes testes consistiu num processador AMD Ryzen 5 5600X e numa placa gráfica NVIDIA GeForce RTX 4070.

Conforme antecipado, o *single scan model* demonstrou ser o mais rápido, seguido pelo *four scans model* e, por último, o *ten scans model*.

Tabela 5.1: Tempos de execução das diversas etapas do sistema de cada modelo

Models	**Preprocessing (ms)**	**Inference (ms)**	**Posprocessing (ms)**	**Total (ms)**
1 scan Model	10.0723	29.1207	6.0800	45.2730
4 scans Model	13.0524	36.7313	7.6960	57.4749
10 scans Model	32.7333	40.9923	11.3564	85.0820

Em conclusão, após a análise dos vários aspetos expostos de cada modelo, é evidente que a escolha *single scan model* carece de justificação quando comparada com as restantes alternativas. Dado que a diferença na eficácia entre este modelo e o *four scans model* é significativa, sem que ocorra um aumento substancial no tempo total de execução do sistema. Em contrapartida, o *ten scans model* é o mais eficaz, no entanto este acarreta um aumento considerável no tempo total de execução do sistema.

Em suma, o *four scans model* surge como a escolha mais clara quando se pretende um sistema de maior eficiência, com tempos de execução reduzidos e um bom rendimento. Neste caso, este modelo demonstra notável aptidão para determinar a localização e orientação das linhas enquanto o UAV está a seguir as linhas de transmissão. No entanto, ele revela as suas limitações num cenário caracterizado pela existência de várias linhas, ou seja, na presença de um poste elétrico.

O *ten scans model* revela-se como a alternativa mais robusta, no qual demonstra uma eficácia notável durante a operação do UAV ao longo das linhas de transmissão. E em cenários caracterizados pela presença de múltiplas linhas, esta opção destaca-se como a mais competente.

Capítulo 6

Conclusão e Trabalho Futuro

A crescente adoção de sistemas de deteção de obstáculos e anticolisão tem-se revelado, nos últimos anos, uma funcionalidade essencial, especialmente no âmbito do desenvolvimento e da pesquisa relacionados com a UAVs. Este facto leva a um aumento da autonomia na navegação dos vários sistemas aéreos que a requerem.

Nesta perspetiva, a utilização de UAVs representa uma das opções mais atrativas para a tarefa da inspeção de linhas de transmissão. O objetivo fundamental deste estudo consiste em aprimorar a capacidade de deteção dos UAVs neste contexto, de forma a viabilizar uma operação mais segura durante a missão de inspeção das linhas de transmissão. Com este propósito, desenvolveu-se um algoritmo com a capacidade de identificar a localização e orientação das linhas de transmissão, através da utilização dos dados recolhidos por um sensor LiDAR.

Em especifico, o algoritmo desenvolvido faz uso de um *deep learning LiDAR-based 3D object detection model* para não só prever a posição, mas também estimar a direção da trajetória das linhas, mediante a localização e orientação das *3D oriented bounding boxes* que o algoritmo prevê.

Com base nos resultados expostos no capítulo 5, é factível concluir que os objetivos centrais desta dissertação foram concretizados.

Todas as fases integrantes do sistema mostram eficácia na sua tarefa designada. Particularmente, o pré-processamento demonstrou ser capaz de eliminar ruídos e reduzir o volume de dados. Os modelos treinados revelaram-se aptos para realizar uma estimativa próxima da localização e orientação das linhas de transmissão. O

pós-processamento, por sua vez, refina estas previsões, o que conduz a um resultado mais preciso.

Ao analisar o desempenho do sistema com os três modelos treinados, torna-se evidente que a escolha do *single scan model* carece de justificação quando comparada às restantes alternativas. Tanto o modelo *four scans model* quanto o modelo *ten scans model* apresentam-se como soluções mais vantajosas, uma vez que cada um deles possui seus respetivos benefícios.

O *four scans model* surge como a escolha mais clara quando se pretende um sistema de maior eficiência, com tempos de execução reduzidos e um bom rendimento. Neste caso, este modelo demonstra notável aptidão para determinar a localização e orientação das linhas enquanto o UAV está a seguir as linhas de transmissão. No entanto, ele revela as suas limitações num cenário caracterizado pela existência de várias linhas, ou seja, na presença de um poste elétrico.

O *ten scans model* revela-se como a alternativa mais robusta, ainda que implique em um aumento no tempo de execução total do sistema. Esta abordagem demonstra uma eficácia notável durante a operação do UAV ao longo das linhas de transmissão. E em cenários caracterizados pela presença de múltiplas linhas, esta opção destaca-se como a mais competente.

Apesar de o conceito do algoritmo ter sido validado com o *dataset* utilizado, é imperativo avaliar o sistema com outros conjuntos de dados com condições distintas.

Neste sentido, a expansão do *dataset* de treino é fundamental para garantir o seu desempenho adequado em uma variedade de cenários com diferentes condições. De forma concreta, exemplos de características que contribuem para o aprimoramento da robustez do sistema incluem a presença de diferentes *backgrounds*, que variam desde várias formas de vegetação até a presença de rochas, bem como a variação dos tipos de linhas de transmissão, que abrangem diferentes tamanhos e estruturas.

Referências

[1] A. Pagnano, M. Höpf, and R. Teti, "A roadmap for automated power line inspection. Maintenance and repair.," *Procedia CIRP*, vol. 12, pp. 234–239, 2013. [Citado na página 1]

[2] L. Matikainen, M. Lehtomäki, E. Ahokas, J. Hyyppä, M. Karjalainen, A. Jaakkola, A. Kukko, and T. Heinonen, "Remote sensing methods for power line corridor surveys," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 119, pp. 10–31, 2016. [Citado na página 1]

[3] C. Sun, R. Jones, H. Talbot, X. Wu, K. Cheong, R. Beare, M. Buckley, and M. Berman, "Measuring the distance of vegetation from powerlines using stereo vision," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 60, no. 4, pp. 269–283, 2006. [Citado na página 2]

[4] F. Ross, R. A. Evaluation, S. J. Mills, M. P. Gerardo, Z. Li, J. Cai, R. Hayward, and R. A. Walker, "Evaluation of Aerial Remote Sensing Techniques for Vegetation Management in Powerline Corridors," *IEEE Transactions on Geoscience and Remote Sensing*, no. 2009, 2010. [Citado na página 2]

[5] Rachit Patel(), Sapna Katiyar and K. Arora, "Metadata of the chapter that will be visualized in Online," *Springer Nature Singapor*, no. August, pp. 1–8, 2016. [Citado na página 2]

[6] R.-t. P. Line and D. Combining, "´ rio de Sistemas Aut o ´ nomos Laborat o orio Final Real-time Power Line Detection Combining IR and Visual Images from a UAV," 2023. [Citado na página 3]

[7] B. I. O. Andr and C. Azevedo, "Real-Time LiDAR-based Power Lines Detection for," 2018. [Citado na página 3]

[8] S. Dong, P. Wang, and K. Abbas, "A survey on deep learning and its applications," *Computer Science Review*, vol. 40, may 2021. [Citado na página 5]

[9] R. Y. Choi, A. S. Coyner, J. Kalpathy-Cramer, M. F. Chiang, and J. Peter Campbell, "Introduction to machine learning, neural networks, and deep learning," *Translational Vision Science and Technology*, vol. 9, no. 2, pp. 1–12, 2020. [Citado na página 5]

[10] E. Ahmed, A. Saint, A. E. R. Shabayek, K. Cherenkova, R. Das, G. Gusev, D. Aouada, and B. Ottersten, "A survey on Deep Learning Advances on Different 3D Data Representations," no. August, 2018. [Citado nas páginas 6, 13, 14 e 23]

[11] Y. Guo, H. Wang, Q. Hu, H. Liu, L. Liu, and M. Bennamoun, "IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE 1 Deep Learning for 3D Point Clouds: A Survey," [Citado nas páginas 6 e 23]

[12] Y. Guo, H. Wang, Q. Hu, H. Liu, L. Liu, and M. Bennamoun, "IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE 1 Deep Learning for 3D Point Clouds: A Survey," [Citado nas páginas 6 e 23]

[13] A. Diab, R. Kashef, and A. Shaker, "Deep Learning for LiDAR Point Cloud Classification in Remote Sensing," *Sensors (Basel, Switzerland)*, vol. 22, no. 20, pp. 1–14, 2022. [Citado na página 7]

[14] B. Li, T. Zhang, and T. Xia, "Vehicle Detection from 3D Lidar Using Fully Convolutional Network," [Citado na página 7]

[15] K. Minemura, H. Liau, A. Monrroy, and S. Kato, "LMNet: Real-time Multiclass Object Detection on CPU using 3D LiDAR," [Citado na página 7]

[16] S. L. Yu, T. Westfechtel, R. Hamada, K. Ohno, and S. Tadokoro, "Vehicle detection and localization on bird's eye view elevation images using convolutional neural network," *SSRR 2017 - 15th IEEE International Symposium on Safety, Security and Rescue Robotics, Conference*, pp. 102–109, 2017. [Citado nas páginas 7 e 8]

[17] J. Beltrán, C. Guindel, F. M. Moreno, D. Cruzado, F. García, and A. De La Escalera, "BirdNet: a 3D Object Detection Framework from LiDAR information," [Citado na página 8]

[18] A. Barrera, C. Guindel, J. Beltrán, and F. García, "BirdNet+: End-to-End 3D Object Detection in LiDAR Bird's Eye View," [Citado na página 8]

[19] W. Ali, S. Abdelkarim, M. Zahran, M. Zidan, and A. E. Sallab, "YOLO3D: End-to-end real-time 3D Oriented Object Bounding Box Detection from LiDAR Point Cloud," [Citado na página 8]

[20] R. Ma, C. Chen, B. Yang, D. Li, H. Wang, Y. Cong, and Z. Hu, "CG-SSD: Corner Guided Single Stage 3D Object Detection from LiDAR Point Cloud," [Citado na página 8]

[21] Y. Zhou and O. Tuzel, "VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection," [Citado nas páginas 9 e 12]

[22] Y. Yan, Y. Mao, and B. Li, "Second: Sparsely embedded convolutional detection," *Sensors (Switzerland)*, vol. 18, no. 10, pp. 1–17, 2018. [Citado nas páginas 10 e 40]

[23] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation," [Citado nas páginas 10, 19 e 40]

[24] C. R. Q. Li, Y. Hao, S. Leonidas, and J. Guibas, "PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space," [Citado nas páginas 11 e 19]

[25] T. Yin, X. Zhou, and P. Krähenbühl, "Center-based 3D Object Detection and Tracking," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, no. Figure 1, pp. 11779–11788, 2021. [Citado nas páginas 12, 27, 28 e 40]

[26] T. Yin, X. Zhou, and P. Krähenbühl, "Center-based 3D Object Detection and Tracking," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, no. Figure 1, pp. 11779–11788, 2021. [Citado na página 12]

[27] J. Ku, M. Mozifian, J. Lee, A. Harakeh, and S. L. Waslander, "Joint 3D Proposal Generation and Object Detection from View Aggregation," [Citado nas páginas 13 e 14]

[28] C. R. Qi, W. Liu, C. Wu, H. Su, and L. J. Guibas, "Frustum PointNets for 3D Object Detection from RGB-D Data," [Citado na página 15]

[29] K. Shin, P. Kwon, and M. Tomizuka, "RoarNet: A Robust 3D Object Detection based on RegiOn Approximation Refinement," [Citado na página 15]

[30] L. Emerging and S. Conference, "A Brief Survey on 3D Semantic Segmentation of Lidar Point Cloud with Deep Learning," *NILES 2021 - 3rd Novel Intelligent and Leading Emerging Sciences Conference, Proceedings*, pp. 405–408, 2021. [Citado na página 17]

[31] C. Xu, B. Wu, Z. Wang, W. Zhan, P. Vajda, K. Keutzer, and M. Tomizuka, "SqueezeSegV3: Spatially-Adaptive Convolution for Efficient Point-Cloud Segmentation," [Citado na página 17]

[32] B. Wu, A. Wan, X. Yue, and K. Keutzer, "SqueezeSeg: Convolutional Neural Nets with Recurrent CRF for Real-Time Road-Object Segmentation from 3D LiDAR Point Cloud," [Citado na página 17]

[33] Y. A. Alnaggar, M. Afifi, K. Amer, and M. Elhelw, "Multi Projection Fusion for Real-time Semantic Segmentation of 3D LiDAR Point Clouds," [Citado na página 18]

[34] Q. Hu, B. Ang, L. Xie, S. Rosa, Y. Guo, Z. Wang, N. Trigoni, and A. Markham, "RandLA-Net: Efficient Semantic Segmentation of Large-Scale Point Clouds," [Citado na página 19]

[35] X. Zhu, H. Zhou, T. Wang, and F. Hong, "Cylindrical and Asymmetrical 3D Convolution Networks," *Cvpr*, 2021. [Citado na página 20]

[36] Z. Zhuang, R. Li, K. Jia, Q. Wang, Y. Li, and M. Tan, "Perception-Aware Multi-Sensor Fusion for 3D LiDAR Semantic Segmentation," *Proceedings of the IEEE International Conference on Computer Vision*, no. Iccv, pp. 16260–16270, 2021. [Citado nas páginas 21 e 22]

[37] A. Dias, J. Almeida, A. Oliveira, T. Santos, A. Martins, and E. Silva, "Unmanned aerial vehicle for wind-turbine inspection. next step: Offshore," in *OCEANS 2022, Hampton Roads*, pp. 1–6, 2022. [Citado nas páginas 35 e 36]

[38] M. Contributors, "MMDetection3D: OpenMMLab next-generation platform for general 3D object detection." `https://github.com/open-mmlab/mmdetection3d`, 2020. [Citado na página 38]

[39] K. Chen, J. Wang, J. Pang, Y. Cao, Y. Xiong, X. Li, S. Sun, W. Feng, Z. Liu, J. Xu, Z. Zhang, D. Cheng, C. Zhu, T. Cheng, Q. Zhao, B. Li, X. Lu, R. Zhu, Y. Wu, J. Dai, J. Wang, J. Shi, W. Ouyang, C. C. Loy, and D. Lin, "MMDetection: Open mmlab detection toolbox and benchmark," *arXiv preprint arXiv:1906.07155*, 2019. [Citado na página 38]

[40] T. Yin, X. Zhou, and P. Krähenbühl, "Center-based 3d object detection and tracking," *CVPR*, 2021. [Citado na página 40]

[41] X. Zhu, H. Zhou, T. Wang, F. Hong, Y. Ma, W. Li, H. Li, and D. Lin, "Cylindrical and asymmetrical 3d convolution networks for lidar segmentation," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 9939–9948, 2021. [Citado na página 40]

[42] Y. Huang, W. Zheng, Y. Zhang, J. Zhou, and J. Lu, "Tri-perspective view for vision-based 3d semantic occupancy prediction," *arXiv preprint arXiv:2302.07817*, 2023. [Citado na página 40]

[43] A. K. Danila Rukhovich, Anna Vorontsova, "Fcaf3d: Fully convolutional anchor-free 3d object detection," in *European conference on computer vision*, 2022. [Citado na página 40]

[44] "Adam — PyTorch 2.0 documentation." [Citado na página 41]

[45] Ahmed Fawzy Gad, "Accuracy, Precision, and Recall in Deep Learning | Paperspace Blog," 2020. [Citado na página 42]