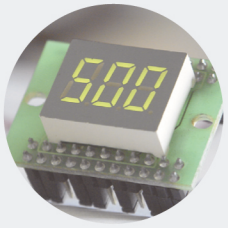
INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA ELECTROTÉCNICA E DE COMPUTADORES






Real-Time LiDAR-based Power Lines Detection for Unmanned Aerial Vehicles

FÁBIO ANDRÉ COSTA AZEVEDO
novembro de 2018

POLITÉCNICO
DO PORTO

Real-Time LiDAR-based Power Lines Detection for Unmanned Aerial Vehicles

Fábio André Costa Azevedo
Nº 1120356

Master in Electrical and Computer Engineering
Branch of Autonomous Systems
Electrical Engineering Department

Instituto Superior de Engenharia do Porto

2018

Dissertation for partial satisfaction of the requirements of the Master in Electrical and Computer Engineering

Candidate: Fábio André Costa Azevedo
Nº 1120356

Supervisor: José Miguel Soares De Almeida
jsa@isep.ipp.pt

Co-Supervisor: Dr. André Miguel Pinheiro Dias
apd@isep.ipp.pt

Master in Electrical and Computer Engineering
Branch of Autonomous Systems
Electrical Engineering Department

Instituto Superior de Engenharia do Porto

November 21, 2018

Agradecimentos

Esta dissertação foi resultado do contributo, direto ou indireto, de diversas pessoas, às quais deixo, desde já, uma palavra de sincera gratidão.

Gostaria de começar os meus agradecimentos pelos Engenheiros José Almeida e André Dias, tanto pelo desafio apresentado, ao me atribuirem um tema ainda pouco explorado, como pelo auxílio prestado durante o desenvolvimento deste projeto.

A todos os que fazem ou já fizeram parte do LSA, estou grato por tudo o que me ensinaram, pela ajuda que me forneceram e pelos momentos de convívio.

Agradeço também aos meus colegas de curso pela caminhada conjunta, em que todos trocámos impressões e conhecimentos benéficos.

Dirijo ainda um grande agradecimento aos meus colegas de grupo de trabalho, Alexandre Oliveira, André Ferreira, Miguel Moreira e Tiago Santos, tanto pelas conquistas obtidas, como pelo apoio constante que me deram, não esquecendo os momentos divertidos! Ao João Pedro Ribeiro fico grato pelas dicas, opiniões e ajuda na resolução de diversos problemas. Ao Denys Sytnyk, por ser o Dinis...

Para com os meus pais, Avelino Azevedo e Eugénia Azevedo, estarei eternamente grato pelos esforços que fizeram, pela educação que me proporcionaram e pelas condições que criaram para que pudesse chegar até aqui.

O meu reconhecimento final é para alguém muito especial na minha vida: a minha namorada e companheira. Obrigado Carla Gomes! Pelo teu carinho, compreensão, apoio e motivação, nunca esquecendo das coisas de que abdicaste em prol do meu trabalho.

Obrigado.

Fábio André Costa Azevedo

This page was intentionally left blank.

Abstract

The growing dependence of modern-day societies on electricity leads to the increasing importance of effective monitoring and maintenance of power lines. Due to the population's renouncement to the installation of new electric power lines, the existing ones are constantly operating at maximum capacity. This leaves no room for breakdowns, as it leads to major economic losses for the electrical companies and blackouts for the consumers.

Endowing Unmanned Aerial Vehicles (UAVs) with the appropriate sensors for inspection the power lines, the costs and risks associated with the traditional foot patrol and helicopter-based inspections can be reduced. However, this implies the development of algorithms to make the inspection process reliable and autonomous.

Visual detection methods are usually applied to locate the power lines and their components. Although, they are generally too sensitive to atmospheric conditions and noisy background. Poor light conditions or a background rich in edges may compromise their results. In order to overcome those limitations, this dissertation addresses the problem of power line detection and modeling based on the use of a Light Detection And Ranging (LiDAR) sensor.

A novel approach to the power line detection was developed, the Power Line LiDAR-based Detection and Modeling (PL^2DM). It is based in a scan-by-scan adaptive neighbor minimalist comparison for all the points in a point cloud. In the segmentation, the breaking cluster points are detected by an analysis of their planar properties.

Exporting the potential power line points to a further step, it performs a scan based straight line detection. The final model of the power line is obtained by matching and grouping the several line segments detected using their collinearity properties. Horizontally, the power lines are modeled as a straight line, while vertically are approximated to a catenary curve.

The algorithm was tested with a real dataset, showing promising results both in terms of outputs and processing time. From there, it was demonstrated that the proposed algorithm can be applied to real-time operations of the UAV, adding object-based perception capabilities for other layers of processing.

Keywords: Power line, LiDAR, real-time, UAV, point cloud, segmentation, catenary

Resumo

A crescente dependência das sociedades modernas no uso de eletricidade conduz a uma crescente importância da eficiência da monitorização e manutenção das linhas elétricas. A renitência das populações à instalação de novas linhas elétricas faz com que as existentes estejam constantemente a operar na sua máxima capacidade. Isto faz com que não possam existir falhas, uma vez que resultariam em grandes perdas económicas para as companhias elétricas e em falhas energéticas para os consumidores.

Equipando um Unmanned Aerial Vehicle (UAV) com os sensores adequados à inspeção de linhas elétricas, podem ser reduzidos os custos e riscos de operação associados às inspeções tradicionais, baseadas em patrulhas pedonais e no uso de um helicóptero. No entanto, isto implica o desenvolvimento de algoritmos para que o processo de inspeção seja fiável e autónomo.

As linhas elétricas e os componentes associados são geralmente localizados através de métodos de deteção visual. Estes métodos são, geralmente, muito sensíveis às condições atmosféricas e a fundos ruidosos. Condições de luz deficientes ou fundos ricos em contrastes são alguns dos fatores que podem comprometer os seus resultados. De forma a ultrapassar essas limitações, esta dissertação endereça o problema da deteção e modelação de linhas elétricas, tendo por base o uso de um sensor Light Detection And Ranging (LiDAR).

Foi desenvolvida uma nova abordagem aos métodos de deteção de linhas elétricas, o Power Line LiDAR-based Detection and Modeling (PL^2DM). Esta abordagem é baseada numa análise individual de varrimentos, em que é feita uma comparação minimalista de todos os pontos, presentes numa dada nuvem de pontos, com uma vizinhança adaptativa. Na segmentação, os pontos de quebra dos grupos criados são detetados tendo em conta as suas propriedades planares.

Passando os pontos passíveis de pertencerem a linhas elétricas para o processamento

seguinte, é realizada, em cada varrimento, uma deteção de linhas retas. O modelo final das linhas elétricas é obtido a partir da associação e agrupamento dos diversos segmentos de reta detetados, tendo por base a sua colinearidade. Na sua projeção horizontal, as linhas elétricas são modeladas como linhas retas. Verticalmente, são aproximadas ao modelo de uma curva catenária.

O algoritmo foi testado com um conjunto de dados reais, tendo mostrado resultados promissores, tanto em termos de dados gerados como de tempo de processamento. Com isso, ficou demonstrado que o algoritmo proposto pode ser aplicado nas operações do UAV em tempo real, adicionando capacidades de perceção baseada em objetos para outras camadas de processamento.

Palavras-Chave: Linhas elétricas, LiDAR, tempo real, UAV, nuvem de pontos, segmentação, catenária

Contents

This page was intentionally left blank.

List of Figures

This page was intentionally left blank.

List of Tables

This page was intentionally left blank.

List of Algorithms

This page was intentionally left blank.

List of Acronyms

1D One-Dimensional

2D Two-Dimensional

2.5D Two-and-a-Half-Dimensional

3D Three-Dimensional

ABD Adaptive Breakpoint Detection

ALS Airborne Laser Scanning

CLF Compass Line Filter

CRAS Centre for Robotics and Autonomous Systems

DiAL Differential Absorption LiDAR

DSM Digital Surface Model

e-RBNN Ellipsoid model based RBNN

ED Euclidian Distance

EDP Energias de Portugal

ENU East-North-Up

ESCS Extruded Surface of Cross Section

FOV Field-Of-View

GNSS Global Navigation Satellite System

GP Gaussian Process

GP-INSAC GP Incremental Sample Consensus

GPF Ground Plane Fitting

GPU Graphics Processing Unit

HMM Hidden Markov Model

HT Hough Transform

IMU Inertial Measurement Unit

INESC TEC Institute for Systems and Computer Engineering, Technology and Science

ISEP Engineering School of Porto Polytechnic

kNN k-Nearest Neighbor

LAN Local Area Network

LiDAR Light Detection And Ranging

LSA Autonomous Systems Laboratory

MEMS MicroElectroMechanical System

NB Naive Bayes

NMEA National Marine Electronics Association

NN Nearest Neighbor

OD Orthogonal Distance

P2P Peer-To-Peer

PCA Principal Component Analysis

PL^{2}DM Power Line LiDAR-based Detection and Modeling

PPS Pulse Per Second

RaDAR Radio Detection And Ranging

RANSAC RANdom SAmple Consensus

RBNN Radially Bounded Nearest Neighbors

ROS Robotic Operating System

RPCA Robust PCA

RTK Real-Time Kinematic

RWR Rolling on Wires Robot

SAR Synthetic Aperture Radar

SLR Scan Line Run

SPR Portuguese Society of Robotics

SVM Support Vector Machine

TIN Triangular Irregular Network

ToF Time-of-Flight

UAV Unmanned Aerial Vehicle

UTC Coordinated Universal Time

VPLD Voxel-based Piece-wise Line Detector

This page was intentionally left blank.

Chapter 1

Introduction

During the last years, the application scenarios with Unmanned Aerial Vehicles (UAVs) have been growing due to an increase in the research effort on the field of aerial robotics [1]. Furthermore, this wider set of applications is related to a continuous change on the research focus, which is starting to focus to higher level tasks (such as navigation and task planning, paying attention to visual odometry, localization and mapping). Until the beginning of this century, the UAV research was mainly focused on hardware development, modeling and control. In the most recent years, researchers are giving more importance to topics related to obstacle detection and collision avoidance.

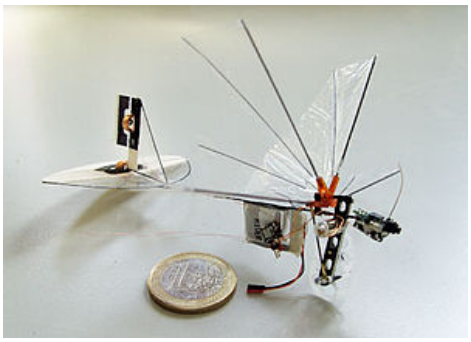
At this point of development, the UAVs can be applied for search and rescue operations, surveillance and inspection of structures, among others [2]. Depending on its application, there are different types of UAVs more well prepared to achieve better performance. Fixed-wings UAVs (figure 1.1(a)) have the advantage of flying longer distances than a multirotor (figure 1.1(c)), however, they lack on the maneuverability offered by the last. Hybrid systems (figure 1.1(b)) are a mixed type of the other two, trying to get the advantages of both systems but increasing the complexity of its control. Ornithopters (figure 1.1(d)) are a type of drones that fly by mimicking wing motions of insects or birds but they are not widely used.

Considering the existing types of UAVs, the close inspection of a structure that is located at a low altitude is an application that is performed better by a multirotor, due to its high Three-Dimensional (3D) maneuverability and capability of operating in harsh environments. Besides that, this type of drone is interesting for electrical power providers [6–8], as it can support a reasonable payload of sensors and allows the data collection from different positions, angles and distances, making it suitable for the inspection of electric assets, such as electric pylons and insulators.

(a) Fixed-wing (FALCOS) [3]

(b) Hybrid (Greased Lightning) [4]

(c) Multirotor (OTUS)



(d) Ornithopter (DelFly Micro) [5]

Figure 1.1: UAV examples.

Due to the growing dependence of modern-day societies on electricity, there is an increasing importance of effective monitoring and maintenance of power lines [9], as the distribution of the power from the source to the user is one of the key factors in the quality of the service provided by the electrical power companies. Considering the progressive development of new green power generation plants in Europe, the need for a higher and more reliable electrical transport capacity is growing, however, the installation of new electric power lines is usually not accepted by the population [10]. This leads to a constant operation at the maximum capacity of the power lines, without having redundancies or reserves to compensate breakdowns. In order to avoid possible economic losses and blackouts for the consumers, the electrical power companies need to adopt a preventive and predictive maintenance philosophy by means of, for example, a periodic visual and thermal inspection [10, 11].

Power lines inspection takes into account not only their elements but also the surrounding objects, especially vegetation [10–12]. The power lines are present in both urban and rural environments. In an urban environment, it is quite easy to ensure a safety

separation from the man-made structures by applying building regulations, however, a rural environment has usually an unmanaged growth of the surrounding vegetation [13]. When the minimum clearance between the vegetation and the conductors or assets is violated, tree falls (figure 1.2(a)) or conductors oscillation during bad weather conditions can lead to the short-circuiting of the line, causing widespread outages [14–16] or even bush-fires, especially in drier environments [11–13, 17, 18]. These threats result in a general acceptance on considering vegetation as one of the most hazardous factors for the overhead power lines' integrity [12–16]. The condition of the power components needs to be checked for both mechanical and electrical faults [12, 19], which occur on conductors, insulators (figure 1.2(b)), electric pylons and other power line equipment [20, 21]. Damaged or corroded parts and contaminated conductors are some of the problems that can be visually detected [19, 20, 22], but corona or gap discharges can also indicate problems on the transmission line [20].

(a) Tree fall over line [23] (b) Defective insulator [24]

Figure 1.2: Power line problems examples.

The inspections of the power lines traditionally rely on human labour [10–13], as they are primarily performed on foot by a team driving between spans (space between two consecutive electric pylons) or from a helicopter flying alongside the line. The power structures can be installed on complex and harsh environments, which makes the on-foot patrols time-consuming and poorly efficient. Helicopter-based inspections are more time-efficient but are both expensive and taxing on the pilot and operator [13, 22, 25].

To overcome those limitations, some remote sensing methods have been proposed to assist or replace traditional ones. Synthetic Aperture Radar (SAR) images [26, 27], optical satellite images [13,28] or Airborne Laser Scanning (ALS) data [29–31] are some of the examples, however, due to the lack of resolution and optical or thermal images, these

methods can only be used for mapping pylons and conductors or vegetation monitoring [11]. A proper inspection of the power line elements can be done by using laser scanning data, optical [32] and thermal [33, 34] images obtained by UAVs [10, 22, 35, 36], Rolling on Wires Robots (RWRs) [10,37] or even land-based mobile systems [38]. Regarding the diversity of the data, fixed-wing UAVs can only provide information about the upper part of the line, contrasting with land-based systems, that only give information about the lower one. RWRs (figure 1.3(a)) need the human intervention to first put them on a conductor and a mechanism to overpass obstacles like insulators, warning spheres and pylons. The constant contact with the conductor can be a disadvantage for this type of robots, as they need a stronger electromagnetic shield. The advantages mentioned above for the multirotor UAVs (figure 1.3(b)) are then confirmed, being those the most suitable vehicles for a more complete inspection of the power line elements.

(a) RWR (LineScout) [39] (b) Multirotor UAV (Byrd II)

Figure 1.3: Power line inspection robots.

Despite the advantages, using a multirotor UAV in a complex and confined environment, as one where a power line can be inserted in, can be challenging for the pilot and the autonomous mission planning. A noisy background (e.g., with vegetation) can make the conductors visually undetectable by the pilot [22]. This arises the need for an autonomous module of obstacle detection and collision avoidance that is capable of actuating in both autonomous and assisted modes [40]. The detection of conductors can also be used for improving the navigation filter and line following maneuvers [22].

In order to increase the perception capabilities of UAVs and allow a safer operation, this dissertation addresses the development of an algorithm for real-time obstacle detection using Light Detection And Ranging (LiDAR) data. The main focus is the detection of power line conductors and the prediction of their extent, as they can be detected as sparse points and be ignored by the collision avoidance module.

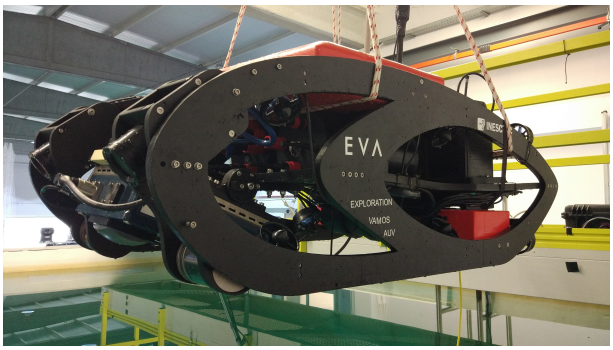
1.1 Background and Motivation

The Centre for Robotics and Autonomous Systems (CRAS)[1], from the Institute for Systems and Computer Engineering, Technology and Science (INESC TEC)[2], and the Autonomous Systems Laboratory (LSA)[3], from the Engineering School of Porto Polytechnic (ISEP)[4], have been combining efforts, during the latest years, for the development of autonomous systems. This partnership has resulted already in the development of several robotic systems for multiple domains: land, water surface, underwater, and air. Their experience in UAVs has started with the development of the fixed-wing FALCOS and GRIFO, but is now more directed to multirotor UAVs, with the OTUS [41], STORK [42] and Byrd II.

(a) TURTLE [43]

(b) ROAZ II [44]

(c) STORK



(d) EVA

Figure 1.4: CRAS and LSA robots.

[1] https://www.inesctec.pt/en/centres/cras
[2] https://www.inesctec.pt/en
[3] http://lsa.isep.ipp.pt/
[4] http://isep.ipp.pt/

CRAS is involved in several projects related to multirotor UAVs, such as ROSM, SpilLess[5] [45, 46] and Drone Project[6] [47]. The later is a partnership with Energias de Portugal (EDP) Labelec[7] and aims to analyze and develop drone-based solutions for the inspection of electrical sector assets, such as wind turbines, dams, substations and power line elements. With this project, INESC TEC and ISEP earned the *Innovation* and *Supplier of the Year* awards, from the third edition of the EDPartners[8], and the *Innovation and Technology Transfer in the Field of Robotics Award*, in 2017, from the Portuguese Society of Robotics (SPR)[9]. The team has also won the *Grand Challenge* of two international competitions of search and rescue robots: euRathlon[10] [48, 49], in 2015, and ERL Emergency Robots[11] [50], in 2017, using the OTUS and STORK UAVs, respectively.

The Drone Project has already produced the second version of its UAV, the Byrd II, which has a payload that is capable of obtaining high-resolution images, thermal images, and LiDAR data. All the data is georeferenced, allowing EDP Labelec to post-process it at the office, evaluate the state of the assets and generate detailed reports.

Regarding the specific application of power line elements inspection in the Drone Project, it was already developed and applied an obstacle detection module that is a variant of the work presented in [40]. Due to the low timestamp resolution of the point cloud points (a single timestamp for a complete scan of the LiDAR), some measured points can be transformed from the sensor to the world frame with a considerable error, generating an obstacle with low accuracy in position. This could lead to a collision with the real position of the power line. That problem can be more noticeable with some possible processor overload, that introduces a delay to the transformation matrix, or due to the error of a navigation filter based in low-cost sensors.

Detecting the power line conductors can help not only to the reduction of the obstacle estimated position error effect, but also to predict where the conductors might be by fitting the sparse detected points into a catenary curve model [51]. This model also allows to implement line following algorithms.

[5] `https://spilless.ciimar.up.pt/`
[6] `https://www.edplabelec.com/pt-pt/node/37658`
[7] `https://www.edplabelec.com/en`
[8] `https://www.edpartners.edp.pt/index.php/en/`
[9] `http://www.sprobotica.pt/`
[10] `https://www.eurathlon.eu/`
[11] `https://www.eu-robotics.net/robotics_league/erl-emergency/about/index.html`

1.2 Objectives

This dissertation addresses the fast extraction of power line conductors from a point cloud, using a LiDAR sensor. This allows to improve the obstacle detection and collision avoidance modules, thus increasing the perception capabilities of the UAV. This capability will provide to the UAV an improvement of autonomy for the inspection of power lines, either in autonomous and pilot-assisted modes.

In order to accomplish the main objective of this dissertation, there are some intermediate goals that need to be addressed:

- Study of the existing works related to the power lines extraction, either by vision and using LiDAR;
- Analysis of the LiDAR raw data information before the point cloud generation in order to identify possible improvements;
- Development of a segmentation step capable of extracting candidate points that could represent a power line inside the received point cloud;
- Creation of a line detection method and mathematical modeling methodology for the detected points;
- Development of a robust algorithm, capable of operating in real-time and successfully detect and model power lines based on the application requirements;
- Validation of the developed method for different environments.

1.3 Structure

This dissertation is divided into eight main chapters. In the next chapter is presented a preliminary study of the works related to the dissertation's topic. It ends with a discussion of their properties, evidencing their advantages and pointing the drawbacks while trying to detect the common assumptions made.

In chapter 3 are listed and described the concepts in which some parts of the algorithm are based. In the section related to LiDAR sensors, is exposed a list of their types, detailing the properties of a spinning LiDAR. The chapter 4 contains an overview of the hardware and software architectures needed to implement the developed algorithm.

The description of the proposed algorithm is made along the chapter 5. There, is presented the base concept of the algorithm, followed by a complete description of

the details associated to each part of the processing. Knowing those properties and considering the algorithm's requirements is crucial for having a proper functionality.

Chapter 6 lists the specifications of the used UAV and LiDAR sensor. At the end is also detailed the software changes done to make them compatible with the requirements presented in the previous chapter.

The outputs of the algorithm for a dataset are presented and analyzed in chapter 7. Here is also made an evaluation to the performance of the developed algorithm.

Finally, in chapter 8 are made the final remarks and a general analysis of the developed work. Some other detailed conclusions are also presented over the other chapters. The chapter 8 ends with suggestions for further work that either brings advantages to this dissertation or can be made on top of its outputs.

Chapter 2

Related Work

In this chapter is exposed some background about the use of Light Detection And Ranging (LiDAR) sensors for some applications. For an overview closer to the dissertation subject, some works related to the segmentation of point clouds based on LiDAR sensors are presented. The particular case of power lines detection is also approached.

The study of existing methods allows a better understanding of the possible challenges related to line recognition. The chapter ends with a brief analysis of the main advantages and disadvantages of the current state of the art.

2.1 LiDAR

LiDAR is an active remote sensor that uses a laser in the visible or near-visible part of the electromagnetic spectrum (figure 2.1) to obtain measures [52]. For this reason, it is also known as optical or laser Radio Detection And Ranging (RaDAR), since they are only separated by their energy source [53]. As its wavelength is smaller than the RaDAR's, it is able to detect tiny objects, such as particles in the atmosphere. LiDAR sensors are divided into three major varieties: range finders, Differential Absorption LiDAR (DiAL), and Doppler DiAL. Their type of measurements can range from altitude, shape, and size of landscape features to atmospheric vertical profiles of aerosols and trace gas densities. Wind velocities are also possible to measure with the Doppler DiAL, based on the Doppler Shift effect [54].

Scientists have used this kind of sensors since the 1960s [52, 56]. The first LiDAR measurements were made in 1963 [57] and, during that decade, it was clearly established that this kind of sensors could provide an optimal source of electromagnetic radiation

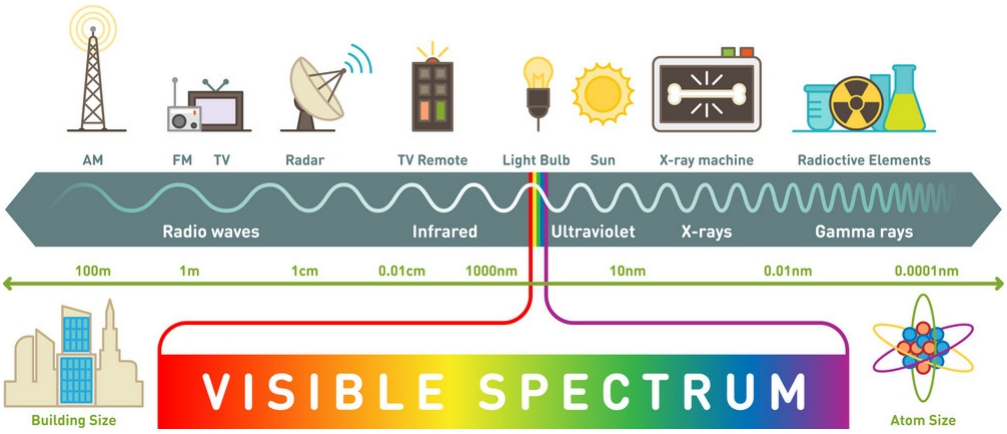


Figure 2.1: Electromagnetic spectrum (adapted from [55]).

for atmospheric studies [58–61]. At the beginning of the 1970s were made some of the early space-based LiDAR measurements from lunar orbit using the laser altimeter on the Apollo 15 mission [62]. Since the first lasers became widely available, ground-based and airborne LiDAR systems started to increase steadily [52].

2.1.1 Segmentation

In the first half of the 1990s, LiDARs began to be assumed as sensors that could provide meaningful data for efficient Three-Dimensional (3D) terrain surveys [63], when associated to an accurate positioning and orientation of the vehicle [64]. Their use combined with the so far used photogrammetry allowed the obtaining of more precise Digital Surface Models (DSMs) and triggered the development of new algorithms for extracting man-made structures and vegetation from urban environments [65–70], and create accurate 3D city models [71–73].

The extraction of valuable spatial data from the large amount of information that LiDAR sensors can provide is difficult and time-consuming, therefore, segmentation is generally a prerequisite for feature extraction. The first segmentation techniques relied on Two-and-a-Half-Dimensional (2.5D) grid or image data [73, 74], where the point cloud was interpolated to allow the application of some image-based segmentation and classification, however, some important spatial information could be lost [70, 75].

To overcome the loss of spatial information, Wang and Tseng, in [76], proposed an octree-structure-based split-and-merge segmentation method more suitable for LiDAR data. The method consists in hierarchically splitting a point cloud set on the octree structure until all the sub-node points are coplanar or less than 3 (minimum number of

points to generate a plane). Using the octree structure for neighboring, 3D planes with similar attributes are merged and a Triangular Irregular Network (TIN) is generated for visualization. In 2005, the same authors [77] complemented their previous work by adding the automatic extraction of edges and corners using the intersection of neighboring calculated 3D best-fit planes. Later, Kutty and Ayyappan [78] proposed a quad-tree segmentation of horizontal projected points that are merged by analyzing a constructed TIN.

In [79], Shan and Sampath presented a binary segmentation method that labels each point of an urban environment in either ground or non-ground using LiDAR raw data. It is based on a sequential evaluation of each two-point sets along the One-Dimensional (1D) LiDAR profile both in slope and elevation. Positive slopes can represent ground to non-ground point transitions, and negative, the inverse. Elevation evaluation helps the point labeling. The final classification results from the combination of the bidirectional labeling followed by a local linear regression to remove some possible wrong-labeled points.

In 2008, Steinhauser *et al.* [80] proposed an algorithm to find drivable road using a LiDAR mounted on a ground vehicle. It assumes that the road is a connected structure and analyzes the Two-Dimensional (2D) scan from each angular step of the sensor. A line fitting is then performed to mark the points as obstacle's, surface's or critical surface's (the last can be either drivable or not depending on the off-road capabilities of the vehicle). Removing the points classified as passable environment, the obstacle points are then clustered or not based in an Euclidian Distance (ED) threshold. After detecting some stationary objects, they also detect and predict the ego-motion of the vehicle.

Regarding urban environments, Moosmann *et al.* [81] presented a solution for dealing with non-flat grounds by means of local convexity criteria (figure 2.2). Using ordered

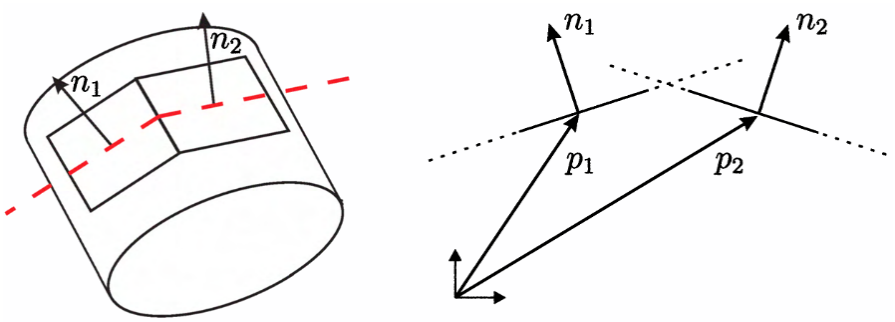


Figure 2.2: Local convexity criteria (adapted from [81]). Local convexity holds if the center point of a surface is below the other surface and vice versa.

points from the LiDAR sensor, the estimated local surface normal results from the average of the displacement vectors cross products between the left and lower, lower and right, right and upper, and upper and left vectors, as in [82]. The segmentation is based on the comparison between two neighbor surfaces. Given the graph, the region growing algorithm is executed, which randomly selects a seed node and the segment grows until no more nodes are added. The segment is then removed from the graph and the operation is repeated until no more surfaces are left. Analyzing the segment normal vector's z value histogram, if the topmost bin contains more votes, it is classified as ground, otherwise, as an obstacle.

Klasing *et al.* have developed a segmentation technique based on the Radially Bounded Nearest Neighbors (RBNN) graph [83] to overcome some processing time and resultant clusters limitations of the k-Nearest Neighbor (kNN) graph [85]. For improving the resultant clusters, in the RBNN graph, every node is connected to all neighbors that lie within a predefined radius r and the clusters with less than n_min points are ignored (figure 2.3). The speedup is achieved by skipping the verification for all the points that have been already associated to a cluster. Although the clustering using RBNN is an interesting candidate technique for real-time processing, it cannot be directly applied for continuous stream of data mainly because the Nearest Neighbor (NN) searching is based on a static *kd-tree*. Thence, the author presented a refined version of RBNN algorithm [84], with real-time capabilities, that continuously monitors NNs and uses a feature space consisting on both incoming points and their estimated normal

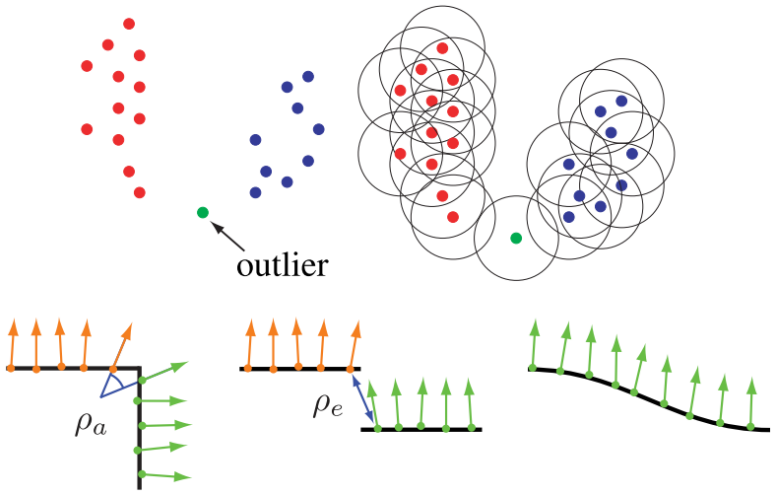


Figure 2.3: RBNN clustering (on top - adapted from [83]) and normal plane vector based segmentation (on bottom - adapted from [84]).

vectors for clustering. Knowing the structure of the range sensor, the obtained points can be ordered by scanning angles and consecutive scans. Using those ordered points, the continuous NNs monitoring relies on a *k-window* search. Given the neighborhood of a point, the algorithm uses the PlanePCA method [86] for estimating the normal vector, as it has shown, in [87], to be the best method both in quality and computational time.

Other segmentation techniques also use Principal Component Analysis (PCA) to get the saliency features [88–91], however it is very sensitive to outliers [92], which lead to a poor plane fitting, due to a bad estimation of normals [93,94]. Aware of these limitations, in [95,96] is presented a Robust PCA (RPCA). This region-growing algorithm starts by selecting as seed the point that presents the least surface curvature. Then it finds the k neighbors of the seed points, calculates the ED, Orthogonal Distance (OD) to the best fitting plane, and the angular difference of the normals. If all the calculated values are below a threshold, the neighbor is added to the current region and removed from the input point cloud, being considered as a seed point for the following iterations. When no more seed points are found to the current region, the region size is evaluated and it is considered valid if the size is above a threshold. The algorithm repeats until no more points are available. The RPCA showed to be more robust than the traditional PCA, especially for non-planar surfaces segmentation.

To segment a point cloud obtained by a ground vehicle, in [98] is presented a combination of 2D and 3D data processing techniques. It generates a 2.5D ego-centered occupancy grid (figure 2.4), at each LiDAR revolution, that stores the maximum absolute difference in z-coordinates of all points falling into the respective grid cell, like in [99]. That grid is then binarized for the segmentation procedure by means of a defined threshold and a bounding box is created around the detected obstacles. The 3D points

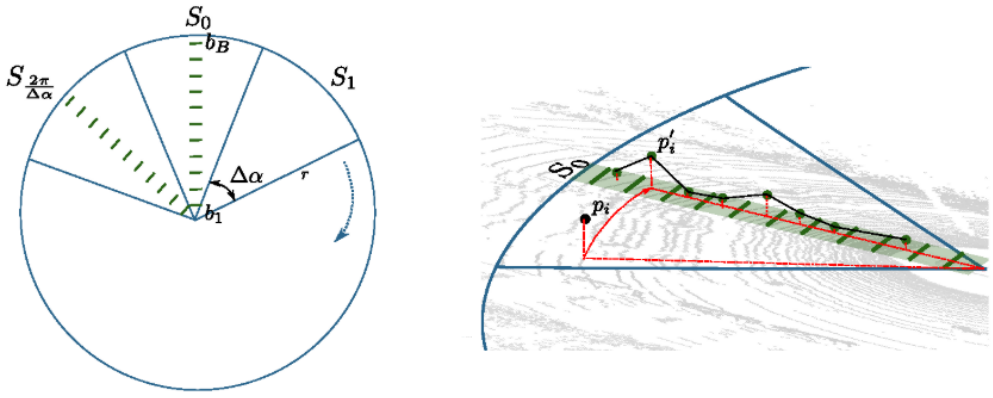


Figure 2.4: Partioned radial grid into bins (adapted from [97]).

of the bounding boxes are then recovered and the object classification is made using point feature histograms and a Support Vector Machine (SVM), previously trained by hand-labeled point clouds. For preventing the under-segmentation of data due to the dimensionality reduction introduced by mapping 3D points to the 2.5D grid structure, Himmelsbach *et al.* [97] proposed an improvement. It first organizes the points in both direction and range and creates a discretization of the range, associating the 3D points to a bin. Getting the point with the lowest height value from each bin, it applies an Incremental Algorithm [100] for line fitting and ground plane detection. All non-ground points are mapped to a 2D grid cell, but if there exists a large gap between the z-coordinates of two consecutive points (that no third point falls into), a 3D segmentation is performed. The algorithms were benchmarked alongside the RBNN [83], having better results in runtime values.

Douillard *et al.* [101] made an evaluation and proposed some segmentation methods, separating the analysis of sparse and dense data. It has empirically shown the benefit of ground extraction prior to object segmentation for dense data. The general approach used for this kind of data is a voxel grid based segmentation that relies on the local neighborhood. *Cluster-All with Variable Neighborhood* was the method that had the best trade-off in terms of simplicity, accuracy and computational times. It segments the non-ground points by local voxel adjacency, being less restrictive for points farther the sensor, as they become sparser. Due to the drawbacks of sparse data voxelization (many empty cells), it presents some ground models of non-constant resolution either providing a continuous probabilistic surface or a terrain mesh built from the structure of a range image. The use of Gaussian Process (GP) methods for having a probabilistic model of sparse terrain was explored in [102]. An iterative approach to those methods is the GP Incremental Sample Consensus (GP-INSAC) algorithm. This method relies on a set of offline learned parameters and is capable of processing data from any source or multiple fused sources, however, it assumes that there exist few outliers. In the mesh-based technique, after constructing the mesh, the ground points are extracted, based on the computation of a gradient field, and the remaining points are clustered using the *Cluster-All* method. Both of these sparse data segmentation methods provided close to real-time performance.

In order to add real-time capability to their previous works [103, 104], in [105] is made a comparison between the use of rectangular and radial grids centered on the sensor. Their segmentation method is based in minimum and maximum height maps difference calculation. Rectangular grids depend on a defined maximum range and horizontal and height resolutions. Radial grids are generated based on a maximum range

and angular, range and height resolutions. In the latter there is the need of stitching the image border blobs, as they correspond to neighbor spatial regions. However, in this analysis, the radial grid requires less cells and produces less fragmentation of the objects, when compared to the rectangular grid, which leads to a faster processing. The core of this approach is the use of smaller images instead of large point cloud data for processing. Radial grids were also used in [106] for segment labeling, and in [107] for ground segmentation.

In 2012, Choe *et al.* [109] proposed an improvement of the RBNN algorithm, like in [84], with the purpose of applying it in a real-time segmentation of urban environments. In order to be able to apply this method, the ground points need to be removed or might be clustered with other urban structures (the segmentation is only based on the ED of points). For that, all the points that lie on horizontal surfaces, detected by calculating the angle between two consecutive points, are potential ground points and will be removed from further calculations if lie on the larger cluster of horizontal points (ground is considered as a dense structure). The remaining points are then clustered regarding not a fixed [84] but a distance-varying radius. Using the continuous point information, the radius of acceptance is calculated based on the neighbor points distance and the sensor structure and then scaled by a predefined value to accept some possible measurement errors. This kind of approach fulfilled the real-time requirements and outperformed the fixed radius one. The work in [110] was based on Hidden Markov Model (HMM) [111] that used the angles between points of consecutive scans to classify them as horizontal, vertical and vegetation. Motivated by this work, in [108], the authors refined their previous method and extended the point classification of [110], adding the slope and invalid types (vegetation is here treated as scatter), simply thresholding measurement distances and angles of consecutive points. For non-horizontal points, the

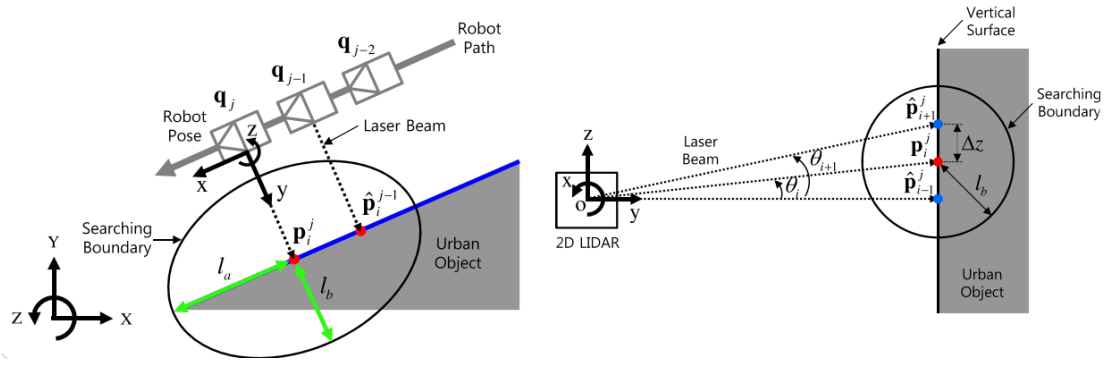


Figure 2.5: e-RBNN searching method (adapted from [108]).

segmentation is here based on an Ellipsoid model based RBNN (e-RBNN), that uses an ellipsoid rather a circular region (figure 2.5), defining a scaled radius based on the neighbor points distance, and another based on the sensor-to-point ED. The cluster classification is then performed using a Naive Bayes (NB)[1] model, as it allows a fast computation time to train or test, provides outputs with probabilities and can be trained with few examples.

Another region growing [112] based algorithm was introduced in [113]. The segmentation is dependent on the unevenness value that is calculated based on the difference between the expected and measured ranges. Thus, this method relies on the knowledge of the sensor's scan geometry. For segmenting the point cloud, it first removes the ground points, using an unevenness threshold, and then grows the remaining points as obstacles by setting an unevenness interval of acceptance, associated with a maximum range difference between neighbor points. Although this method explores the sensor geometry, its region growing approach makes it unsuitable for real-time applications, due to the high processing time.

Later, in 2017, was presented an approach of 3D segmentation more directed to autonomous ground vehicles [114]. It uses as input a 360 degrees coverage point cloud of the sensor and performs a two-step segmentation: it first extracts the ground based on a Ground Plane Fitting (GPF) followed by a clustering methodology named Scan Line Run (SLR). In the GPF step, the point cloud is divided into a number of segments, along the vehicle motion direction, and is assumed that the ground can be fitted to a plane and the points with the lowest height are more likely of belonging to the ground. For each segment, a set of seed points is selected and all the points are classified as ground or not, based on an OD threshold, being the ground ones used for the refinement of the ground plane estimation. For treating the non-ground points is used an adaptation of the work in [115], called SLR (figure 2.6). For each ring of the scan, the consecutive non-ground points are grouped into runs with an associated label. For the subsequent rings runs', the label can be inherited from the previous if the ED between the neighbor points is below a threshold. If any label needs to be merged, the one with the lowest value is kept and the other is replaced during the final pass of the algorithm through the remaining point cloud. Although it showed to be applicable in real-time, the GPF was only tested using even terrain, so its behavior when encountering rough, uneven terrain and rapid slope changes was not evaluated.

Other existing works also use image color information from a camera [116] or Extruded Surface of Cross Sections (ESCSs) [117] for segmenting 3D point clouds.

[1] `http://www.statsoft.com/textbook/naive-bayes-classifier`

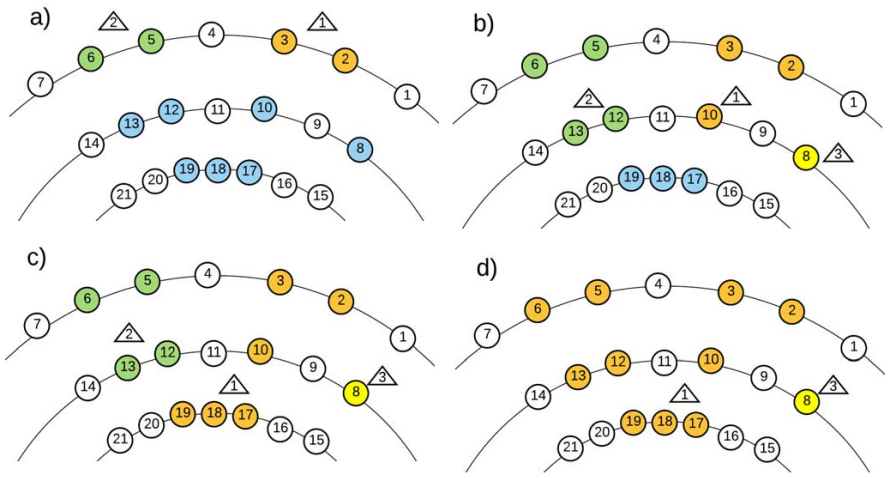


Figure 2.6: The four stages of SLR clustering algorithm. Circles represent points and triangles report the cluster labels (adapted from [114]).

2.1.2 Line Extraction

The use of LiDAR sensors to provide a 3D model of the power lines is useful, not only to evaluate its mechanical conditions but also to detect and monitor some possible dangers, like encroaching objects [14,17,32,118]. This data can also be used to evaluate the risks associated with the conductor blowout on certain wind conditions or to predict the ones related to the vegetation growth. Therefore, this modeling can provide a multifaceted analysis of power line maintenance and risk evaluation, saving time and costs [119].

In 2004, Melzer and Briese [120] introduced a new method to reconstruct power lines using airborne LiDAR data. After filtering the terrain and some possible outliers and vegetation, it applies a Hough Transform (HT) [121] to extract lines on the horizontal plane projected points. Due to its sensitivity to the remaining outliers and the inhomogeneity of cell's point density, an iterative version of HT is applied instead. After grouping the detected lines, their model is estimated by a catenary fitting with parameters obtained from randomly selected line primitives based on RANdom SAmple Consensus (RANSAC) [122].

Based on the assumption that only trees, power lines, and building edges generate a significant height difference between the first and last pulses of LiDAR, in [123] is presented a stochastic method for separating trees and power lines in an urban environment. The point cloud is preprocessed to extract buildings.

Another classification technique is presented in [124, 125] for clearance anomalies detection, by separating the points into ground, power lines, towers (or pylons) or other

objects. It also starts to consider real-time operation using a sweeping 2D LiDAR attached to the helicopter fuselage, acquiring at a rate of 2 Hz. Using global positioning, the power lines are modeled by a polynomial fitting that allows interpolating segments of power lines not detected by taking into account previous geometrical information.

In [29] is proposed a helicopter point cloud segmentation method more specific for power lines extraction. It identifies three types of data: power lines, vegetation and surfaces; using an ellipsoid-based neighborhood along the direction of flight, as it assumes that the flight is parallel to the power lines. Based on the eigenvalues of the covariance matrix of each cluster, a classification is attributed and used for training an algorithm. Each cluster classified as possible power line is then specified by its mean point and the largest eigenvalue, which comprises a local affine model. The line span is then extracted by fitting the power line to a straight line, on the horizontal plane, and a catenary, on the vertical plane, along that direction. Due to the assumption of a flight aligned with the power lines, this method will likely fail to detect perpendicular or transversal power lines.

The work of Jwa *et al.* [126] presents a voxel segmentation technique to classify voxels as linear, non-linear or undefined. This classification is performed by meeting trained parameters of the HT, eigenvalues, and point density combination. In order to have an initial orientation, for each power line candidate point is performed an eight-hypothesis CLF (figure 2.7), selecting the one with the least sum of the squared residuals between hypothesis and observations. Applying an outlier testing and removal, the power line direction is calculated with the member points and a sub-cubic box, containing them, is generated and used as *a priori* information for the next steps. The proposed Voxel-based

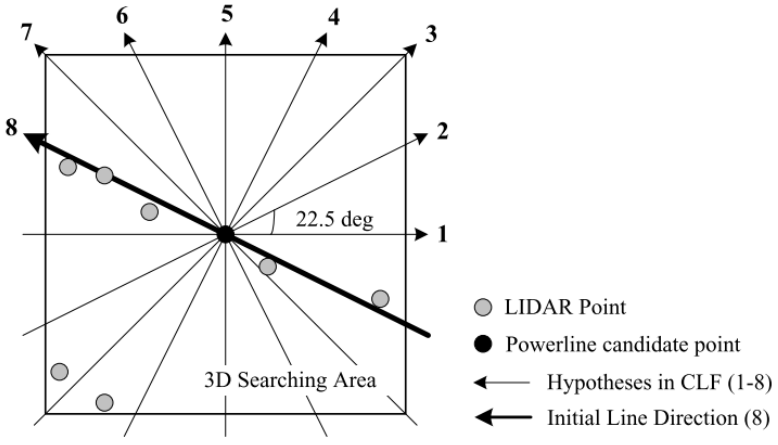


Figure 2.7: Eight-hypothesis CLF [126].

Piece-wise Line Detector (VPLD) process will propagate the sub-cubic boxes based on the previous information of the estimated catenary model, updating it with the new information. If the number of points is not enough to model a catenary, a line equation is used instead.

In [127] is used a semi-automatic method to get multiple power line models. From an airborne LiDAR point cloud data, the power line points are manually selected and used as input for the algorithm. The points are then clustered to their nearest neighbor, if a distance threshold is not overpassed, and a polynomial model is used for the power line fitting.

Zhu and Hyyppä [30] proposed a method for extracting lines in forest areas. It first selects candidate points based on their height difference and density distribution on each cell from the grid. Those points are then projected on the horizontal plane forming a binary image. Image processing algorithms are then applied to extract the power lines, taking into account aspects like continuity, binary blob area, and linearity. The 3D line is then recovered from the resulting 2D blob, however, no fitting model is presented.

In 2014, Xiang [128] presented a work-flow for the 3D reconstruction of power lines. In his work was approached the importance of a proper filtering of data in the quality of the final results. The newly presented vertical spacing followed by an improved density-based filtering outperformed the combination of the adaptive TIN and 8-neighbor culling mechanism. Applying the HT to the filtered data, projected on the horizontal plane, are obtained some line segments that will be concatenated by similarity (likely to belong to the same wire; figure 2.8). Some possible remaining vegetation points are then removed on the power line extraction and clustering, using a more restrictive vertical spacing filtering. The power lines are then fitted and reconstructed by using a line model for the

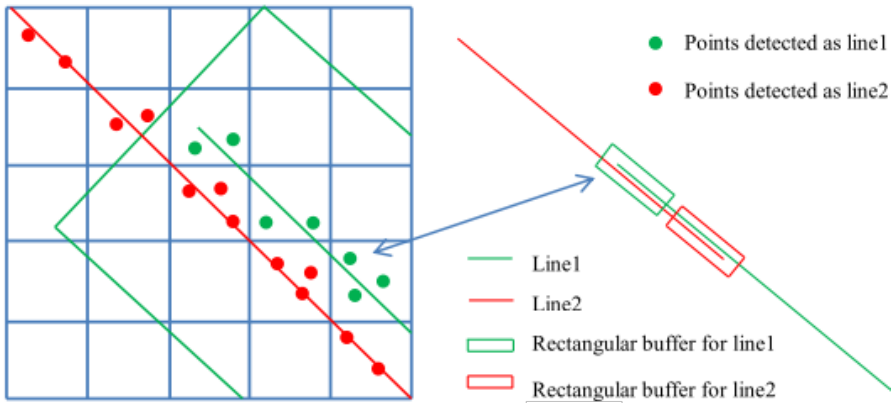


Figure 2.8: Rectangular buffer to concatenate lines detected by HT (adapted from [128]).

horizontal plane, and a catenary for the vertical one. The hazard tree detection is also discussed, however, only vertical hazards are effectively detected.

Adapted for ground vehicle LiDAR data, in [129] the initial point cloud is voxelized and filtered according to terrain clearance, up-down continuity, and feature eigenvector. Analyzing the neighbor voxels, the point density can also be used to remove the remaining non-power line points. A HT is then applied to the horizontally projected points for detecting the power lines. A cluster-growing method based on local straight line fitting is used to reach the final fitting to a polynomial. The local line fitting properties are also used for cluster recovery from occlusions.

In [38] is proposed another method suited for ground vehicles. It extracts off-road points by removing the curb ones, aided by *a priori* knowledge of the road size. The extracted points are then sequentially segmented by height, spatial density, and size and shape filters. The power line candidate points are extracted by using HT and then clustered based on their ED. The spatial density and size and shape filters require a voxel-based division of the data and are mainly used to detect pylons. The result of the mathematical fitting of the power lines to a line on the horizontal plane and a catenary on the vertical plane can then be used to aid the pylon detection. Another method for pylon detection is presented in [130] for aerial vehicles.

Guo *et al.* [31] presented a method for power lines reconstruction that classifies the point cloud data into power line, vegetation, building, ground, and pylon, based on their previous work [131]. In order to prevent a false-positive in the classification of vegetation as pylons, it is verified if the pylons are connected by wires, using a HT. The power line points that belong to a span between two neighbor pylons are segmented into profiles orthogonal to the span direction and their similarity is checked (figure 2.9). Using a RANSAC-like method for setting the initial parameters of the catenary curve and searching for candidate samples, the span is iteratively reconstructed and the points are added to the estimation if they satisfy the so far predicted model. Due to the assumption that, over the same span, the power lines model have approximately the same parameters, this method outperformed the one in [29] on situations with power line partial occlusion or data sparseness (up to a certain degree).

An automatic clearance anomaly detection algorithm was developed in [12]. It first filters the terrain and then detects the existing pylons for isolating the power line spans and perform an individual analysis. The resulting power line points are then segmented into clusters, combining geometric distribution analysis and conditional Euclidean clustering with linear feature constraints inside the divided spans. The resulting clusters are fitted into a horizontal linear and a vertical catenary model. Comparing the power

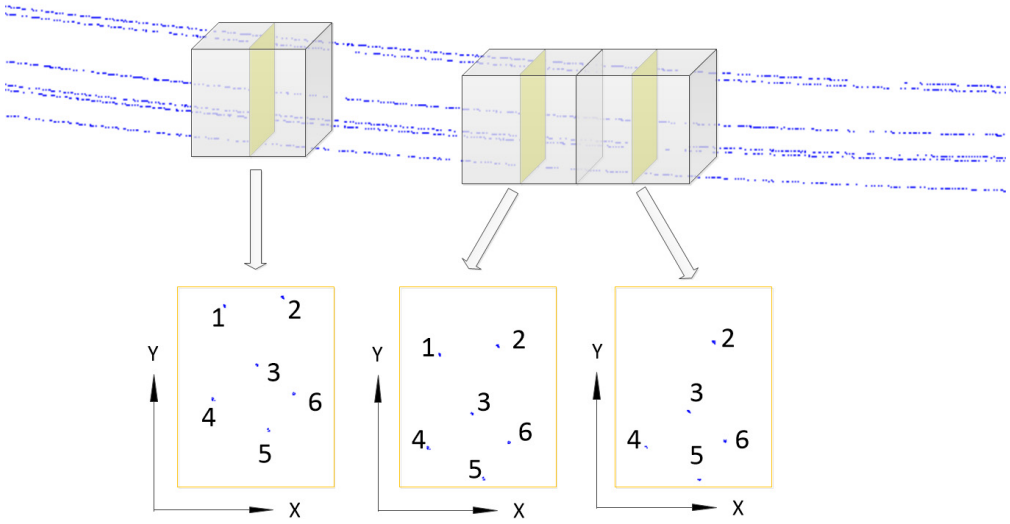


Figure 2.9: Similarity detection by span segmentation (adapted from [31]).

lines model with the previous filtered terrain and vegetation, the algorithm is capable of detecting clearance anomalies on the line corridor.

Regarding the power line model fitting, Jaw *et al.* in [16] studied the effect of the wind on the conductors' motion while they are acquired by LiDAR sensors.

2.2 Discussion

After exposing the existing works related to the segmentation of point clouds and extraction of power lines techniques, an overall analysis will help in the development of the present project. Some of the enumerated techniques had already detected some limitations related to the segmentation and power line detection, proposing solutions to overcome them. Both those informations and some assumptions made and proven in the existing works can be either improved or directly applied to this project.

Regarding the 3D segmentation of an input point cloud, it is common to use a voxel grid division, as all the spatial information of the points is kept, unlike the 2.5D grids or the plane projected images. This division of the environment in cells allows an easier (neighboring information) and quicker (clustering) analysis of the recorded structures. When the data is being treated for each complete scan of the LiDAR (360 degrees revolution), a sensor-centered radial grid is preferred. This fact is based on the common nature of the 3D LiDAR sensors, as they acquire the scene in a radial fashion and the data becomes sparser with the distance to the sensor. The radial grids are capable of

dealing with this data sparseness due to the larger spatial coverage of the farthest cells, preventing some over-segmentation or meaningless cells (with too few points). This non-constant spatial coverage generally leads to a smaller number of generated cells when compared to the traditional rectangular grids and, thus, to a faster processing.

The basis of the point cloud segmentation on mobile robotics is the division of ground and non-ground points. As the ground can play a role of a link between other structures, its removal is beneficial because not only provides a deeper segmentation of the point cloud but also greatly reduces the number of points to analyze, as the ground is usually a structure that contains a large number of points. The speedup obtained with ground removal may, however, not be sufficient to reach real-time processing. For reaching it, the existing works had proved the need for LiDAR sensor data structure knowledge. This prior information vastly decreases the number of operations needed, due to the knowledge of the points' neighborhood.

After the segmentation, the classification of the obtained clusters is mainly done by using offline trained algorithms.

In the power line extraction techniques there is an effort to first isolate the candidate to power line points from the others. This division is made based on the covariance matrix eigenvalues of neighbor points and the point density. The resultant points are then projected into a horizontal plane to generate a binary image and detect lines or pylons based on the size and shape of the blobs, and to detect lines by applying a HT. The detection of the pylons position helps in the span-by-span analysis, where power lines similarity can be used. The power line fitting models used are generally a line on its horizontal projection, and either a catenary or a polynomial for the vertical plane. This models, however, can be affected in the presence of considerable wind [16]. Having the fitted lines, clearance anomalies can be detected by analyzing the shortest distance to a non-power line point.

In the presented methods, only in [124, 125] is slightly considered the real-time operation for power line extraction. However, as it uses a 2D LiDAR at a low rate, consequently having a small number of points to analyze, this method not suitable for larger amounts of data at a higher rate. The studied power line extraction methods also depend on several trained thresholds that can lead to a deterioration of the presented results for other point cloud examples.

Using multirotor Unmanned Aerial Vehicles (UAVs), the ground estimation for subsequent removal is trickier. For ground vehicles, is easier to detect, with high confidence, some ground points that can be used as seeds for region growing algorithms. As the aerial vehicle's operation is independent on the ground nature and can be done in both

urban and rural environments, the ground may have a lot of variations and be similar to, for example, the top of buildings. One challenger is the fact that the aerial vehicles can easily change its altitude, changing the point density of the detected structures below them, which leads to situations where the ground might not be the cluster with more points.

This page was intentionally left blank.

Chapter 3

Fundamentals

In this chapter is made a brief overview of some concepts that are considered important for helping in the understanding of the developed work. The exposed information is related with mechanical features of the sensors, mathematical concepts and middleware architecture.

3.1 LiDAR

The rangefinder Light Detection And Ranging (LiDAR) is a sensor used to measure distances to objects around it. Its operation is based on the Time-of-Flight (ToF) of a fired pulse of light. For many years, these LiDAR sensors were either spinning or flash, but, lately, their technology is evolving and begin to appear new ones, like the MicroElectroMechanical System (MEMS) mirror and phased-array LiDARs [132, 133].

3.1.1 Spinning LiDAR

In its basic configuration, a spinning LiDAR makes measurements by periodically firing a light beam. Adding a moving mirror towards the fired laser beam, and spinning the moving structure of the sensor (figure 3.1), is possible to obtain a Three-Dimensional (3D) map of the environment around the sensor [132–134].

This LiDAR configuration is able of sensing in a 360 degrees Field-Of-View (FOV) and can emit the laser source at a higher power level than a stationary one, achieving longer ranges [134]. This last feature is related with the eye safety rules: as the spinning LiDAR only fires in a particular direction at each moment, the additional power won't risk the eyes.

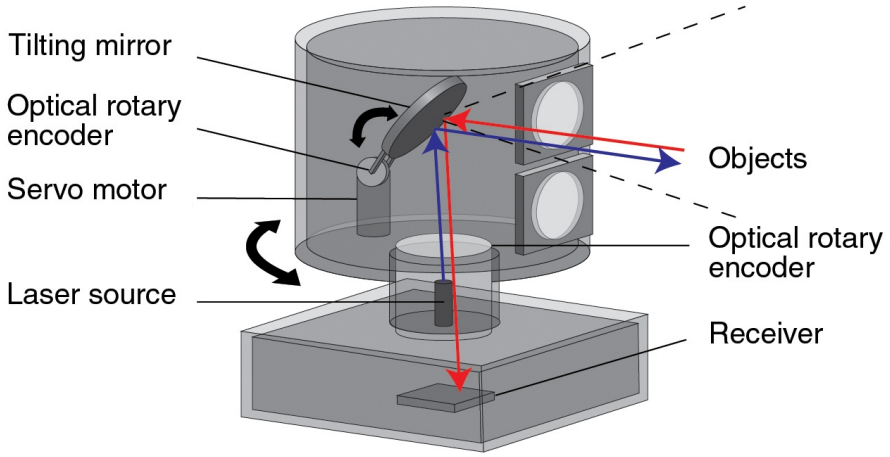


Figure 3.1: Spinning LiDAR example [135].

Despite the advantages, these sensors are usually bulky, mechanically fragile to shocks and vibrations, and expensive, easily reaching several thousands of euros [133,134]. The lifetime of their mechanical parts is low (usually up to 2000 hours) [132].

3.1.1.1 Convert measures to spatial points

A beam corresponding point P is returned by a 3D spinning LiDAR in a polar coordinate system $P = (\alpha, \omega, r)$, where α is the azimuth angle, ω the elevation angle, and r the measured range (refer to figure 3.2). For converting it to an euclidean 3D sensor-centered frame, the following equations must be applied:

$$x = r \cdot \cos(\omega) \cdot \sin(\alpha) \tag{3.1}$$

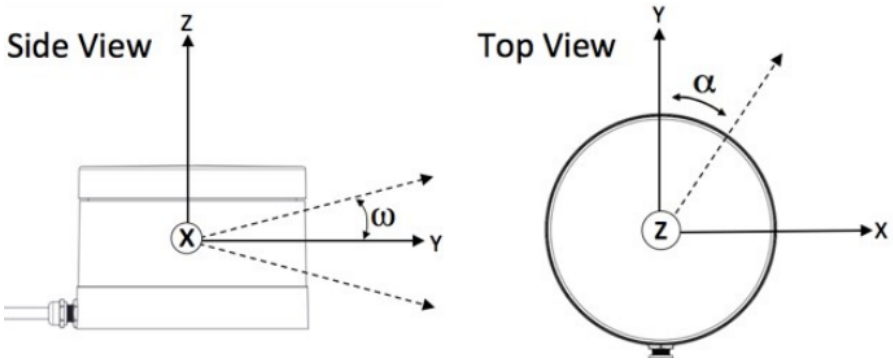


Figure 3.2: Spinning LiDAR coordinate system example (adapted from [136]).

$$y = r \cdot \cos(\omega) \cdot \cos(\alpha) \tag{3.2}$$

$$z = r \cdot \sin(\omega) \tag{3.3}$$

Having in mind that every point is measured at a different time, if the sensor or the detected objects are not stationary, special care shall be taken to correct or interpret the obtained data.

3.1.2 MEMS mirror LiDAR

In the MEMS mirror LiDAR, the highly-precise mechanical scanning mirrors of the spinning LiDARs are replaced by spinning MEMS micro-mirrors to perform the transmitting part by directing the beams [137] (figure 3.3).

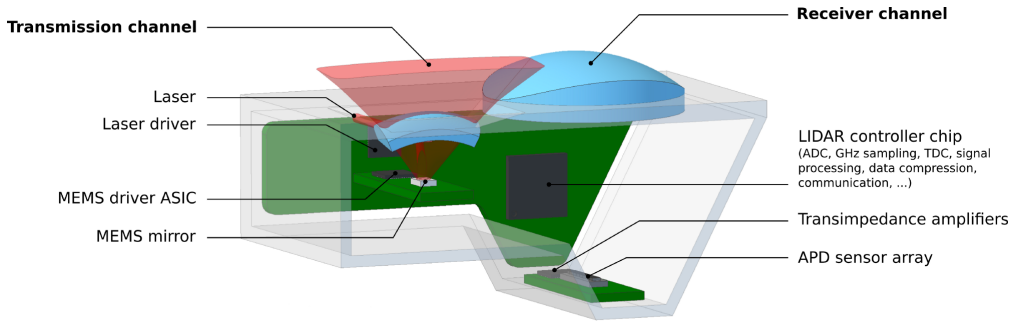


Figure 3.3: MEMS mirror LiDAR example [137].

This huge reduction on the mechanical parts allows a huge shrinking on the LiDAR size, significantly lowering its cost and sensitivity to vibrations [137]. Despite this, it can still be susceptible to shock and vibrations and mirror drifts out of alignment, needing to be recalibrated. Big changes in temperature may also lead to an uncalibrated system [132].

Another drawback of this LiDAR is its FOV of 120 degrees or less, but due to its lower cost, a complete 360 degrees coverage system can be built by using multiple sensors [134].

3.1.3 Solid-state LiDAR

Solid-state LiDARs are sensors that have no moving pieces. Their designs are being developed to solve the cost, size, reliability and complexity of mechanical-based LiDARs [133].

3.1.3.1 Flash LiDAR

This kind of LiDAR periodically fires a powerful beam of light towards a lens that will spread it to the environment. The reflected light is then captured by an array of photo-sensors. As only a small portion of light returns, there is the need for very sensitive (and expensive) receivers, what makes the cost of these sensors reach the hundreds of thousands of euros [132]. Its main advantage is the complete FOV acquirement at one moment, however, the FOV coverage is generally very small.

3.1.3.2 Phased-array LiDAR

In a phased-array, there exists an array of a large number of optical antennas synced up in a specific way. By controlling their phase it is possible to form a radiation pattern, or spot, that has a certain size and is pointed in a certain direction [132] (figure 3.4).

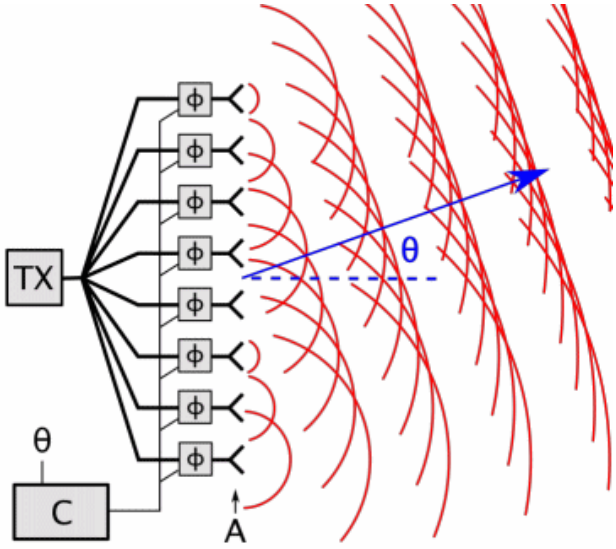


Figure 3.4: Phased-array LiDAR steering concept (adapted from [132]).

These systems are more economical interesting than the others (less than a thousand euros) [132], but tend to produce beams that diverge more, making it hard to achieve a combination of long range, high scanning resolution, and wide FOV [134].

3.2 3D Frame Relations

When a sensor is attached to a mobile robot or a global reference is needed for navigation, there is the need of a homogeneous transformation (**M**) between all the reference

coordinate systems. The homogeneous transformation matrix ($\mathbf{M}$) is given by [138]:

$$\mathbf{M_{4\times4}} = \begin{bmatrix} & \mathbf{R_{3\times3}} & & \mathbf{t_{3\times1}} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.4}$$

Where $\mathbf{R}$ is the rotation matrix that maps the angular relation between the two frames and $\mathbf{t}$ is the translation between the two coordinate systems' origin.

Having the figure 3.5 as an example, the point P can be mapped into the global frame (w), knowing its coordinates in the local frame (b), by applying the following equation:

$$\begin{bmatrix} P_w \\ 1 \end{bmatrix} = \begin{bmatrix} & \mathbf{R}_b^w & & \mathbf{t}_w^b \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} P_b \\ 1 \end{bmatrix} = \mathbf{M}_b^w \cdot \begin{bmatrix} P_b \\ 1 \end{bmatrix} \tag{3.5}$$

Being:

- $P_w = [x_w \quad y_w \quad z_w]^T$ the point P coordinates in the global frame;
- $\mathbf{R}_b^w$ the rotation matrix from the local frame to the global frame;
- $\mathbf{t}_w^b$ the origin of the local frame denoted in the global frame coordinates;
- $P_b = [x_b \quad y_b \quad z_b]^T$ the point P coordinates in the local frame;
- $\mathbf{M}_b^w$ the homogeneous transformation matrix that maps the local frame coordinates into the global frame.

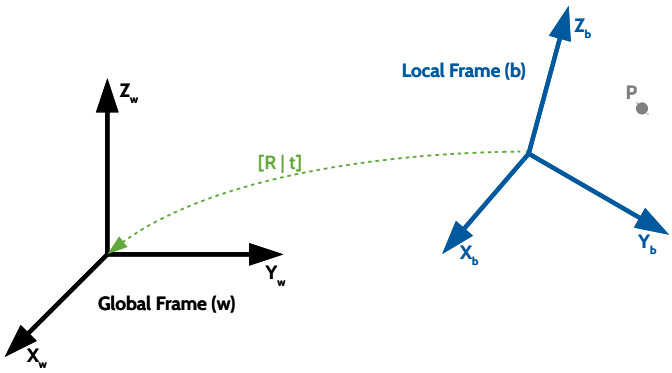


Figure 3.5: Conversion of reference coordinate frames.

A 3D rotation matrix $\mathbf{R}$ can be obtained by the Euler angles (roll (ψ), pitch (θ) and yaw (ϕ)) [139] and results from the ordered sequence of rotations around the three frame axes:

$$\mathbf{R} = \mathbf{R}_x(\psi) \cdot \mathbf{R}_y(\theta) \cdot \mathbf{R}_z(\phi) \tag{3.6}$$

Where a rotation of an angle ψ around the x-axis is:

$$\mathbf{R}_x(\psi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\psi) & -\sin(\psi) \\ 0 & \sin(\psi) & \cos(\psi) \end{bmatrix} \tag{3.7}$$

The rotation of an angle θ around the y-axis is defined as:

$$\mathbf{R}_y(\theta) = \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{bmatrix} \tag{3.8}$$

And, finally, rotating an angle ϕ around z-axis:

$$\mathbf{R}_z(\phi) = \begin{bmatrix} \cos(\phi) & -\sin(\phi) & 0 \\ \sin(\phi) & \cos(\phi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.9}$$

To complement the explanation above, in figure 3.6 are depicted the Euler angles in the specific case of an Unmanned Aerial Vehicle (UAV) body frame.

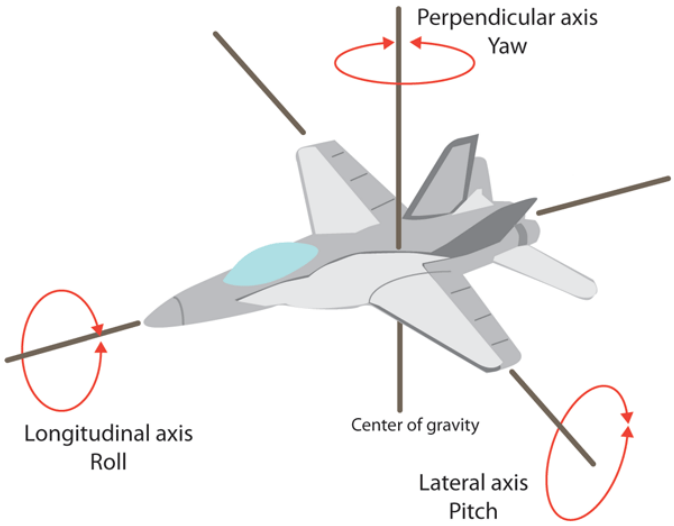


Figure 3.6: UAV Euler angles representation (adapted from [140]).

3.3 Eigenvalues and Eigenvectors

Eigenvalues are a special set of scalars associated with a linear system of equations (matrix equation) [141, 142]. Each eigenvalue λ has a corresponding eigenvector. Their determination is extremely important in engineering, where it is equivalent to matrix diagonalization and arises in applications like system's stability analysis.

Having a matrix $\mathbf{A}$ that represents a linear transformation, if there is a vector $\mathbf{v} \in \mathbb{R}^n \neq \mathbf{0}$ that satisfies

$$\mathbf{A}\mathbf{v} = \lambda \cdot \mathbf{v} \tag{3.10}$$

For some scalar λ, then λ is an eigenvalue of $\mathbf{A}$ with the corresponding eigenvector $\mathbf{v}$.

Simplifying equation 3.10:

$$(\mathbf{A} - \lambda \cdot \mathbf{I})\,\mathbf{v} = \mathbf{0} \tag{3.11}$$

Where $\mathbf{I}$ is the identity matrix.

The eigenvalues λ can then be obtained by solving:

$$\det(\mathbf{A} - \lambda \cdot \mathbf{I}) = |(\mathbf{A} - \lambda \cdot \mathbf{I})| = 0 \tag{3.12}$$

Substituting the calculated eigenvalues λ in equation 3.11 are obtained the associated eigenvectors $\mathbf{v}$.

3.4 3D Line Fitting Using Covariance Matrix

A line in $\mathbb{R}^3$ is defined by a point $\mathbf{r}_0 = [a\,b\,c]^T$ on the line and a direction $\mathbf{r} = [r_x\,r_y\,r_z]^T$ that is parallel to it. The set of points of this line comprises the equation 3.13 (vector equation of the line) [143]:

$$[x\,y\,z]^T = \mathbf{r}_0 + t \cdot \mathbf{r}\ ,\ t \in \mathbb{R} \tag{3.13}$$

Considering a set of N data points in $\mathbb{R}^3$, organized in a matrix $\mathbf{X}_{N\times 3}$, with an arithmetic average of the variables $\mu_X = [\overline{\mathbf{x}_1}\ \overline{\mathbf{x}_2}\ \overline{\mathbf{x}_3}]$, the element i of the unbiased data matrix $\mathbf{X}_c$ is given by:

$$\mathbf{X}_{\mathbf{c}i} = \mathbf{X}_i - \mu_X \tag{3.14}$$

The unbiased covariance matrix of $\mathbf{\Sigma}_X$ is then calculated using [144–146]:

$$\mathbf{\Sigma}_X = \mathbf{X}_c^T \mathbf{X}_c \tag{3.15}$$

The resultant covariance is expressed by a square 3x3 matrix with the eigenvalues (λ_1,λ_2,λ_3) and the eigenvectors ($\mathbf{v}_1$,$\mathbf{v}_2$,$\mathbf{v}_3$) can be calculated using the method described in section 3.3. The eigenvectors indicate the directions through which the data is dispersed, and the associated eigenvalues represent their relative importance.

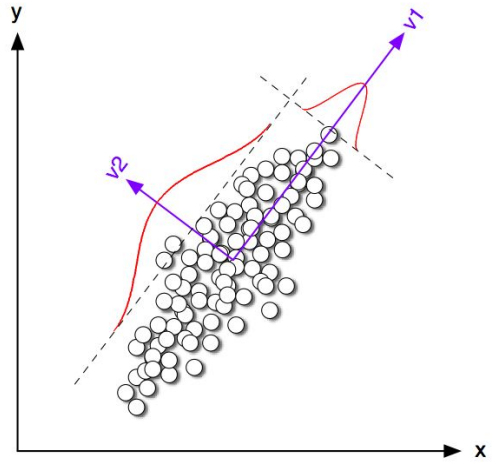


Figure 3.7: Eigenvectors of 2D data [147].

Assuming that the eigenvalues are ordered by their importance ($\lambda_1 > \lambda_2 > \lambda_3$), if $\lambda_1 \gg \lambda_2 \approx \lambda_3 \approx 0$, it means that the data can be approximated to a 3D line, and dimensionally reduced to one dimension, along the direction of the line, given by $\mathbf{v}_1$. If only $\lambda_3 \approx 0$, the data propagates into two dimensions and can be approximated to a plane with a normal defined by $\mathbf{v}_1 \times \mathbf{v}_2$. Otherwise, it is not possible to apply a reduction to the data's dimension.

When the line fitting is possible, the estimated parameters are easily obtained. The fitted line will pass through the center of the data points, having the direction given by the most relevant eigenvector:

$$\mathbf{r}_0 = \mu_X \tag{3.16}$$

$$\mathbf{r} = \mathbf{v}_1 \tag{3.17}$$

3.5 Minimum Distance Between 3D Lines

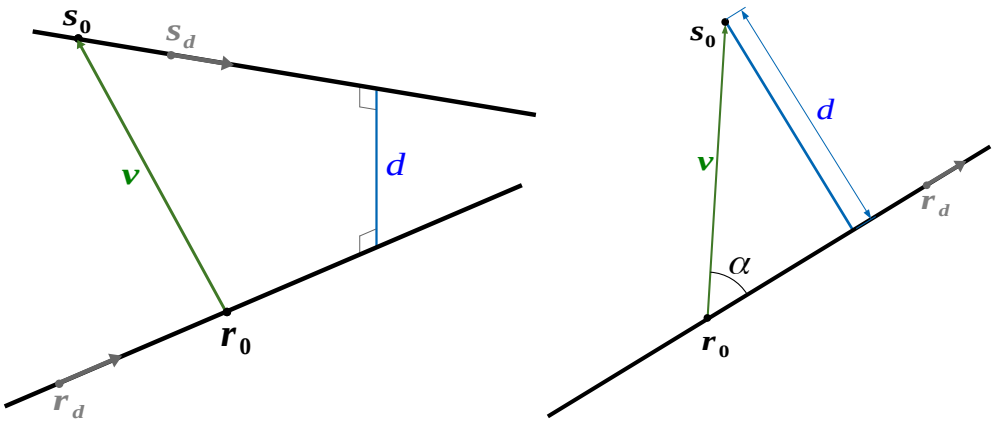
Having two 3D lines, $\mathbf{r} = \mathbf{r}_0 + t_r \cdot \mathbf{r}_d$ and $\mathbf{s} = \mathbf{s}_0 + t_s \cdot \mathbf{s}_d$, the minimum distance between them is given by the magnitude of the vector that connects a point in line $\mathbf{r}$ to a point in line $\mathbf{s}$ and is orthogonal to both lines [148, 149].

Taking two points from the lines ($\mathbf{r}_0$ and $\mathbf{s}_0$), the vector $\mathbf{v}$ connecting them is defined

by $\mathbf{v} = \mathbf{s}_0 - \mathbf{r}_0$, and the distance between the lines is the component of $\mathbf{v}$ that is orthogonal to lines $\mathbf{r}$ and $\mathbf{s}$. That direction can be found by calculating the normalized cross product of lines directions ($\mathbf{r}_d \times \mathbf{s}_d$). Applying the dot product with $\mathbf{v}$, the minimum distance d will be the absolute value of the result (equation 3.18).

$$d = \left| \frac{\mathbf{r}_d \times \mathbf{s}_d}{\|\mathbf{r}_d \times \mathbf{s}_d\|} \cdot \mathbf{v} \right| \tag{3.18}$$

Note that equation 3.18 is only valid to skew lines[1], once that if the lines are parallel ($\mathbf{r}_d = \mathbf{s}_d$), the norm of their cross product is zero ($\|\mathbf{r}_d \times \mathbf{s}_d\| = 0$).



(a) Skew lines minimum distance. (b) Point to line distance

Figure 3.8: Minimum distance representation.

Parallel lines have the same distance at any point belonging to them. This property allows to consider the calculation of the minimum distance between them as a point to line distance problem: taking a point $\mathbf{s}_0$, in line $\mathbf{s}$, calculate its minimum distance to line $\mathbf{r}$. From figure 3.8(b) is possible to visualize that $d = \|\mathbf{v}\| \cdot \sin\alpha$. As the cross product of $\mathbf{v}$ with $\mathbf{r}_d$ is $\mathbf{v} \times \mathbf{r}_d = \|\mathbf{v}\| \cdot \|\mathbf{r}_d\| \cdot \sin\alpha$, the derivation of equation 3.19, that provides the distance, is straight-forward.

$$d = \left\| \frac{\mathbf{v} \times \mathbf{r}_d}{\|\mathbf{r}_d\|} \right\| \tag{3.19}$$

[1]Two or more non-coplanar lines which have no intersections but are not parallel, also called agonic lines. Since two lines in the plane must intersect or be parallel, skew lines can exist only in three or more dimensions (`http://http://mathworld.wolfram.com/SkewLines.html` [Accessed: 11-Oct-2018]).

3.6 Catenary Curve Model

By definition, a catenary is a curve formed by a wire, rope, or chain hanging freely from two points that are not in the same vertical line and forming a U shape. The power lines are generally modeled as a catenary curve that is defined by a hyperbolic cosine function [29, 51]:

$$y = a + c \cdot \cosh\left(\frac{x-b}{c}\right) \tag{3.20}$$

Where a and b are the translation values from the origin along the y and x-axis, respectively, and c is a parameter that affects the catenary scale.

As equation 3.20 is a transcendental equation[2] in c, it must be solved numerically. This means that a set of previously known points belonging to the catenary curve is needed. In [29] is exposed an already proven theorem for calculating the parameters:

Theorem 1 *For a catenary with parameter $a = 0$, let (x', y') be any point on the catenary, and let* $\mathbf{t}$ *be the tangent vector at (x', y'). The minimum distance from* $\mathbf{t}$ *to the point (x', y') will equal the catenary parameter c.*

Figure 3.9 helps theorem 1 understanding and provides further equations' notation.

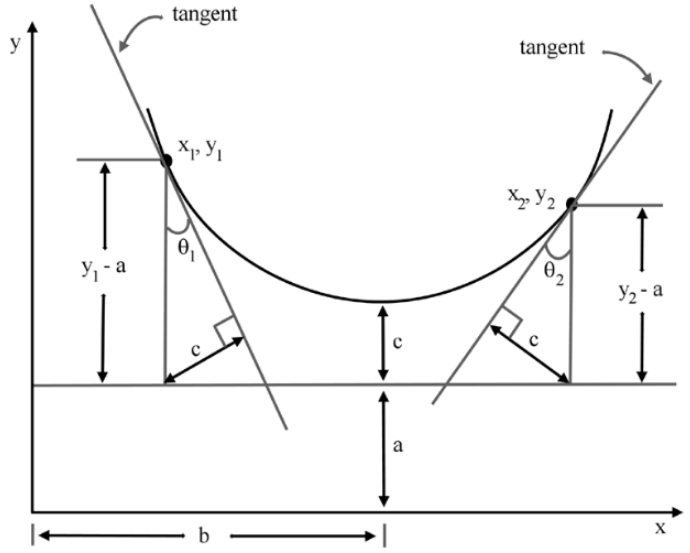


Figure 3.9: Computing catenary parameters (a,b,c) [29].

[2] An equation or formula involving transcendental functions: a function which is not an algebraic function. In other words, a function which "transcends," i.e., cannot be expressed in terms of, algebra (`http://mathworld.wolfram.com/TranscendentalFunction.html` [Accessed: 30-Aug-2018]).

From the figure, c is easily computed in terms of a, y_1 and θ_1, using elementary geometry:

$$c = (y_1 - a) \cdot \sin(\theta_1) \tag{3.21}$$

And in terms of a, y_2 and θ_2:

$$c = (y_2 - a) \cdot \sin(\theta_2) \tag{3.22}$$

Combining 3.21 and 3.22, resolving in a:

$$a = \frac{y_2 \cdot \sin(\theta_2) - y_1 \cdot \sin(\theta_1)}{\sin(\theta_2) - \sin(\theta_1)} \tag{3.23}$$

Substituting a in 3.21 or 3.22, c can be computed as well as b:

$$b = x_1 + c \cdot \operatorname{arccosh}\left(\frac{y_1 - a}{c}\right) \tag{3.24}$$

Another odd property of the catenary curves [150] is that, if $a = 0$, the ratio of the area under a catenary curve to the arc length is independent of the interval over which they are measured and is equal to the parameter c.

3.7 Robotic Operating System (ROS)

ROS [151, 152] is a modular open-source software framework for robotic applications that is becoming increasingly-widely used in the robotics research community. Its implementation comprises a distributed communications infrastructure, low-level drivers for a wide variety of sensors and actuators, a set of development and visualization tools, and a large collection of robotics-specific algorithms. ROS communication between processes is based on a Peer-To-Peer (P2P) topology, being language-neutral (it supports very different languages, like `C++`, Python, Octave, and LISP). This framework has a rich set of online resources[3] and their source code for packages is typically hosted in public repositories[4].

A ROS system usually comprises a number of independent processes, called *nodes*, that communicate with each other through a *publish-subscribe* mechanism. The communication is done by passing typed *messages* over *topics*. *Nodes* can be either used as device drivers, being connected directly to hardware, or to compute the data they

[3] `http://ros.org`
[4] https://github.com

subscribe to and publish the result. ROS also provides support for synchronous and asynchronous remote procedure calls (called *services* and *actions*, respectively).

The P2P communication between the *nodes* is established by a central broker agent, the *roscore*: when a new *node* starts, it connects to *roscore* and lists both the *topics* that it will publish and those that it wants to subscribe to; the *roscore* passes back the information of the *nodes* publishing the desired *topics*, allowing the new and existing *nodes* to make the connection. *Roscore* is also capable of working with *nodes* distributed across a Local Area Network (LAN) (figure 3.10).

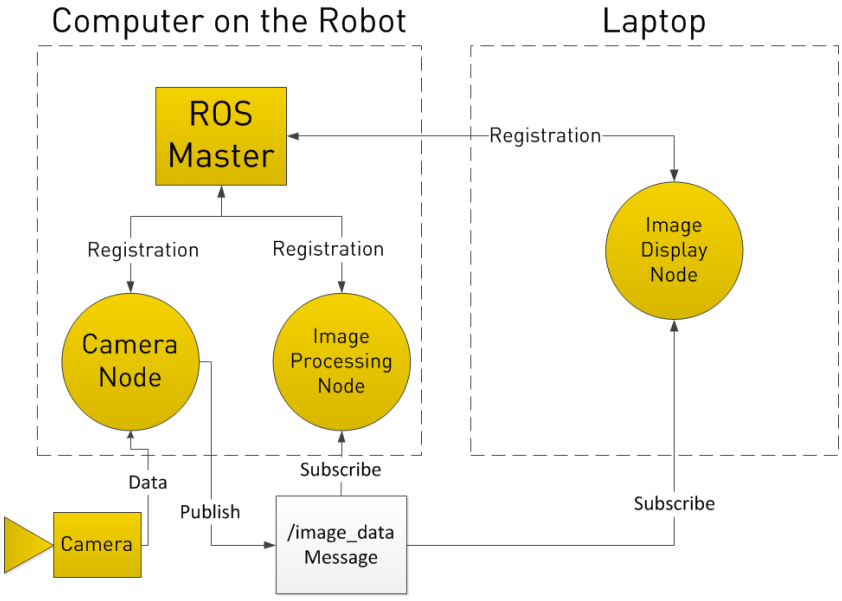


Figure 3.10: ROS high-level architecture using a LAN [153]. Circular boxes correspond to *nodes*, light square box is a *topic*, and the *roscore* is represented by the darker square box.

Besides *nodes*, ROS affords the use of *nodelets*. They behave similarly to *nodes* but have the benefit of zero-copy cost between *nodelets* of the same *nodelet handler* [154].

Concerning the code organization, individual *nodes* are collected into *packages*, which are themselves grouped into thematic *meta-packages*, containing an explicit list of meta-information that provides, among other things, the *package* dependences for build, run and test. This type of organization encourages the collaborative development, as it facilitates the *package* migration between systems and developers.

Chapter 4

System Design

The use of multirotor Unmanned Aerial Vehicles (UAVs) for electrical assets inspection has several advantages already discussed in chapter 1, but it comes with the cost of requiring some perception sensors. Those can be used to acquire data for the inspection purposes, but also to feed some collision avoidance algorithms or to perform some predefined autonomous maneuvers. The acquisition of a complete 360 degrees 3D point cloud for extracting power lines can be done by using a spinning Light Detection And Ranging (LiDAR) sensor (see section 3.1).

This chapter contains an overview of the hardware and software architectures defined for applying the developed algorithm. Taking advantage on the ROS modular structure, the system's software is based on this middleware, in order to use available ROS packages and favor its integration in other possible future applications.

4.1 Hardware architecture

The developed algorithm is intended to be applied in a multirotor UAV, using a 3D point cloud provided by a LiDAR sensor as input, during an electrical power assets inspection. The onboard computer will be responsible for the algorithm processing, using an architecture as the one presented in figure 4.1.

Nowadays LiDAR sensors can easily provide several hundreds of thousands of points each second. Given this large amount of data, in the UAV system, the LiDAR sensor needs to establish a communication with the onboard computer through an ethernet connection (figure 4.1, in red), or other that allows a high transmission rate. The computer first configures the sensor and then processes all the incoming stream of data.

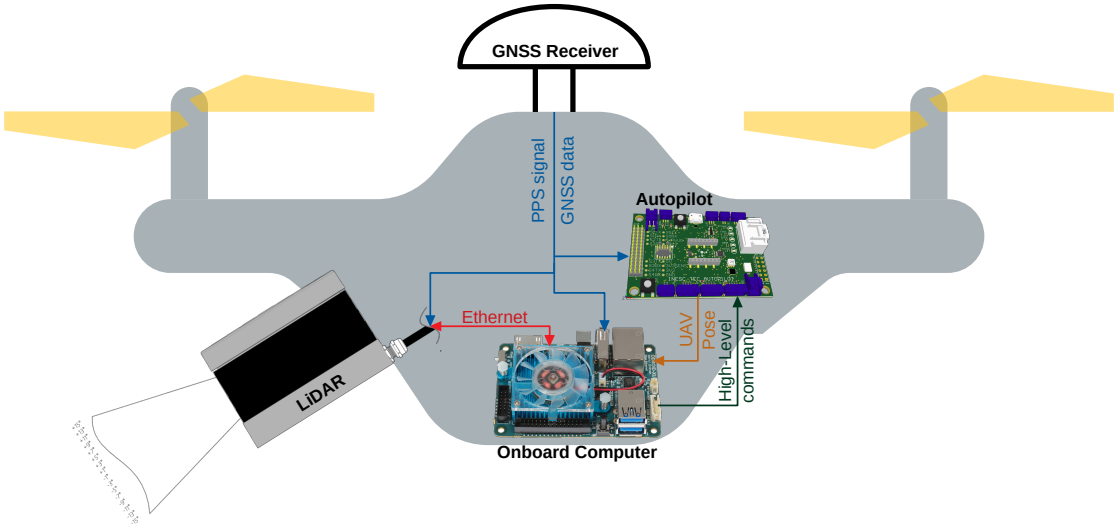


Figure 4.1: High-level hardware architecture (some graphics adapted from [155–157]).

Having a Global Navigation Satellite System (GNSS) receiver is possible to synchronize all the subsystems' data timestamps, using Coordinated Universal Time (UTC) as reference. The receiver provides a Pulse Per Second (PPS) signal and GNSS data (figure 4.1, in blue) that are used to synchronize the LiDAR sensor's data timestamp, the system time of the onboard computer, and the estimated inertial data from the autopilot. In the autopilot, GNSS data is also used for estimating the UAV pose, by fusing it with the Inertial Measurement Unit (IMU) measurements.

4.2 Software architecture

The main objective of point cloud segmentation is to provide an easier perception of the surrounding environment by creating clusters and detect features using application-specific algorithms. In figure 4.2 is exposed the high-level software pipeline to segment a point cloud generated by a LiDAR sensor. Apart from the input data, all the pipeline components were developed with the ROS framework.

The estimated UAV pose is provided to the computer's ROS environment (figure 4.1, in orange) by means of a topic published by the *mavros*[1] ROS package. This topic can be used to establish relations between the sensor, UAV, and global coordinate frames using the *tf2*[2] ROS package, that tracks the available frames' relations (equation 3.4)

[1] `http://wiki.ros.org/mavros`
[2] `http://wiki.ros.org/tf2`

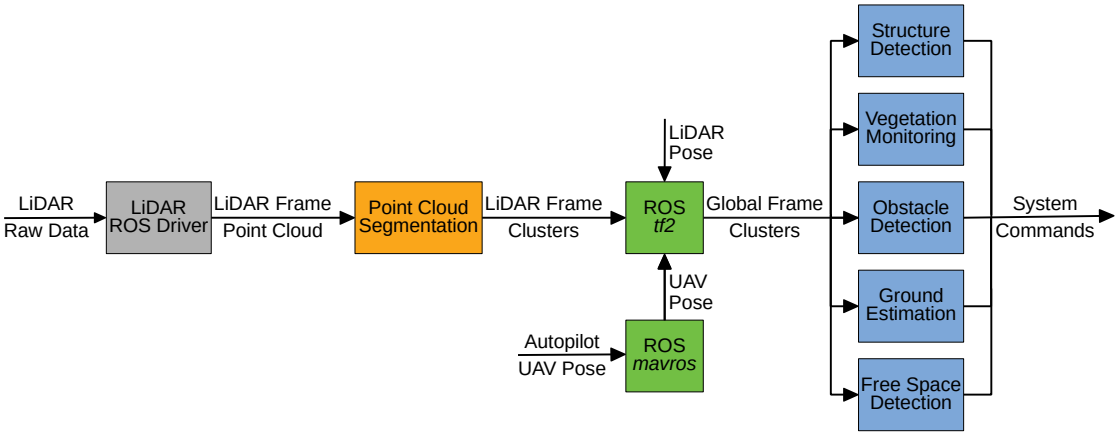


Figure 4.2: High-level software pipeline.

over the time. The LiDAR pose can be either a defined fixed relation with respect to the UAV, if it is fixed to the platform, or a dynamic relation, if attached to a gimbal, for example. In the last case, the LiDAR pose can be defined in a similar way as the UAV's.

Point cloud segmentation can be done at different levels, depending on the desired resolution and features, and the available time. Usually, the algorithms developed for the segmentation have a limited and predefined number of object classes to classify the detected structures. Those classes are defined according to the application requirements. In mobile robotics applications, the ground estimation is generally desired, not only to identify clear paths (ground vehicles) or landing spots (aerial vehicles) but also to simplify further processing of the remaining points (see section 2.2).

Using the remaining clusters properly referenced into a global frame, other algorithms, like obstacle's or free space detection, can be applied. The detection of obstacles is important to avoid collisions, but the free space recognition is also necessary for navigation and to identify unknown areas. Knowing those areas may be needed for inspection or mapping surveys. Structure detection or vegetation monitoring are application-specific algorithms. For power lines inspection, the classification of the objects into power lines, pylons and vegetation may be necessary to detect faults and potential hazards. All of those algorithms can feed a control system of the UAV either by sending high-level commands to the autopilot (figure 4.1, in dark green) or controlling actuators, like releasing some object near to a structure.

In this project is expected to develop a perception ROS package able to detect and estimate power line 3D information, providing meaningful data for subsequent algo-

rithms. The development can be assisted by the use of some already available and stable ROS packages. This fact, however, does not reject the possibility of implementing some changes to those packages' operation, in order to perform better over some defined project requirements.

Having a power line detection algorithm with real-time processing capability is useful for the creation of online maps that can be used on the UAV control layer. Achieving faster processing requires the knowledge of the LiDAR sensor structure. On 3D spinning LiDARs, due to the large amount of data provided, the azimuth angular resolution is usually high. Therefore, even when attached to a mobile frame, the pose difference is neglectable between two consecutive azimuth measurements, which allows to directly establish comparisons between them, not needing to map it to the global frame. This strategy allows processing time saving and position error reduction, as the UAV pose estimation (and associated errors) is not used.

Due to the above-mentioned properties, the point cloud segmentation can be performed using the provided points in the LiDAR frame, as depicted in figure 4.2. The resultant clusters can then be mapped to the global frame, in order to aid the structure recognition or visualize structured maps.

Chapter 5

PL^{2}DM Algorithm

This chapter details the algorithm developed to fulfill the objectives and requirements addressed in the dissertation, the Power Line LiDAR-based Detection and Modeling (PL^{2}DM). It starts by presenting the concept in which the algorithm is based, following with the detailed description of each data processing block. The complete processing results on the detection and mathematical modeling of the power lines.

5.1 Concept

In order to achieve the main objective of detecting the power lines in a point cloud, the process needs to be divided into several steps: the segmentation of an input point cloud, the identification of candidate power line segments, and finally, the estimation of the power line fit based on the mathematical model.

5.1.1 Segmentation

This is the most critical step to achieve a fast detection of the power lines. Due to a large amount of data provided by Light Detection And Ranging (LiDAR) sensors, its analysis needs to be a real-time and optimized step in order to ensure that will not introduce a huge delay in the overall power line detection.

For time-saving purposes, the evaluation of each point received can be made with respect to the LiDAR's reference frame, not requiring to transform them into a global frame. Adding to this, when possible, an analysis that relies on the values of range and directions in relation to the sensor (polar coordinates), instead of one based in points mapped into the Euclidean space, is preferable.

Even if has low accuracy, a first classification of the obtained clusters needs to be made at this step. The classification is critical for the performance of subsequent processing, as the non-meaningful clusters are ignored. This reduces the data quantity and allows a deeper (and, probably, more time-consuming) analysis.

5.1.2 Line Detection

The input for this layer is the points that were previously classified as potential power lines, mapped into a global frame. Since that the quantity of data is, in principle, much less than the initial input point cloud, here the points are submitted to another clustering process. This process is responsible for refining the input clusters based on collinearity properties. This procedure assumes that, in each LiDAR scan, the detected fragments of power lines can be approximated to a line. Each cluster containing lines is passed to a further layer, ignoring others with low evidence of collinearity. From the clustering refinement, there may exist some isolated points that are not considered as lines, however, they are not removed for the next steps.

5.1.3 Power Line Modeling

The mathematical modeling of the power lines is the last step of the algorithm. This layer is responsible for saving the line clusters provided by the previous layer and iteratively trying to estimate power lines model in the global frame. The isolated points can possibly be associated with previously modeled lines that have a high degree of confidence.

The output generated by this layer may be useful to other algorithms running in the vehicle, like the ones related to the navigation, not only to consider a power line as an obstacle, but also as a possible target to inspect or follow.

5.1.4 Algorithm Architecture

The figure 5.1 summarizes the concept of the PL^{2}DM architecture. Whenever a point cloud (${}^lPCL$) provided by the LiDAR is available, the segmentation layer is triggered. If there exists any evidence of the existence of lines, the candidate line points (${}^lL_p$) are provided to the next layer.

Based on the frame relations, described in section 3.2, the points ${}^lL_p$ are mapped from the local LiDAR frame (l) into the global frame (w). These relations are stored by the Robotic Operating System (ROS) package *tf2* that receives and tracks information about the pose of the robot in the global frame (${}^wp_b$) and the pose of the LiDAR sensor in the robot frame (${}^bp_l$).

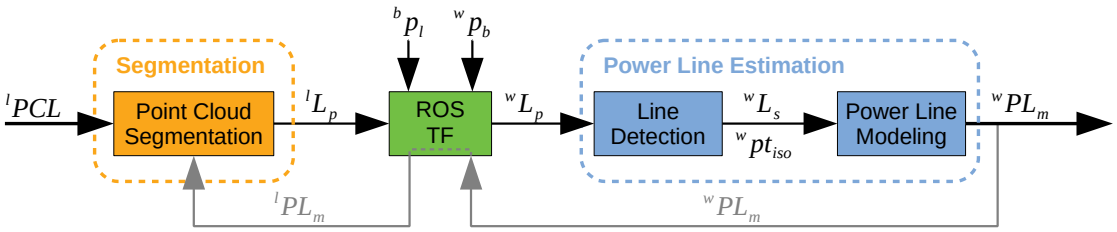


Figure 5.1: Concept of the proposed algorithm architecture.

The line detection is then performed over the candidate points in the global frame (${}^{w}L_p$). The clusters that comprise the collinearity constraints (${}^{w}L_s$), or with isolated points (${}^{w}pt_{iso}$), are passed to the power line modeling module. It tries to match the new information with the one previously stored, trying to model the detected power lines (${}^{w}PL_m$).

The information of the obtained models can then be used to feed other algorithms. Moreover, mapping those models in the local frame (${}^{l}PL_m$) is still of great importance for the next segmentation procedures, as they may help in the classification of the clusters.

5.2 Algorithm Procedure

During the development of the PL^{2}DM algorithm, were created some rules related to the input data organization and power line properties. Therefore, the algorithm might fail if some of those rules are not respected.

5.2.1 Segmentation

The segmentation of a point cloud is the first step of the PL^{2}DM. Once that it is the closest layer to the sensor data input, a correct definition of the organization of the data is crucial for its proper functionality.

5.2.1.1 Required Data Organization

For analysis, the input point cloud is represented by a Two-Dimensional (2D) matrix, where each column represents an azimuth angle α, and each line an elevation angle ω (figure 5.2, following the nomenclature of figure 3.2). Due to time constraints, the stream of data provided by the LiDAR is assumed to be acquired (and sent) sequentially in a raster-scanning order (orange arrow in figure 5.2), being used the same order for the data analysis.

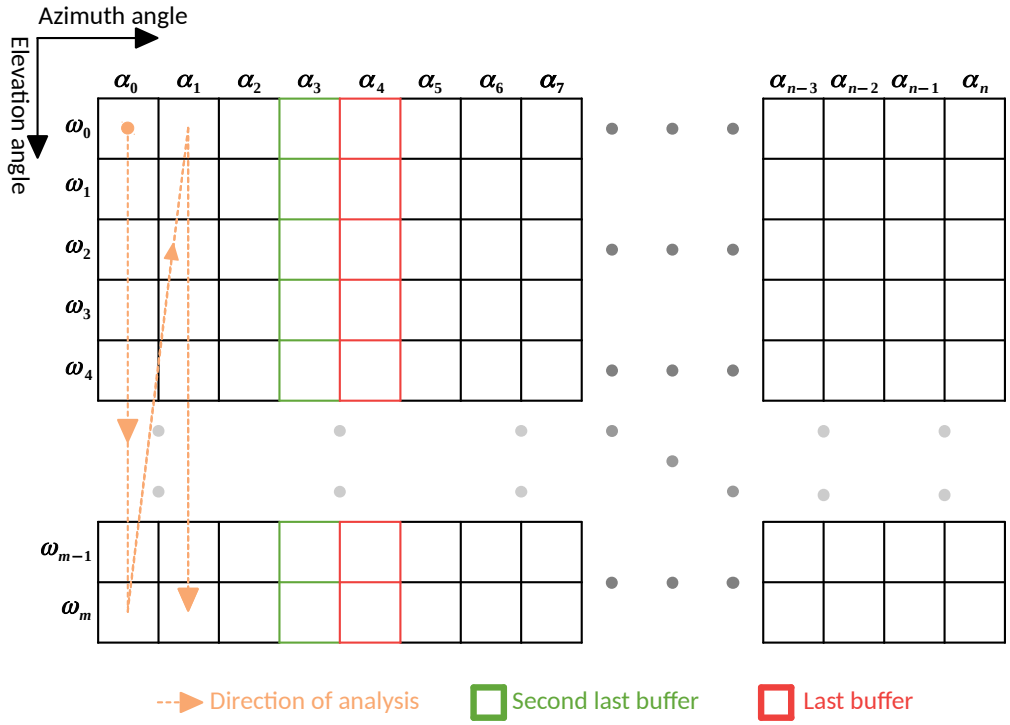


Figure 5.2: LiDAR data organization.

During the data processing, some already processed points are kept into memory to perform a neighbor comparison based on the layout presented in figure 5.3. Those points are saved into two buffers: one of them contains the last processed points for each elevation angle, and another contains the second last processed points (red and green cells in figure 5.2, respectively).

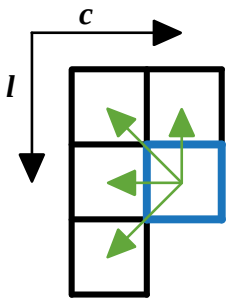


Figure 5.3: Layout for neighbor comparison.

The layout chosen for neighbor comparison (figure 5.3) is the one that takes the least iterations to compare all the points in the data matrix, considering the direction of

analysis. Each point being processed (blue cell in figure 5.3) is compared against four neighbors (top and left ones) unless it belongs to the first column, first line, or last line of the data matrix. In this case, it is only evaluated with the available neighbors that respect the layout.

5.2.1.2 Adaptive Neighbor Comparison

Whenever the input point cloud is dense, the data matrix cells are all properly filled, and the neighbor comparison is performed between contiguous cells. In figure 5.4 is evidenced the importance of the second last points buffer (bold green cell in figure 5.4) to perform the point evaluation.

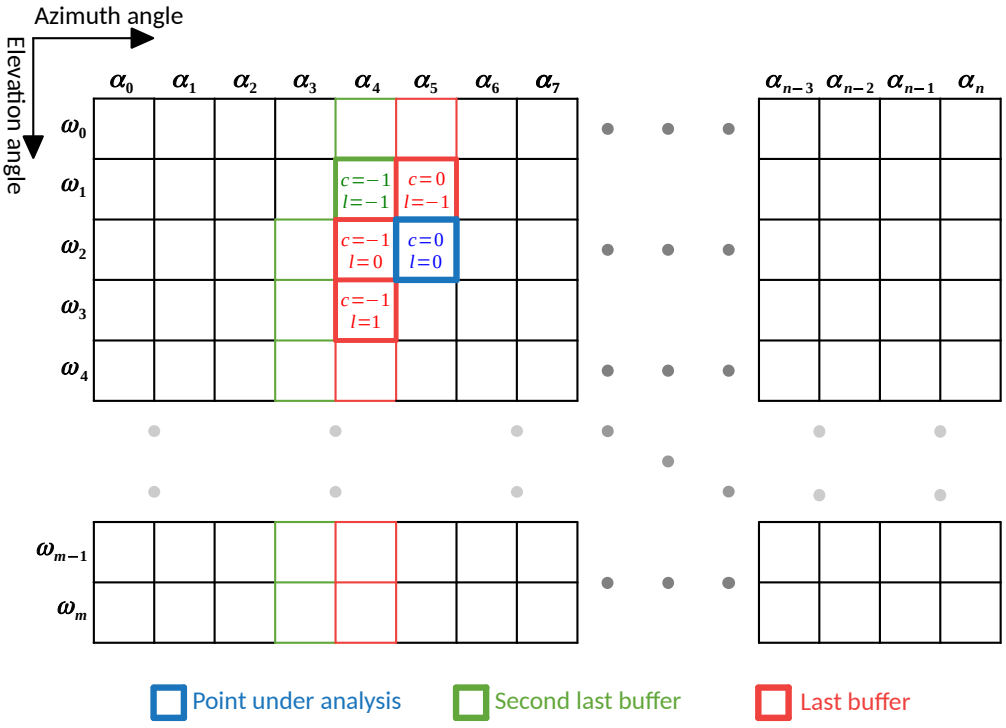


Figure 5.4: Point analysis procedure.

In cases where the point cloud is sparse, i.e., it might have some invalid points (from invalid ranges), the layout for neighbor comparison can be adaptive to the data matrix. Figure 5.5 depicts the matrix representation of a point cloud with invalid points.

When analyzing a point, if some of its contiguous neighbors corresponds to an invalid point, the comparison can be made with a non-contiguous one. In this case, the value of $\Delta\alpha$ needs to be evaluated and, if the value is below a threshold, the point is compared

with the non-contiguous neighbor, otherwise, it is assumed that the point has no neighbor for that position.

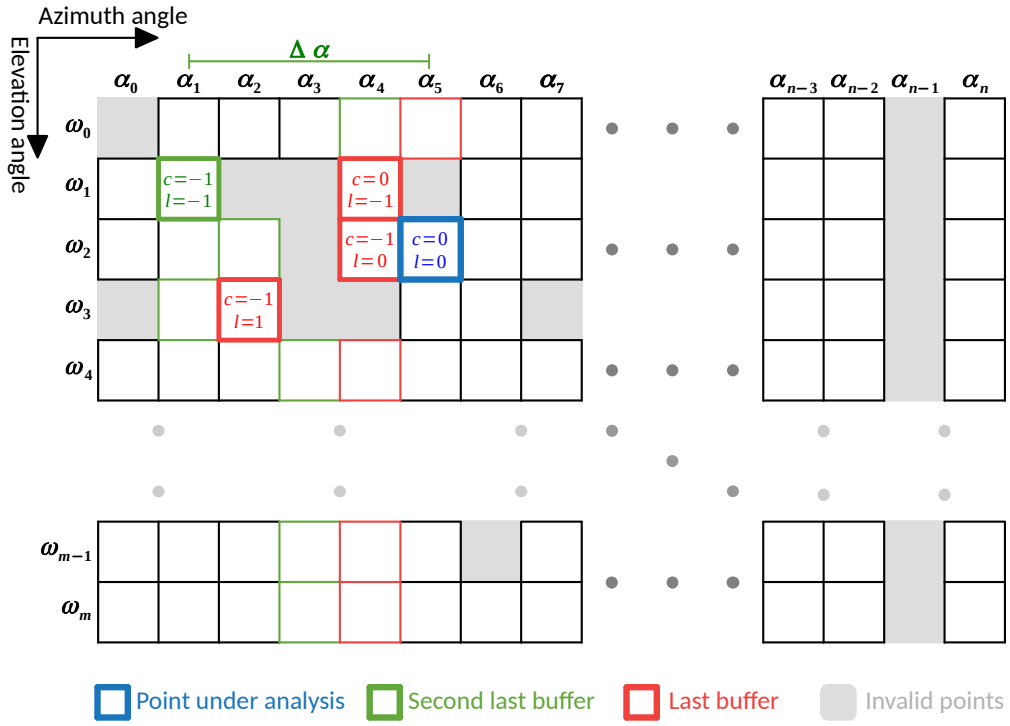


Figure 5.5: Point analysis procedure with a sparse data matrix.

Comparing figures 5.4 and 5.5 is possible to notice how the layout of neighbors is adapted in the presence of invalid cells. This adaptation represents an effort to approximate a sparse data matrix to a dense analysis, reducing the effect of possible over-segmentation generated by sensor acquisition failures.

5.2.1.3 Expected Range Calculation

The proposed algorithm relies on a range-based segmentation, comparing the true and expected ranges. Besides the range value, all the input points shall have azimuth and elevation angles associated. As already pointed in subsection 3.1.1, those three values are the ones that are usually provided by a spinning LiDAR, which means that the algorithm has the capability of interpreting data directly from the sensor, not needing any preprocessing of the values.

The segmentation of a scene consists in clustering the data points that share some defined properties. In a scene with some structures, it is usual to find planar features,

so the coplanarity among the points can be a property to consider for clustering. Other features like the distance between consecutive points or the angle between them can also help the division.

Due to their radial measurement nature, the points acquired with a LiDAR become sparser with the range, which makes the clustering by distance threshold tricky. Among other coexistent methods, in [158,159] is used an Adaptive Breakpoint Detection (ABD) method for dividing the clusters of points. It consists in defining an incidence angle λ of the previous beam with a virtual line that passes through its point (p_{n-1}), calculating the maximum distance D_{max} allowed to cluster the current point (p_n) with the previous, using the law of sines [160] (figure 5.6). The defined λ can be interpreted as the worst case incidence angle for a line on which points can be reliably detected.

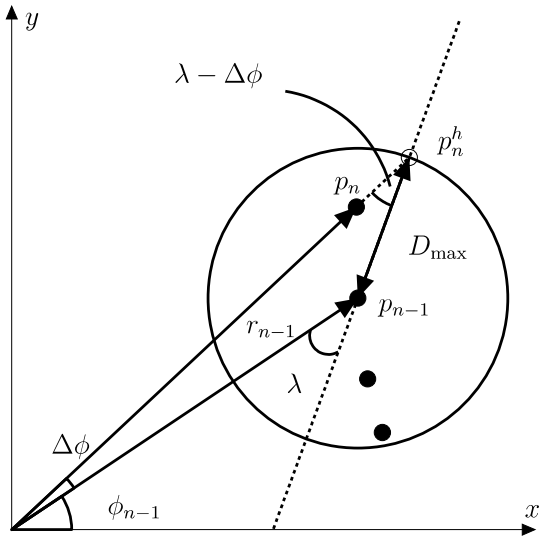


Figure 5.6: ABD threshold D_{max}.

The drawback of this approach is that the value of λ is predefined and remains unchanged regardless of the actual incidence angle of the measured surface. Therefore, its correct tunning is critical, as the quality of the segmentation is strongly dependent on it. Adding to this, it lacks on the definition of a relation between the previous and the current range values, not validating if the current range should be higher or lower than the previous. Once satisfied the D_{max} constraint, the points are clustered, which can lead to clusters with significant changes in angular relations.

To overcome some of the identified limitations, the proposed algorithm uses a method that is capable of calculating the expected range relation of neighbor beams, taking into account the estimated plane (detailed in 5.2.1.4 and 5.2.1.5) of the cluster where the

point belongs. This way, the calculated range relation will consider simultaneously the distance and angle between the points.

For obtaining the range relation between beams, the analysis is separated into two parts: one that considers the vertical displacement (Z's direction), due to differences in the elevation angle, and another related to the horizontal displacement (XY plane), associated with the azimuth angle. The vertical analysis is made by having as reference the intersection line of the plane formed by all the beams through a fixed azimuth angle, with the estimated plane (Π) of the cluster (figure 5.7). The horizontal analysis is similar, but instead is considered the plane generated by some beams over a fixed elevation angle (figure 5.8).

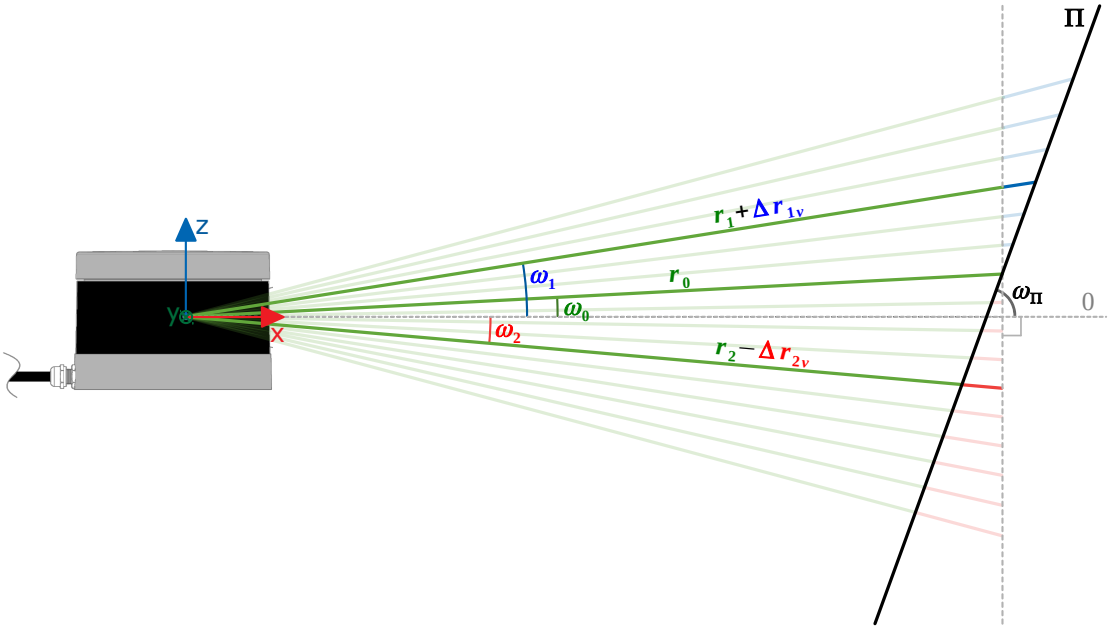


Figure 5.7: Vertical measurements of a LiDAR.

Taking figure 5.7 as reference for the further equations, it is possible to infer how the range of neighbor beams are related. Considering two beams with range r_0 (reference) and $r_1 + \Delta r_{1v}$ (neighbor), and the intersection line with the plane Π, the equations of the 2D lines, in the sensor's frame are given by:

$$z_0 = m_0 \cdot x_0 + b_0 \tag{5.1}$$

$$z_1 = m_1 \cdot x_1 + b_1 \tag{5.2}$$

$$z_\Pi = m_\Pi \cdot x_\Pi + b_\Pi \tag{5.3}$$

As the lines representing the beams pass through the reference, $b_0 = b_1 = 0$. The value of b_Π can be obtained by the intersection of equations 5.1 and 5.3, where $x_\Pi = x_0$:

$$b_\Pi = m_0 \cdot x_0 - m_\Pi \cdot x_0 = (m_0 - m_\Pi) \cdot x_0 \tag{5.4}$$

Now taking the intersection between equations 5.2 and 5.3, where $x_\Pi = x_1$:

$$x_1 = \frac{(m_0 - m_\Pi)}{(m_1 - m_\Pi)} \cdot x_0 \tag{5.5}$$

Substituting, in equation 5.5, $x_0 = r_0 \cdot \cos(\omega_0)$, $x_1 = (r_1 + \Delta r_{1v}) \cdot \cos(\omega_1)$, $m_0 = \tan(\omega_0)$, $m_1 = \tan(\omega_1)$, $m_\Pi = \tan(\omega_\Pi)$:

$$(r_1 + \Delta r_{1v}) \cdot \cos(\omega_1) = \frac{(\tan(\omega_0) - \tan(\omega_\Pi))}{(\tan(\omega_1) - \tan(\omega_\Pi))} \cdot r_0 \cdot \cos(\omega_0) \tag{5.6}$$

Knowing that $\tan(a) - tan(b) = \frac{\sin(a-b)}{\cos(a) \cdot \cos(b)}$, the final equation that calculates the range relation is given by:

$$\frac{(r_1 + \Delta r_{1v})}{r_0} = \frac{\sin(\omega_0 - \omega_\Pi)}{\sin(\omega_1 - \omega_\Pi)} \tag{5.7}$$

Generalizing equation 5.7 for a reference range r_{ref} and a neighbor range r_{nb}, in the vertical plane (r_{nbver}):

$$ver_{scale} = \frac{r_{nbver}}{r_{ref}} = \frac{\sin(\omega_{ref} - \omega_\Pi)}{\sin(\omega_{nb} - \omega_\Pi)} \tag{5.8}$$

The approach for the XY plane is similar to the above, and the horizontal range relation is given by:

$$hor_{scale} = \frac{r_{nbhor}}{r_{ref}} = \frac{\sin(\alpha_{ref} - \alpha_\Pi)}{\sin(\alpha_{nb} - \alpha_\Pi)} \tag{5.9}$$

Combining the two results, having a reference range, the expected neighbor range will be obtained by solving:

$$r_{nb} = hor_{scale} \cdot ver_{scale} \cdot r_{ref} \tag{5.10}$$

When performing a comparison between a point and its neighbor, if the cluster where the neighbor belongs is suitable for being approximated to a plane Π, ω_Π and α_Π are calculated based on its normal, otherwise, it is assumed that $\omega_\Pi = \pi/2$ (orthogonal to XY) and $\alpha_\Pi = \alpha_{ref} + \pi/2$ (perpendicular to the reference beam). After calculating them, the angular values are constrained to the interval $\omega_\Pi, \alpha_\Pi \in [0, \pi[$.

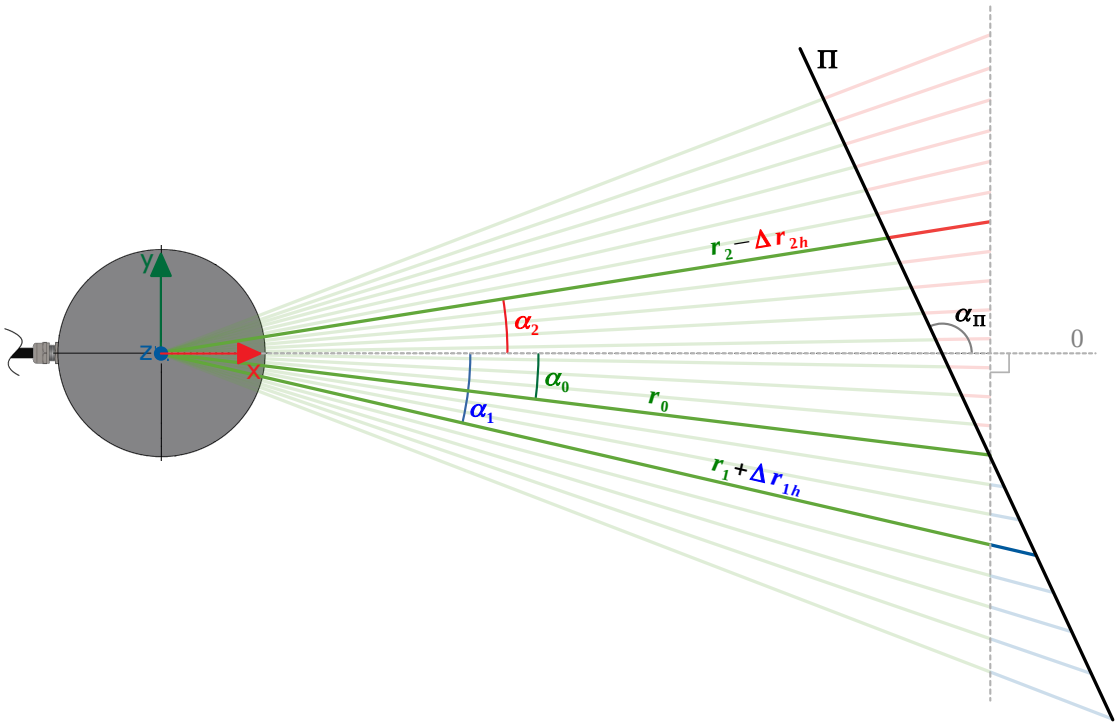


Figure 5.8: Horizontal measurements of a LiDAR.

An important note to this method is the fact that equations 5.8 and 5.9 can have a null denominator, so they must be used with special care. For example, in equation 5.8, this happens when $\omega_{nb} = \omega_\Pi$ or $\omega_{nb} = \omega_\Pi \pm \pi$. Even when those equalities are avoided, a very small value of the denominator will rapidly increase the value of ver_{scale}, which means that the relation between ω_{nb} and ω_Π needs to be limited by setting a maximum to the allowable value ver_{scale}. The algorithm will consider the calculated value of r_{nb} not valid if it overpasses the maximum range of the LiDAR.

Another physical limitation that is applied to this approach is the fact that the value of a range is always positive. To ensure this, the values of ver_{scale} and hor_{scale} are only accepted if they are positive, so their numerator and denominator must be both positive or negative, which corresponds to, in the case of the ver_{scale} value, $\omega_{ref}, \omega_{nb} \in]\omega_\Pi, \omega_\Pi + \pi[$ or $\omega_{ref}, \omega_{nb} \in]\omega_\Pi - \pi, \omega_\Pi[$, respectively. These constraints are represented in the top row graphs of the figure 5.9, in blue. As the comparison is performed between neighbor beams, ω_{ref} and ω_{nb}, and α_{ref} and α_{nb}, have similar values, so these limits will likely not be respected only in cases where their values are close to ω_Π and α_Π.

Whenever some of the presented constraints is not satisfied, the values of ω_Π and α_Π are set to their default, $\omega_\Pi = \pi/2$ and $\alpha_\Pi = \alpha_{ref} + \pi/2$.

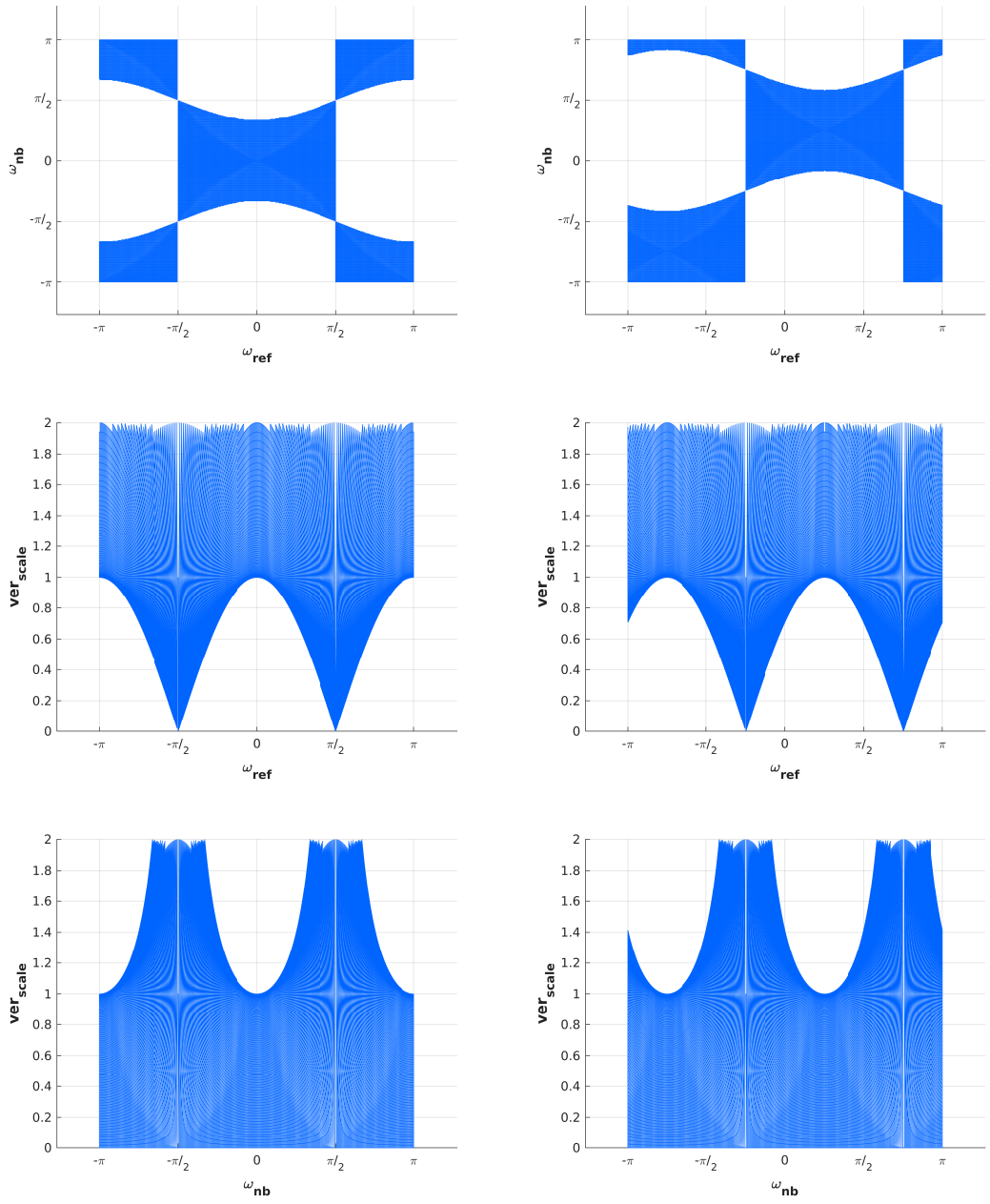


Figure 5.9: Analysis of angular constraints for ver_{scale}. ver_{scale} value is limited to 2. Left column represents the case where $\omega_{\Pi} = \pi/2$ and right column $\omega_{\Pi} = 3\pi/4$. Top row depicts the combined values of ω_{ref} and ω_{nb} that make ver_{scale} positive, in blue. Mid and bottom rows detail the influence of ω_{ref} and ω_{nb}, respectively. It is notorious that when ω_{ref} tends to ω_{Π} or $\omega_{\Pi} \pm \pi$, the value of ver_{scale} tends to zero. On the other hand, when ω_{nb} approaches ω_{Π} or $\omega_{\Pi} \pm \pi$, the value of ver_{scale} will increase to a predefined limit.

5.2.1.4 Point Clustering

There are two properties of the points that are used to create clusters. Besides the comparison between the beam true and expected ranges, it is also evaluated the relation of the normals. One of them is obtained directly from the previously estimated plane where the neighbor points lie, and another results from the local planar estimation. This local estimation comprises the point under analysis and the neighbors that pass the range relation constraint.

The points are first compared with their neighbors regarding the range values, using the methodology detailed in 5.2.1.3. The points may then clustered if the relation of the true (r_{true}) and expected (r_{exp}) ranges is below a predefined threshold ζ_{range}:

$$\frac{\max(r_{true}, r_{exp})}{\min(r_{true}, r_{exp})} - 1 < \zeta_{range} \tag{5.11}$$

As the relation is based in a quotient, for farther distances, it is allowed a larger error to cluster the points. If this condition is not fulfilled, the points are not clustered and is performed a comparison with the next neighbor, otherwise, the relation of the normals is evaluated.

The local planar estimation (Π_{local}) is based on the plane that fits better the set composed by the point under analysis and its neighbor points that had passed the range condition. The local plane normal ($\mathbf{n}_{\Pi_{local}}$) is then obtained using the covariance matrix of the set, in a strategy similar to the one described in the subsection 3.4, adapted to planar data.

Having a valid estimated normal to the local plane, the comparison between the angles of the local and the reference (Π - associated to the neighbor's cluster) planes is achieved by calculating the minimum angle (θ) between their normals, that is obtained from the result of their dot product:

$$\theta = \arccos\left(\frac{|\mathbf{n}_{\Pi_{local}} \cdot \mathbf{n}_{\Pi}|}{\|\mathbf{n}_{\Pi_{local}}\| \cdot \|\mathbf{n}_{\Pi}\|}\right) \tag{5.12}$$

After calculating the value of θ, the points will belong to the same cluster if θ is below a predefined angular threshold ($\theta < \zeta_{angle}$). If the cluster of the neighbor point has no valid normal to represent it, the clustering is made based only in the range threshold (ζ_{range}). The same happens if the reference point has no sufficient neighbors to form a plane, i.e., less than two. When the point being tested has more than one valid neighbor to associate, belonging to distinct clusters, the two clusters are merged.

When a point is associated to a cluster, the resulting representative normal is adjusted to the weighted average of it with the normal calculated for the point. Merging two clusters results in an expansion of the cluster with the lowest index, adding the points of the other cluster and averaging the representative normals, properly weighted based on the number of points.

5.2.1.5 Obtaining ω_Π and α_Π

As already discussed in subsection 5.2.1.3, the parameters ω_Π and α_Π are crucial for the comparison of a point with its neighbors. The value of ω_Π corresponds to the angle, with the XY plane, of the line that results from the intersection of the vertical plane formed by the beams over a fixed azimuth (α_{ref}), and the reference plane Π. The value of α_Π is the horizontal angle (in the XY plane) of the line resultant from the intersection of a plane formed by the beams over a fixed elevation angle (ω_{ref}), through a reference azimuth direction (α_{ref}), with the plane Π.

To obtain the value of ω_Π is defined a horizontal line $\mathbf{l}_{hor}$, passing through the origin with a direction $\mathbf{d}_{l_{hor}} = [\cos(\alpha_{ref})\ \sin(\alpha_{ref})\ 0]$. The normal of the vertical plane can then be defined by a horizontal vector $\mathbf{n}_{l_{hor}}$, orthogonal to the line $\mathbf{l}_{hor}$:

$$\mathbf{n}_{l_{hor}} = \begin{bmatrix} \sin(\alpha_{ref}) & -\cos(\alpha_{ref}) & 0 \end{bmatrix} \tag{5.13}$$

The intersection of the planes define the desired line, and its direction $\mathbf{d}_{l_\omega}$ is given by the normalized cross product between their normals:

$$\mathbf{d}_{l_\omega} = \frac{\mathbf{n}_\Pi \times \mathbf{n}_{l_{hor}}}{\|\mathbf{n}_\Pi \times \mathbf{n}_{l_{hor}}\|} \tag{5.14}$$

Ensuring that the vertical component of $\mathbf{d}_{l_\omega}$ is positive, the angle of it with the horizontal plane is calculated based in the dot product of $\mathbf{d}_{l_\omega}$ and $\mathbf{d}_{l_{hor}}$:

$$\omega_\Pi = \arccos\left(\frac{\mathbf{d}_{l_\omega} \cdot \mathbf{d}_{l_{hor}}}{\|\mathbf{d}_{l_\omega}\| \cdot \|\mathbf{d}_{l_{hor}}\|}\right) \tag{5.15}$$

For calculating the value of α_Π, both α_{ref} and ω_{ref} values need to be considered. A new line ($\mathbf{l}_{ver}$) is then defined, passing also through the origin, but with a direction $\mathbf{d}_{l_{ver}}$, now given by:

$$\mathbf{d}_{l_{ver}} = \begin{bmatrix} \cos(\alpha_{ref}) \cdot \cos(\omega_{ref}) & \sin(\alpha_{ref}) \cdot \cos(\omega_{ref}) & \sin(\omega_{ref}) \end{bmatrix} \tag{5.16}$$

Taking the normalized cross product of $\mathbf{d}_{l_{ver}}$ and $\mathbf{n}_{l_{hor}}$, is obtained the normal $\mathbf{n}_{l_{ver}}$ of the desired plane:

$$\mathbf{n}_{l_{ver}} = \frac{\mathbf{d}_{l_{ver}} \times \mathbf{n}_{l_{hor}}}{\|\mathbf{d}_{l_{ver}} \times \mathbf{n}_{l_{hor}}\|} \tag{5.17}$$

Similarly to equation 5.14, the intersection line direction ($\mathbf{d}_{l_\alpha}$) is calculated by using:

$$\mathbf{d}_{l_\alpha} = \frac{\mathbf{n}_\Pi \times \mathbf{n}_{l_{ver}}}{\|\mathbf{n}_\Pi \times \mathbf{n}_{l_{ver}}\|} \tag{5.18}$$

Now ensuring that the second component (y) of $\mathbf{d}_{l_\alpha}$ is positive and its first component (x) is non-null, the angle α_Π is obtained based on the horizontal projection of the intersection line direction:

$$\alpha_\Pi = \arctan\left(\frac{\mathbf{d}_{l_\alpha}(y)}{\mathbf{d}_{l_\alpha}(x)}\right) \tag{5.19}$$

Once that the value of the angle needs to be $\alpha_\Pi \in [0, \pi[$, and the arctan function returns values in the interval $\left]-\frac{\pi}{2}, \frac{\pi}{2}\right[$, if the obtained value of $\alpha_\Pi < 0$, its value is converted to $\alpha_\Pi = \pi + \alpha_\Pi$.

5.2.1.6 Cluster Classification

The generated clusters are classified as *planar*, *potential lines* and *undefined* during the segmentation step. The *planar* clusters usually have a great number of points clustered with a small error between their expected and true ranges and angular difference θ near to zero. The clusters classified as *potential lines* are, generally, composed by a set of few points that have the same properties of the *planar*'s when compared with *potential lines* neighbors. When compared with *planar* neighbors, this kind of clusters have a large range relation error, once they are always closer to the sensor.

The type of the clusters is determined by a voting system. Whenever some of the properties mentioned above is verified, the corresponding voter is incremented in the cluster. If the number of votes for a type, in relation to the number of points, is not significant, the cluster is classified as *undefined*.

The voter of the *potential lines* type can also be supported by the known information about the previously detected power lines (see subsection 5.2.3).

5.2.1.7 Algorithm

The algorithm 1 summarizes the information presented over the subsection 5.2.1. In some lines of code are referred the subsections that have the necessary explanations to understand the algorithm.

Algorithm 1 ${}^{l}\mathcal{L}_p \leftarrow \text{Segment}({}^{l}\mathcal{PCL})$

Initialize variables
for each $pt \in {}^{l}\mathcal{PCL}$ **do**
  $r_{ref} \leftarrow pt$ range
  $\alpha_{ref} \leftarrow pt$ azimuth angle
  $\omega_{ref} \leftarrow pt$ elevation angle
  $cl_{ref} \leftarrow -1$
  $\mathcal{NB}_{valid} \leftarrow pt$
  for each pt_{nb} (see subsections 5.2.1.1 and 5.2.1.2) **do**
    $r_{nb} \leftarrow pt_{nb}$ range
    $\alpha_{nb} \leftarrow pt_{nb}$ azimuth angle
    $\omega_{nb} \leftarrow pt_{nb}$ elevation angle
    $cl_{nb} \leftarrow pt_{nb}$ cluster
    $\Delta\alpha = |\alpha_{ref} - \alpha_{nb}|$
    if $\Delta\alpha > \Delta\alpha_{thr}$ **then**
      Go to next neighbor
    end if
    Calculate r_{exp} (see subsections 5.2.1.3, 5.2.1.4 and 5.2.1.5)
    if range relation $< \zeta_{range}$ (Eq. 5.11) **then**
      $\mathcal{NB}_{valid} \leftarrow \mathcal{NB}_{valid} \cup pt_{nb}$
    end if
  end for
  Calculate $\mathbf{n}_{\Pi_{local}}$ (see subsection 5.2.1.4)
  for each $pt_{nb} \in \mathcal{NB}_{valid}$ **do**
    $cl_{nb} \leftarrow pt_{nb}$ cluster
    $\mathbf{n}_{\Pi} \leftarrow pt_{nb}$ cluster normal
    if $\|\mathbf{n}_{\Pi_{local}}\| > 0$ **and** $\|\mathbf{n}_{\Pi}\| > 0$ **then**
      Calculate θ (Eq. 5.12)
    end if
    if $\|\mathbf{n}_{\Pi_{local}}\| == 0$ **or** $\|\mathbf{n}_{\Pi}\| == 0$ **or** $\theta < \zeta_{angle}$ **then**
      if $cl_{ref} == -1$ **then**
        $cl_{ref} \leftarrow cl_{nb}$
      else if $cl_{ref} \neq cl_{nb}$ **then**
        Merge cl_{ref} and cl_{nb}
      end if
    end if
  end for
  if $cl_{ref} == -1$ **then**
    Create new cluster
  else
    Add pt to cluster cl_{ref}
  end if
end for

43: **for each** cluster cl **do**
44: Evaluate cl type (cl_{type}) based in a voter (see subsection 5.2.1.6)
45: **if** $cl_{type} == potential\_line$ **then**
46: ${}^{l}\mathcal{L}_p \leftarrow {}^{l}\mathcal{L}_p \cup cl_{points}$
47: **end if**
48: **end for**
49: **return** ${}^{l}\mathcal{L}_p$

5.2.2 Line Detection

The line detection algorithm is a separate thread that is triggered whenever new *potential line* points are generated from the segmentation process. It is responsible for analyzing the incoming points, refining their clustering, and for trying to fit those clusters to a straight line. From this processing stage, the points are mapped into an Euclidean global frame, in order to allow the line matching between scans.

5.2.2.1 Cluster Refinement

Refining the clusters of the received points is needed to increase the quality of the line generated by the fitting step (subsection 5.2.2.2). This first clustering is based in the distance between points.

For associating the points, it is followed an approach similar to the Radially Bounded Nearest Neighbors (RBNN) clustering [83], using a predefined d_{max} as a breaking condition to stop the process. The value setting for d_{max} is based on the minimum expected distance between two power lines present in the environment of the inspection.

Although the use of RBNN can be time-consuming, its application is possible due to the low number of *potential line* points that are usually detected on each scan.

5.2.2.2 Line Fit

Analyzing the created clusters cl_{lpts}, the algorithm tries to approximate their points to a 3D line, using the method described in 3.4. The process of line fit is only applied to clusters with more than three points, in order to decrease the probability of having an erroneous estimation of a line.

The resultant fitted line is considered valid or not based on the value of λ_{rel}. This value represents the relation between the two most significant eigenvalues (λ_1, λ_2). If the condition of $\lambda_1/\lambda_2 > \lambda_{rel}$ is met, the fitting is considered valid, otherwise, the correspondent cluster is removed from the further processing steps.

Due to some possible gaps on the line detection, different clusters can belong to the same line and, in that case, they shall be merged. For that, the collinearity of the distinct fitted lines is evaluated, using three conditions:

- The angular difference of the direction vectors ($\sigma = \arccos(\mathbf{v}_0 \cdot \mathbf{v}_1 / \|\mathbf{v}_0\| \cdot \|\mathbf{v}_1\|)$) needs to be below an angular threshold σ_{thr};
- The shortest distance between the lines (using the approach described in section 3.5), defined by their line segments, cannot be greater than d_{max};
- The distance, evaluated locally, of the line segments must be less than d_{max}.

If all of those three conditions are met, the lines are considered collinear, their clusters are merged, an a new fitted line is calculated. As the skew lines can have their shortest distance in a point far from the line segments, it arises the need for evaluating their distance locally.

Once the lines are estimated, if there exists some clusters that has no fitted line (low number of points), they can also be added to other cluster containing a line. This adding process is based on the point to line distance (described in section 3.5). If the distance value is below a more restrictive value than d_{max}, the point is added to the line cluster.

5.2.2.3 Algorithm

In the algorithm 2 is exposed the process described over the subsection 5.2.2.

Algorithm 2 $({}^{w}\mathcal{L}_s, {}^{w}pt_{iso}) \leftarrow$ Line_Detect$({}^{w}\mathcal{L}_p)$

```
1:  Initialize variables
2:  for each pt ∈^w L_p do
3:    associated ← false
4:    for each cl ∈ cl_lpts do
5:      for each pt_cl ∈ cl do
6:        if distance(pt,pt_cl) < d_max then
7:          cl ← cl ∪ pt
8:          associated ← true
9:          break
10:       end if
11:     end for
12:     if associated then
13:       break
14:     end if
15:   end for
```

16: **if** $\sim associated$ **then**
17: Create new cluster new_{cl}
18: $new_{cl} \leftarrow pt$
19: $cl_{lpts} \leftarrow cl_{lpts} \cup new_{cl}$
20: **end if**
21: **end for**
22: **for each** $cl \in cl_{lpts}$ **do**
23: **if** cl has more than 3 points **then**
24: Estimate best fit line (see subsection 5.2.2.2)
25: **if** $\lambda_1/\lambda_2 > \lambda_{rel}$ **then**
26: ${}^w\mathcal{L}_s \leftarrow {}^w\mathcal{L}_s \cup cl$
27: **end if**
28: **else**
29: ${}^w pt_{iso} \leftarrow {}^w pt_{iso} \cup cl$
30: **end if**
31: **end for**
32: **for each** $line \in {}^w\mathcal{L}_s$ **do**
33: **for each** $line_{cmp} \in {}^w\mathcal{L}_s \backslash line$ **do**
34: $is\_collinear \leftarrow$ Verify collinearity (see subsection 5.2.2.2)
35: **if** $is\_collinear$ **then**
36: Merge $line$ and $line_{cmp}$
37: Estimate new best fit line
38: **end if**
39: **end for**
40: **end for**
41: **for each** $line \in {}^w\mathcal{L}_s$ **do**
42: **for each** $pt \in {}^w pt_{iso}$ **do**
43: **if** min_distance($line$,pt) $< d_{max}/scale$ **then**
44: $line \leftarrow line \cup pt$
45: **end if**
46: **end for**
47: Estimate new best fit line
48: **end for**
49: **return** $({}^w\mathcal{L}_s, {}^w pt_{iso})$

5.2.3 Power Line Modeling

The power line modeling is the last step of the proposed algorithm. Here, the estimated straight lines received from each scan are matched and grouped. Once grouped, this process tries to estimate the mathematical model of the power lines. If some power line is detected, the information about its positioning can be returned to feed the segmentation process of the next scans (see subsection 5.2.1).

5.2.3.1 Line Matching

The lines detected in each scan are stored into sets that represent a possible power line each. To match the new incoming lines, it is performed a comparison of them with all the last added lines of the sets. The lines are then added to the set if they respect some collinearity constrains, similarly to what is done in subsection 5.2.2.2.

Choosing the last added line to compare with the new ones is due to a more probable linear similarity between them, as they correspond to scans temporally closer than the others. The isolated points are also compared with the power line models. If the new lines do not match any of the already built models, a new one is created.

5.2.3.2 Model Estimation

For the online processing of the data, knowing where the lines are, at each moment, is far more important than precisely match them with a mathematical model. For this reason, the model estimation relies on the method presented in section 3.6, for a catenary curve model, using only the first ($\mathbf{l}_{first}$) and last ($\mathbf{l}_{last}$) added lines of each set.

Each line segment is stored by saving the mean point of the original data (μ_l), the direction vector of the fitted line ($\mathbf{d}_l$), and its minimum and maximum x values (x_{min} and x_{max}). The power line to be estimated is modeled by a straight line into the horizontal plane and by a catenary curve into the vertical one.

To obtain the horizontal model of the line it is needed to simply project the line segment into the horizontal (XY) plane. For having it into the simplified format of $y = m \cdot x + b$, the values of m and b are easily calculated:

$$y = \frac{\mathbf{d}_{ly}}{\mathbf{d}_{lx}} \cdot x + \left(\mu_y - \frac{\mathbf{d}_{ly}}{\mathbf{d}_{lx}} \cdot \mu_x \right) \tag{5.20}$$

The vertical analysis is also a 2D approach, where the vertical coordinate is z, and the horizontal (x_{hor}) results from a composition of the x and y values, based on the estimated horizontal line. Having a line with an angle $\delta = \arctan(m) = \arctan(\mathbf{d}_{ly}/\mathbf{d}_{lx})$, the value of x_{hor} is obtained by doing $x_{hor} = x/\cos(\delta)$. Based on the figure 3.9 and equations of section 3.6, the variables correspondences are: $y \rightarrow z$ and $x \rightarrow x_{hor}$.

The angles θ_{first} and θ_{last} are also obtained from the direction vectors $\mathbf{d}_{lfirst}$ and $\mathbf{d}_{llast}$, respectively:

$$\theta_{first} = \left| \arctan \left(\frac{\sqrt{\mathbf{d}_{lfirst\_x}^2 + \mathbf{d}_{lfirst\_y}^2}}{\mathbf{d}_{lfirst\_z}} \right) \right| \tag{5.21}$$

Adapting equation 3.23 to the used variables, the value of a is given by:

$$a = \frac{z_{last} \cdot \sin(\theta_{last}) - z_{first} \cdot \sin(\theta_{first})}{\sin(\theta_{last}) - \sin(\theta_{first})} \tag{5.22}$$

From equation 5.22 it is easily observable that the values of $\sin(\theta_{first})$ and $\sin(\theta_{last})$ have to be different, which means that $\theta_{first} \neq \theta_{last}$, assuming that $\theta_{first}, \theta_{last} \in \left]0, \frac{\pi}{2}\right[$. In figure 3.9 is visually explicit that the value of a is always smaller than z_{first} and z_{last}.

The c parameter is obtained by adapting equation 3.21:

$$c = (z_{first} - a) \cdot \sin(\theta_{first}) \tag{5.23}$$

Due to the effect of gravitational forces, the power lines always have a curvature opening pointing upwards, which means that the value of c must be always positive.

Finally, the value of b is calculated based in adapted equation 3.24:

$$b = x_{hor\_first} + c \cdot \operatorname{arccosh}\left(\frac{z_{first} - a}{c}\right) \tag{5.24}$$

As the hyperbolic cosine is the sum of two vertically symmetric exponential functions $\left(\cosh(k) = \frac{\exp(k)+\exp(-k)}{2}\right)$, its minimum value is 1. This minimum is a constrain to the argument of the arccosh, that needs to be greater or equal to 1. Given this, in equation 5.24, the condition of $\frac{z_{first}-a}{c} \geq 1$ has to be respected.

Summarizing the limitations described above, the catenary estimated parameters are considered valid if:

- $\theta_{first} \neq \theta_{last}$;
- $a < z_{first}$ **and** $a < z_{last}$;
- $c > 0$;
- $\frac{z_{first} - a}{c} \geq 1$.

Although the incoming isolated points are associated to the power lines sets, they are not used to perform the estimation of their model online. The objective of aggregating them is to have more data available to a possible offline model refinement algorithm.

5.2.3.3 Parameter Return

After modeling the power lines or, at least, associating the detected lines to a valid set, is generated a bounding box (defined into the global frame) that comprises them. The

algorithm returns the power line models, if any, associated to the bounding box and the direction (sensor's azimuth) of the last detected lines. This information can be used to feed the segmentation algorithm, refining its output for the subsequent scans.

5.2.3.4 Algorithm

The algorithm 3 details the power line modeling, described over the subsection 5.2.3.

Algorithm 3 ${}^{w}\mathcal{PL}_m \leftarrow$ Estimate_Model(${}^{w}\mathcal{L}_s$,${}^{w}\,pt_{iso}$)

```
1: Initialize variables
2: for each line ∈^w L_s do
3:    associated ← false
4:    for each pw_line ∈^w PL_m do
5:       l_last ← pw_line last added line
6:       is_collinear ← Verify collinearity of line and l_last
7:       if is_collinear then
8:          pw_line ← pw_line ∪ line
9:          associated ← true
10:         l_first ← pw_line first added line
11:         params_valid ← Estimate catenary curve parameters (see subsection 5.2.3.2)
12:         if params_valid then
13:            Update pw_line parameters
14:         end if
15:         break
16:      end if
17:   end for
18:   if ∼ associated then
19:      Create new power line set new_pw
20:      new_pw ← line
21:      ^wPL_m ←^w PL_m ∪ new_pw
22:   end if
23:   Update bounding box and direction of detection
24: end for
25: for each pt ∈^w pt_iso do
26:   for each pw_line ∈^w PL_m do
27:      l_last ← pw_line last added line
28:      if min_distance(l_last,pt) < d_max/scale then
29:         pw_line ← pw_line ∪ pt
30:      end if
31:   end for
32: end for
33: return ^wPL_m
```

This page was intentionally left blank.

Chapter 6

Implementation

The proposed algorithm was intended to be applied in an Unmanned Aerial Vehicle (UAV) using a Light Detection And Ranging (LiDAR) sensor to percept the surrounding environment, as already suggested in chapter 4. The STORK UAV is a multirotor, equipped with a LiDAR, that is frequently used to perform inspection tasks. This chapter exposes the main characteristics of the used UAV and the respective sensor, detailing some properties and adaptations that are crucial to the algorithm's implementation.

6.1 STORK UAV

The multirotor UAV STORK [42] (figure 6.1) is a custom hexacopter designed to achieve both efficiency and versatility. Its primary application is the power assets inspection. Nonetheless, it has been also used in the first trials of the SpilLess project, and in several precise mapping surveys. This range of applications can be attained by means of an adaptive payload methodology: using the same frame, some payload sensors can be easily replaced by others that provide a different type of data.

This UAV is capable of navigating in both manual and autonomous modes, having also the ability to perform some autonomous maneuvers, like takeoff, landing or the inspection of a structure, using the onboard sensors. In its current state, STORK UAV is low-level controlled by a customized autopilot (INESC TEC Autopilot) and has an onboard computer that is responsible for controlling the UAV at a higher level (ODROID-XU4[1], running Ubuntu 16.04 LTS[2] and Robotic Operating System (ROS) Kinetic Kame[3]).

[1] `https://wiki.odroid.com/odroid-xu4/odroid-xu4`
[2] `http://releases.ubuntu.com/16.04`
[3] `http://wiki.ros.org/kinetic`

Figure 6.1: STORK UAV

6.1.1 Payload sensors

6.1.1.1 Navigation

For navigation, this UAV has two Inertial Measurement Units (IMUs) and two Global Navigation Satellite System (GNSS) receivers. Besides the default low-cost IMU sensors, the autopilot can also use the STIM300[4], that is a high-performance IMU. Similarly, the single-band GNSS receiver Ublox NEO-M8T[5] is the low-cost alternative of K501G[6], a dual-band receiver that supports onboard Real-Time Kinematic (RTK) positioning [161].

Although the low-cost combination of sensors generally fulfills the applications requirements, others like precise mapping surveys imply the use of highly-accurate sensors.

6.1.1.2 Perception

To percept the surrounding environment, STORK UAV has two visual cameras (a fixed Teledyne Dalsa G3-GC10-C2050[7], pointing 45 degrees forward-down, and a FLIR Point-Grey CM3-U3-13S2C-CS[8], attached to a gimbal), a LiDAR sensor (Velodyne VLP-16[9]), and in some applications a thermographic camera (FLIR A65[10]) can be used.

[4] https://www.sensonor.com/products/inertial-measurement-units/stim300
[5] https://drotek.com/shop/en/u-blox/884-ublox-neo-m8t-gps-lis3mdl-compass-xxl.html
[6] http://www.comnavtech.com/products-detail.asp?id=2&sw=1920&sh=1080
[7] http://www.teledynedalsa.com/en/products/imaging/cameras/genie-nano-gige
[8] https://www.ptgrey.com/chameleon3-13-mp-color-usb3-vision
[9] https://velodynelidar.com/vlp-16.html
[10] https://www.flir.com/products/a65

The data provided by these sensors can be used as an input for processing algorithms that will be used by the navigation layer, or to create other outputs, like Three-Dimensional (3D) models of a structure.

6.2 Velodyne VLP-16

Velodyne VLP-16 (figure 6.2) is a spinning LiDAR (see section 3.1) that creates 360 degrees 3D images by using 16 laser beams mounted in a housing that spins from 5 to 20 times per second. It can operate either in single (*Strongest* or *Last* return) or dual (both returns) mode. This allows the sensor to provide up to 300,000 points each second operating in single mode, or twice that in dual return mode [136].

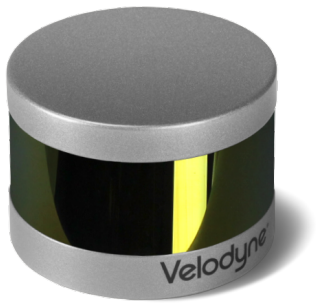


Figure 6.2: Velodyne VLP-16 [162]

For interfacing with this LiDAR sensor, there is an already developed ROS package that is publicly available for use[11].

6.2.1 Operational details

The knowledge of the Velodyne VLP-16's operational details is important for the development of further time-efficient algorithms related with point cloud segmentation.

6.2.1.1 Beam Structure

This sensor is capable of measuring distances up to 130 meters. Each adjacent beam of the 16 lasers set is vertically spaced by 2 degrees, which results in a vertical Field-Of-View (FOV) of 30 degrees. The horizontal angular resolution is variable and depends on the sensor's rotation frequency, once the firing period is kept. Table 6.1 shows the horizontal angular resolution for different rotation frequencies.

[11] `http://wiki.ros.org/velodyne`

Table 6.1: Horizontal angular resolution.

Frequency (Hz)	Resolution (°)
5	0.1
10	0.2
15	0.3
20	0.4

In order to prevent possible interferences between the fired beams, the sequence of firing is not made from one end to the other. Instead, the vertical firings are organized in a sequence described in table 6.2.

Table 6.2: Firing sequence.

Laser ID	Vertical Angle (°)
0	-15
1	1
2	-13
3	3
4	-11
5	5
6	-9
7	7
8	-7
9	9
10	-5
11	11
12	-3
13	13
14	-1
15	15

6.2.1.2 Data Format

Velodyne VLP-16 returns the read points organized into packets, each one composed by a header, 12 data blocks, a timestamp and factory bytes. In a data block is contained a two-byte flag, a two-byte azimuth and 32 three-byte data points (two bytes for range and one for intensity), which means that when the sensor is operating in single mode, each data block contains two firing sequences of 16 lasers. When in dual mode, each data block only contains the information of one firing sequence, representing both the

strongest and the last returns (if they are the same, the second-strongest return is also reported). Figure 6.3 depicts the structure of one packet in single return mode.

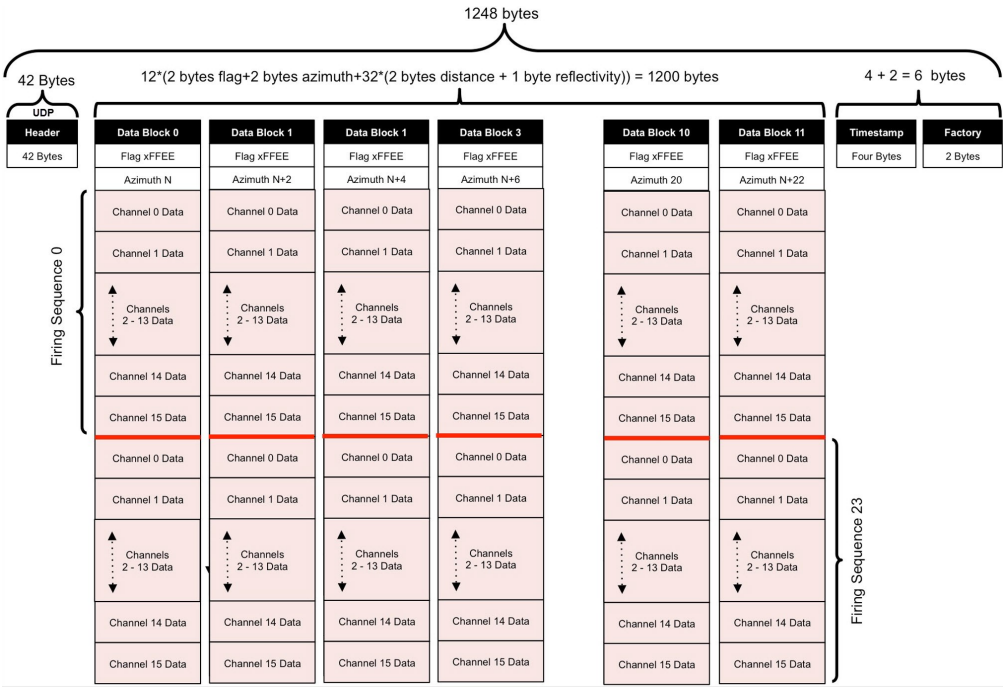


Figure 6.3: Single return mode packet structure [136].

For converting the returned points from spherical to cartesian coordinates using equations 3.1, 3.2 and 3.3:

- The range value (r) is directly obtained from each data block;
- The elevation angle (ω) is inferred by the channel number (based on table 6.2);
- The azimuth angle (α) can be:
  - Directly obtained from the data block, in dual return mode;
  - Calculated as the mean azimuth value between two consecutive data blocks if it is the second firing sequence of a data block, in single return mode.

6.2.1.3 Timing

The timestamp contained in each packet given by the sensor marks the moment of the first data point in the first firing sequence and corresponds to the number of microseconds elapsed since the top of the hour. When the sensor powers up it begins counting microseconds using an internal time reference.

The VLP-16 can synchronize its timestamp with Coordinated Universal Time (UTC)[12] time. This feature is useful for matching the provided data with an inertial navigation system and generate precise maps or avoid obstacles. For synchronizing it requires the use of an external GNSS receiver that generates a Pulse Per Second (PPS) signal and a National Marine Electronics Association (NMEA) GPRMC[13] message. From the GPRMC message, the sensor reads the minutes and seconds, setting its timestamp to the number of microseconds past the hour, per UTC.

6.2.2 Software driver

The available *velodyne* ROS package is divided into sub-packages that allow the establishment of a connection with several Velodyne LiDAR sensors (among them, the VLP-16 model), receive their data and create a point cloud with it.

The three main sub-packages that are contained on the *velodyne* package are *velodyne_msgs*, *velodyne_driver* and *velodyne_pointcloud*. In *velodyne_msgs* are placed the ROS message type definitions for the Velodyne 3D LiDARs. The other two are closely related to the data acquisition and processing.

6.2.2.1 *velodyne_driver*

Velodyne_driver is the ROS device driver for several Velodyne 3D LiDARs. It is responsible for establishing the connection with the sensor and publishing the received packets into a ROS topic called `\velodyne_packets`.

This package allows configuring the parameter *n_packets*, that corresponds to the number of sensor's read packets to group before publishing them to the topic. If *n_packets* is greater than 1, the published message will have the timestamp of the last received packet. Depending on the processing power of the system, a value of *n_packets* too small may lead to the loss of some messages, due to the high-frequency publishing. On the other hand, if the value is too high, it may lead to a bad conversion of the points to the global frame in a moving platform, due to the loss of temporal resolution for the platform's attitude matching.

6.2.2.2 *velodyne_pointcloud*

This package subscribes to the `\velodyne_packets` topic and converts the packets into a 3D point cloud, publishing it to the topic `\velodyne_points`.

[12] Primary time standard by which the world regulates clocks and time.
[13] `https://www.gpsinformation.org/dale/nmea.htm`

In its implementation, this package publishes a sparse point cloud containing only accepted points, i.e., valid points reported into the packets that are within a defined range and angular interval. The resultant points are usually published into the sensor's local frame, however, if the user intends to, it can be published into another desired reference frame (assuming that there exists a valid transformation relation between them).

6.3 Customized Velodyne ROS package

The already available Velodyne ROS package will be used to generate the input data for the developed algorithm. The *velodyne_driver* (subsection 6.2.2.1) is responsible for converting the raw sensor's data into a ROS topic, making it easily interpreted inside the ROS environment. The package *velodyne_pointcloud* (subsection 6.2.2.2) subscribes to that topic and converts the data into a 3D point cloud, using the nodelet *CloudNodelet*, that provides the needed functions.

6.3.1 Default Data Structure

The *CloudNodelet* transforms the Velodyne data into a set of points mapped into the Euclidean space, the point cloud, publishing them to the `\velodyne_points` topic, in the same order as the points are received. Each point has associated the data structure presented in table 6.3.

The values of x, y and z correspond to the measured point converted into the 3D Euclidean space, *intensity* is the measured intensity provided by the sensor. The value of *ring* encodes the vertical angle of the beam (ω), where $\omega = -15 + 2 \cdot ring$.

Table 6.3: Default point data structure.

Type	Name
$float$	x
$float$	y
$float$	z
$float$	$intensity$
$uint16\_t$	$ring$

However, it is worth noting that the reference frame used by the nodelet does not match the one provided by the manufacturer in the sensor's manual [136] (depicted in figure 3.2). In the default frame, y_{def} points forward, x_{def} right and z_{def} upwards, but in the one used by the nodelet, x_{nod} points forward, y_{nod} left and z_{nod} upwards. This represents a rotation of $\pi/2$ around the z axis.

For obtaining the values of x, y and z in the sensor's default frame, the equations 3.1, 3.2 and 3.3 are valid. In the nodelet's frame, the values need to be adapted to $x_{nod} = y_{def}$ and $y_{nod} = -x_{def}$.

6.3.2 Adapted Data Structure

The main drawback of existing nodelet is the fact that the default data structure is not published in an ordered fashion. This is a requirement of the proposed algorithm (pointed out in subsection 5.2.1.1), so the nodelet needed to be adapted.

Although the points are reported by the VLP-16 in an unordered fashion, all the data points are stored and accessible at the same time. Having table 6.2 as a reference, the data access, by a crescent vertical angular value, can be done by using:

$$laser_{ID} = 2 \cdot it - 15 \cdot (it > 7) \tag{6.1}$$

Where *it* is an iterator of a *loop* cycle that ranges from 0 to 15.

To achieve a faster processing of the data by the proposed algorithm, the values of *range* and *azimuth* angle were also added to the default data structure (table 6.4). Having the *range* avoids the need for its calculation from the x, y and z values, as well as for the *azimuth*. The data organization into a Two-Dimensional (2D) matrix format also benefits from these values.

Table 6.4: Adapted point data structure.

Type	Name
$float$	x
$float$	y
$float$	z
$float$	$intensity$
$uint8\_t$	$ring$
$uint16\_t$	$azimuth$
$float$	$range$

The type of *ring* was changed to an *8-bit* because it varies only between 0 and 15. The value of *range* and *azimuth* are directly obtained from the sensor data. An *unsigned 16-bit* type for the *azimuth* is possible by approximating its value (in degrees) with a precision of 2 decimal digits, multiplied by 100.

Adding *range* and *azimuth* to the structure does not affect the processing time of the *CloudNodelet*, as they are already available and used to map the points into the

Euclidean space. Placing these values into the end of the structure allows keeping the compatibility of the data with previously developed algorithms, once it is parsed from the beginning of a reference position, given by a defined offset in the published topic.

This page was intentionally left blank.

Chapter 7

Results

The multirotor Unmanned Aerial Vehicle (UAV) STORK is used to perform electrical power assets inspections, however, it is also capable of gathering data to generate accurate maps of an environment. During other applications beyond the inspections tasks, it is common to encounter some electrical assets, being mainly power lines and pylons. To validate the PL^{2}DM algorithm a dataset of a mapping survey was used, where there were some power lines (described in section 7.1).

This chapter briefly describes some properties of the used dataset and presents the outputs generated by the proposed algorithm. Associated to the obtained results, an analysis of the PL^{2}DM performance was made.

7.1 Experimental Dataset

The used dataset was recorded during a mapping survey of a rock stockpile in the Malaposta quarry, in Santa Maria da Feira, Aveiro, Portugal (figure 7.1). The flight of the survey was performed in manual mode and with special care due to the presence of power lines. Above the mapped stockpile, there was a span composed of 6 power lines and a guard cable, which made this dataset suitable for testing the algorithm.

From the UAV onboard images, in figure 7.2, it is noticeable the presence of the described power lines. In the image on the right can be seen that the background created by the stockpile is very noisy, which can lead to a worse performance of visual power line detection algorithms. As they are mainly edge based, the presence of the rocks might lead to a poor quality of the output results or to an increase of the processing time, once that there are many edges to be analyzed. The presence of the power line's shadows on

Figure 7.1: Malaposta quarry.

the ground (also visible in figure 7.2) can also generate false positives, depending on the threshold values used in the edge detector.

Figure 7.2: STORK UAV onboard images.

As the main objective of the flight was the mapping of the stockpile, the UAV had the Velodyne sensor mounted under its frame, pointing downwards, with a negative 75 degrees pitch rotation. This kind of configuration allows the Light Detection And Ranging (LiDAR) sensor to percept only the environment below and sideways with respect to the UAV position. This means that everything that is immediately above, in front or in the back of the UAV cannot be detected.

Both Inertial Measurement Units (IMUs) and Global Navigation Satellite System (GNSS) receivers supported by the UAV were installed and running during the flight. Due to the UAV being controlled manually, the estimation of its pose relied on the data incoming from the low-cost set of sensors (see subsection 6.1). Meanwhile, the raw data

provided by the more expensive set was being logged at a high rate to ensure a good quality of the post-processed stockpile map.

Knowing that the power line span was almost aligned with North, and analyzing the UAV's trajectory (figure 7.3), represented into an East-North-Up (ENU) referential, is possible to deduce that the power line was detected from several positions. The UAV performed a flight on both sides of the power line, passing also under and above it when switching sides.

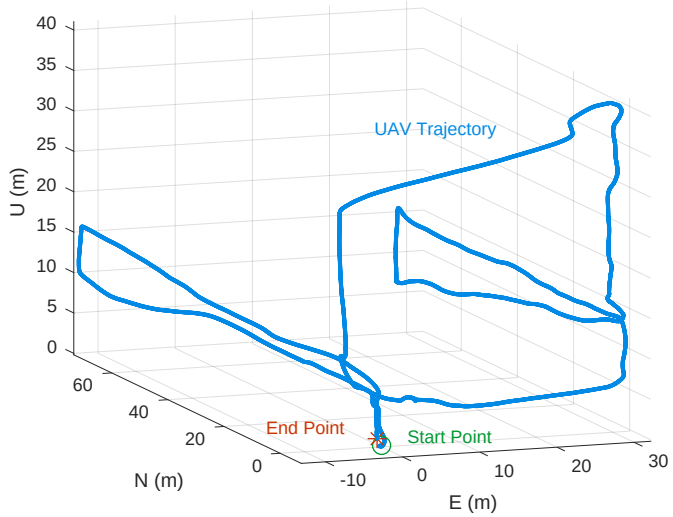


Figure 7.3: UAV trajectory.

Among others, the available dataset has the topic `\velodyne_packets` and the UAV stamped pose, published by a *mavros* package. To apply the proposed algorithm, the topic `\velodyne_packets` was subscribed by the adapted version of the package *velodyne_pointcloud*, that publishes the point cloud to the `\velodyne_points` topic, respecting the new structure for the data points. This is the topic which the proposed algorithm subscribes to. The pose of the UAV is then used with a *tf2* package, generating the proper stamped relations between the sensor's, UAV's and global frames.

7.2 Parameters

During the description of the developed algorithm, in chapter 5, were introduced some parameters in whose its functionality relies on. In table 7.1 are presented the values of the main ones used to obtain the results of section 7.3. For each parameter listed is associated a brief description of its influence in the algorithm's behavior.

Table 7.1: Used parameters values.

	Parameter	Unit	Value	Description
Segmentation	$\Delta\alpha$	*degrees*	10	Maximum azimuth (α) difference between a point and its neighbor.
	ζ_{range}	–	$0,02$	Maximum relation error of true (r_{true}) and expected (r_{exp}) ranges.
	ζ_{angle}	*degrees*	5	Maximum angular error of local ($\mathbf{n}_{\Pi_{local}}$) and cluster plane ($\mathbf{n}_{\Pi}$).
Line Detection and Power Line Modeling	λ_{rel}	–	100	Minimum relation of λ_1/λ_2 to accept the line fitting parameters.
	d_{max}	*meters*	$0,75$	Maximum distance to cluster points.
	σ_{thr}	*degrees*	2	Maximum angular error to merge line segments.

7.3 Results from the Dataset

The results obtained are divided by the several layers of the proposed algorithm. Each result is then analyzed, being made a global analysis of the algorithm's performance at the end of this section.

7.3.1 Segmentation and Point Classification

Using the values of parameters of table 7.1, the segmentation result for one Velodyne scan is depicted in figure 7.4. Each color of the points in the figure represents a different cluster. From there, it is clear that the points with planar properties tend to be represented by the same color, as they are associated to the same cluster.

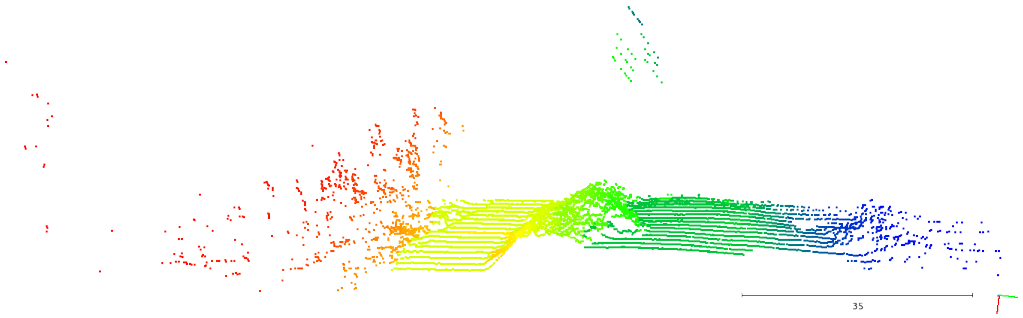


Figure 7.4: Segmented point cloud from one scan, colored by point clusters. Scan direction from blue to red points.

For a better understanding of the resultant clusters, in figure 7.5 the structures detected in that scan are identified. Having the point cloud associated with the structures, it becomes evident that clusters' breakpoints occur mainly in the presence of the power lines, stockpiles, and vegetation, as expected.

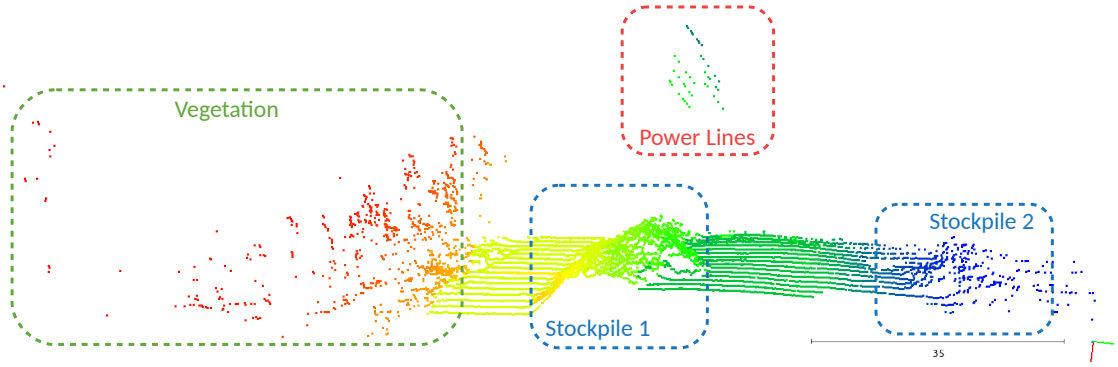


Figure 7.5: Object identification in the point cloud.

Focusing on the clusters created for the power line points, both figures 7.4 and 7.5 evidence an over-segmentation of the points. This can result from the difficulty of detecting a power line contiguously. As they have a small section, working with Velodyne in single mode, both *Strongest* and *Last* return modes might fail to detect it when other structures are behind, due to the beam dispersion [136]. This raises the importance of the cluster refinement (subsection 5.2.2.1) and line association (subsection 5.2.2.2) performed during the *Line Detection* step.

From the resultant clusters, the algorithm classifies the points based on a voting system that evaluates the relations between a point and its neighbors. Figure 7.6 shows the classification attributed to the points. Blue represents *undefined* points, green *planar*, and red the *potential line* ones (see subsection 5.2.1.6).

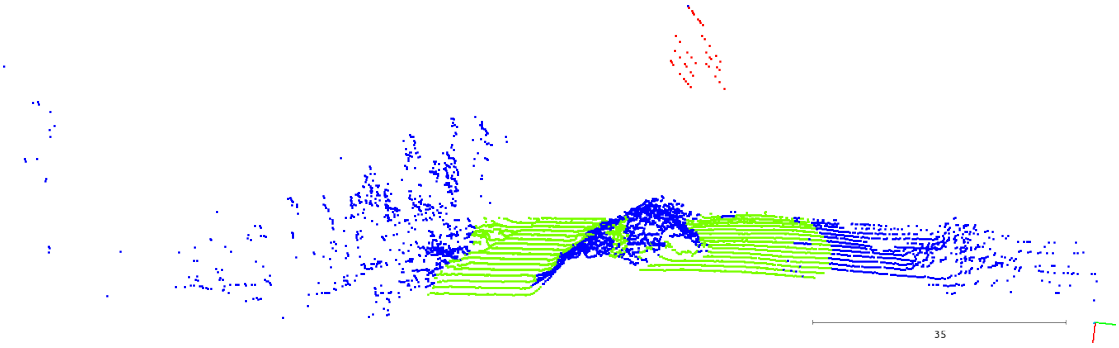
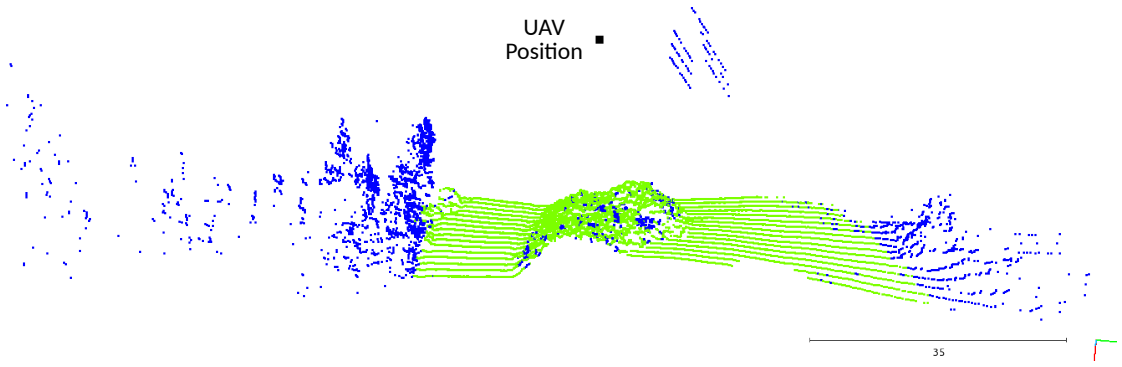


Figure 7.6: Segmented point cloud from one scan, colored by point type. Green corresponds to *planar*, red to *potential lines* and blue to *undefined*.

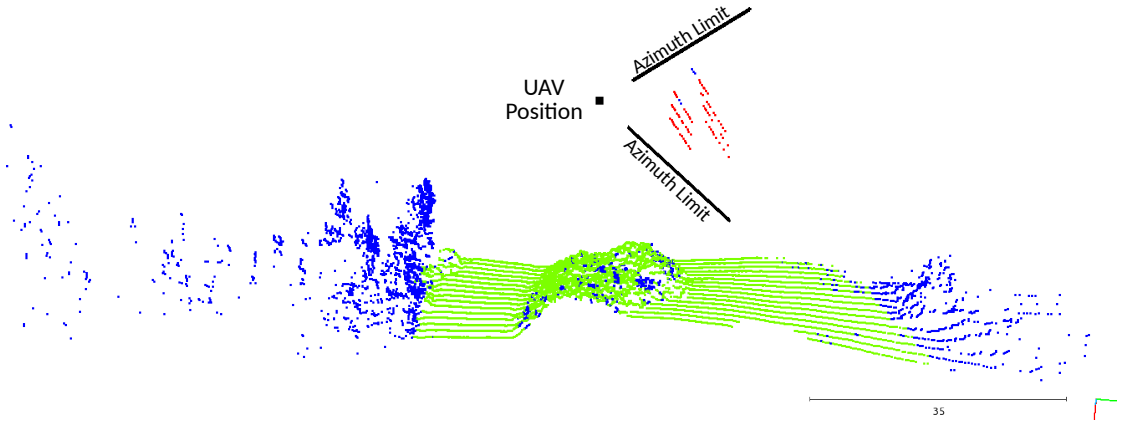
The results in figure 7.6 show an almost correct classification of the power line as *potential line* points (one point is classified as *undefined*). The stockpiles and vegetation are predominantly composed by *undefined* points and the ground is considered *planar*.

A part of the stockpile is not well classified as a *planar* structure. This may occur depending on the point where the cluster begins to grow. As explained in subsection 5.2.1.4 and in algorithm 1 (lines 23-36), whenever a point has no sufficient neighbors to estimate a local plane, or the neighbor cluster has no normal estimated, the clustering is only based in the range relation (ζ_{range}). This leads to a cluster growing without a representative normal associated. Eventually, a point containing a valid local planar estimation may be added to it and used as a reference for further comparisons, however, the points already added to the cluster are not re-evaluated, remaining as part of the final cluster.

Another property of the developed algorithm that had revealed to be useful was the feedback information about the line model. The figure 7.7 evidences its effect, where both images refer to the same scan. Here occurs the effect described above, being the stockpile considered as *planar*.



(a) Without feedback from previously detected lines.



(b) With feedback from previously detected lines.

Figure 7.7: Effect of the feedback about previously detected lines.

In figure 7.7(a) the algorithm was running without getting any feedback about the position of the lines from the previous scans. As they were detected by the LiDAR without any background behind them, the voter is not capable of distinguishing them from the vegetation or outliers, attributing the *undefined* classification.

Adding the information feedback, the point classification task had a better performance, labeling almost all the power line points as *potential lines*. In figure 7.7(b), the information received by the segmentation is represented by the red lines. Their aperture, from the LiDAR's center, is related with the azimuth limits. Their length and initial and final points represent the range interval where is expected the lines to be found. This value is obtained from the bounding box created when the lines were detected.

7.3.2 Line Detection

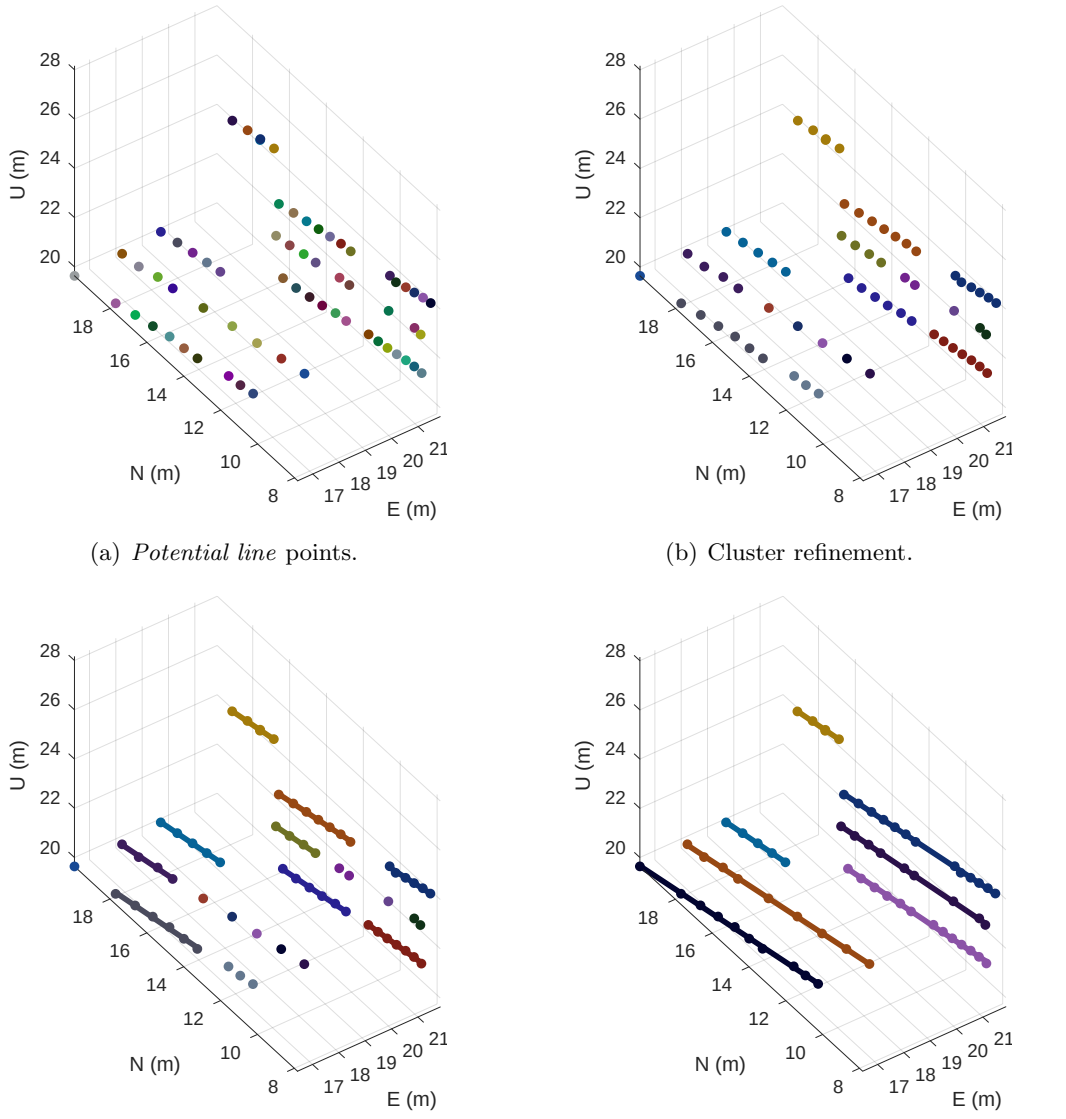
The line detection thread is triggered whenever new *potential line* points are found in the segmentation of each scan. The process of fitting the incoming points to lines is depicted in figure 7.8, for one scan. From this point forward, the points are mapped into an ENU global frame that follows the same reference frame used in figure 7.3.

Having the incoming *potential line* points (figure 7.8(a)), the algorithm follows the sequence presented in subsection 5.2.2. It first performs the cluster refinement (subsection 5.2.2.1) based in the defined value of d_{max}, resulting both in some clusters and isolated points (figure 7.8(b), colored by cluster).

From the resultant clusters, it is performed a line fitting to the ones containing more than three points, as referred in subsection 5.2.2.2. If the eigenvalues, obtained from the line estimation, comprise the relation imposed by the parameter λ_{rel}, the line is considered valid and associated with the cluster. Otherwise, it is not considered for the remaining steps. In the example scan, the algorithm was capable of fitting 9 valid lines.

As it can be seen in figure 7.8(c), there were still some disperse points and some separate line segments that corresponded to the same line. This may be overcome by verifying the collinearity of the valid segments, with each other, and then with the isolated points (subsection 5.2.2.2).

If it is verified that two line segments are collinear, or that some isolated point belongs to a line, the correspondent points are clustered and the line parameters are re-estimated (applying the same constraints to the eigenvalues as before). In most of the cases (like the one depicted in figure 7.8(d)), the algorithm is capable of properly merging the line segments and recover the isolated points information by adding them to the correspondent line.



(a) *Potential line* points.

(b) Cluster refinement.

(c) Line fitting.

(d) Merge collinear lines and points.

Figure 7.8: Line detection process.

7.3.3 Power Line Modeling

The line matching step for trying to obtain the power line model is based on collinearity properties of the candidate lines. Therefore, it is clearly dependent on the quality of line detection. Once it is used a global reference frame for the line modeling, this means that, in consequence, the line matching is very sensitive to the possible lack of accuracy of the UAV pose estimation.

The relations between the LiDAR, UAV and global frames were created by using the estimated pose of the UAV, provided by a *mavros* topic. This estimation is performed onboard by the autopilot of the UAV, publishing information to the ROS environment at a rate of 30 Hz. The computer can then access this data and create the needed relations between frames during the flight.

Running all the dataset, the algorithm has generated the points depicted in figure 7.9 as being power lines. Each attributed color corresponds to a distinct detected line. As it can be seen, although the algorithm was capable of correctly matching the lines over the time, this was only possible due to a local comparison between the candidate line and the last added line of the model. If a comparison with the global model of the line was desired, it would have to be very noise permissive to allow the association. The uncertainty in the estimated position of the power lines, observable in figure 7.9, was mainly caused due to an erroneous height estimation. This conclusion is suggested by a more notorious effect of those estimation errors in the side and cut views.

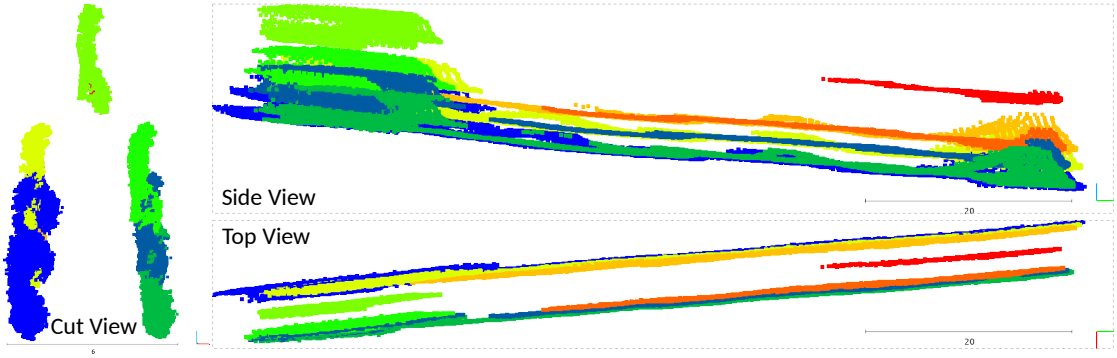


Figure 7.9: Power lines obtained with the default sensors configuration.

With such uncertainty in the estimated position, the mathematical approximation of the power line to a catenary curve was impossible to obtain. Even if, at some point, the catenary parameters could be obtained, the estimated height variation would make them not suitable for the next measurements of the line.

As it was referred in section 7.1, the objective of this dataset was the mapping of the stockpile. Thereby, the data of the high-accuracy set of sensors was available and properly timestamped with the Velodyne data. In order to use this dataset to test the algorithm, the post-processed pose estimation of the UAV was adapted and used to create the frame relations over time. The pose estimation was available at a rate of 1 kHz. The algorithm's resultant power line points using that estimation is exposed in figure 7.10.

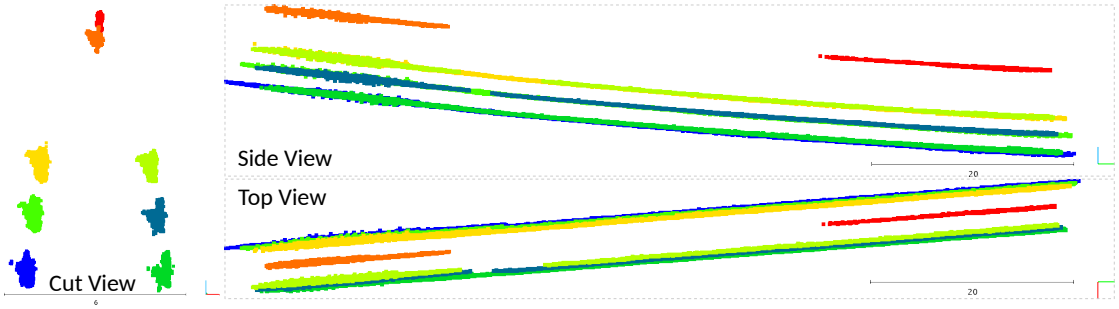


Figure 7.10: Power lines obtained with the high-accuracy sensors configuration.

The resultant points are far better than the ones presented in figure 7.9, however, they still have a small estimation uncertainty at some points, which does not happen in the post-processed map. This effect is generated by the fact that only one pose is considered for the whole scan, contrary to what happens in the mapping survey case, where each point is time and pose labeled individually.

Applying the methodology described in subsection 5.2.3, the algorithm was capable of detecting 8 distinct power lines in the whole data set. As previously known, there were present 6 power lines and a guard cable (considered also a power line by the algorithm), which means that it has generated one line more. Analyzing figure 7.11, it can be seen that the extra line was created by a detection break in the guard cable, where the algorithm was not capable of merging them due to their large distance (> 30 meters).

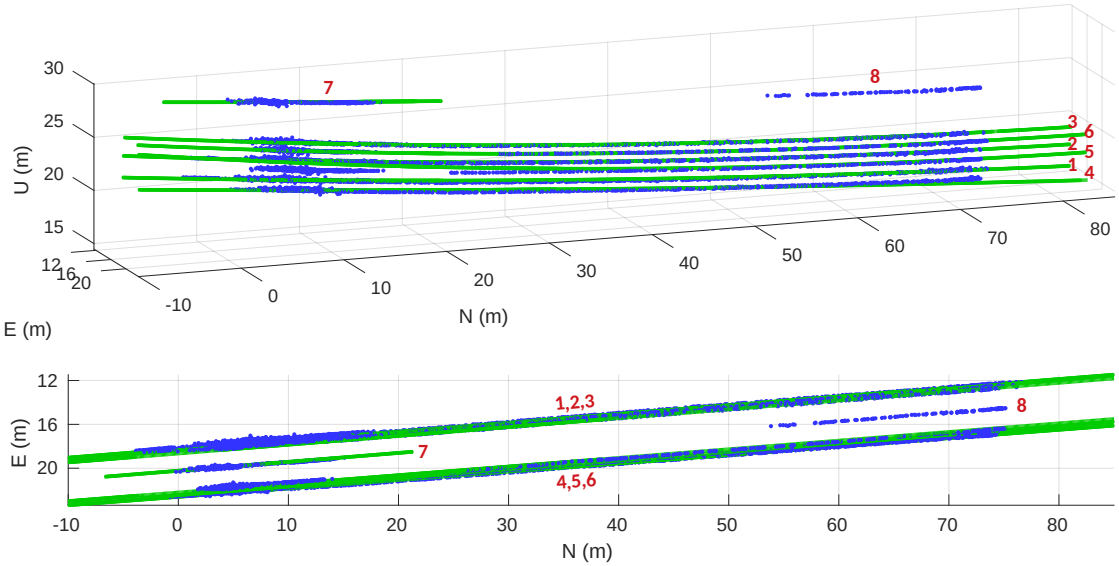


Figure 7.11: Estimated power line models (green) and line segments centers (blue).

The power line modeling was based in the direction and center points of all the line segments associated to the same model. In figure 7.11, the valid power models obtained are displayed in green and the line segment centers in blue. The red numbers correspond to a labeling of the detected power lines to match the results presented below. The top graph shows the power lines in perspective, while the bottom graph corresponds to their horizontal projection.

From figure 7.11 can also be perceived that power line cluster 8 has no mathematical model associated. The algorithm was not capable of reaching a valid model due to the straightness of the segment, having an almost constant vertical slope. This situation was already discussed in subsection 5.2.3.2, where was imposed the constraint $\theta_{first} \neq \theta_{last}$ in equation 5.22. For the remaining clusters, this fact has also caused some differences in the model estimation of the other segments, due to the sensibility of the value of a to a near null denominator, in equation 5.22.

The power lines were modeled as a straight line in the horizontal plane. Their direction was obtained from the average of the correspondent line segments direction. Their vertical model was an approximation to a catenary curve, using the method described in subsection 5.2.3.2, based in [29]. When a valid set of the catenary parameters a, b and c was found, if the vertical error of the line segments centers to the model was reduced, at that moment, the set was associated to the line. Otherwise, the previously estimated parameters were kept.

Figure 7.12 shows the vertical model of the power lines. For better visualization, the line was divided into two parts along its span.

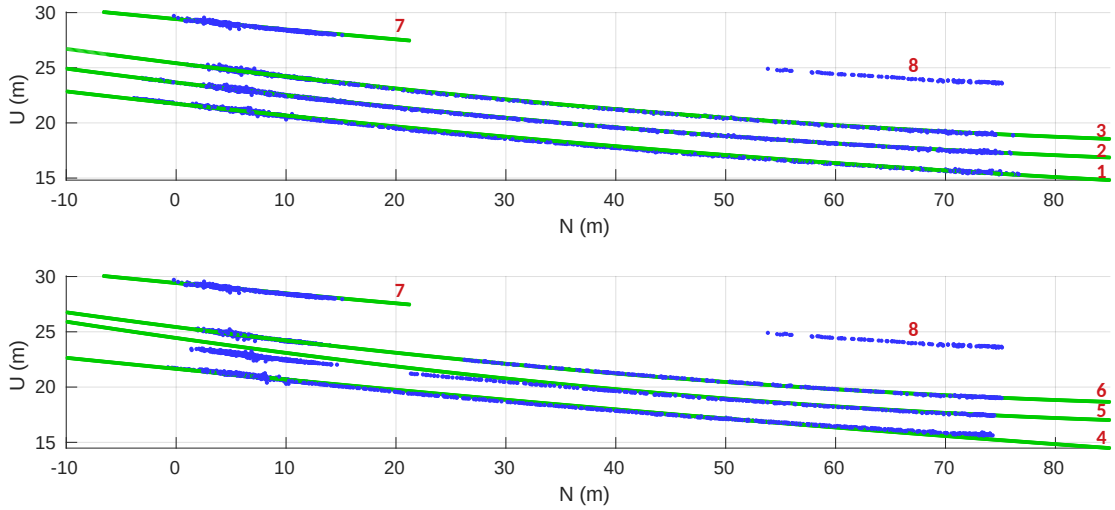


Figure 7.12: Vertical analysis of estimated power lines.

The power line clusters 1, 2 and 3 (represented in the top graph) have a vertical model that is better adjusted to their data points when compared to the clusters 4, 5 and 6 (represented in the bottom graph). This happened because the power lines 1, 2 and 3 were continuously seen from the moment when they become to be detected. Unlike them, the other lines were only partially detected in the first part of the flight (until N between 10 and 20 meters). The rest of their line segments were sequentially merged in an inverse way (decreasing N), when the UAV was returning to the initial point. This means that the final model has resulted from the combination of, at least, two distinct models, and no better valid model was found after the merging.

In table 7.2 are listed the parameters of the detected power lines, associated with the error of the fitting. The value of ψ represents the horizontal angular direction of the lines with respect to the North. E_m, N_m and U_m are the mean values of the line segment centers used to fit the power lines.

Apart from line 5, with an absolute vertical mean error of $0,49$ meters, all the other lines have an estimation with an average error lesser than $0,07$ meters. In the horizontal fitting, all the mean errors are placed below the $0,14$ meters. The value of ψ is nearly the same for all lines. As the calculation of b and c are dependent on a, it can be observed that the values of those parameters have some fluctuation. This happens due to the similar vertical slope of the line segments, as explained above.

Table 7.2: Power line models parameters

	Horizontal						**Vertical**				
	Error		*Parameters*				*Error*		*Parameters*		
Line	$\mu(m)$	$\sigma^2(m^2)$	ψ(deg)	$E_m(m)$	$N_m(m)$	$U_m(m)$	$\mu(m)$	$\sigma^2(m^2)$	a	b	c
1	0,12	0,0100	-4,55	16,14	24,86	19,35	0,07	0,0080	-1534,07	167,99	1546,67
2	0,09	0,0111	-4,57	16,39	24,29	21,26	0,04	0,0036	-1030,92	126,04	1046,99
3	0,14	0,0151	-4,61	16,58	24,20	22,98	0,05	0,0046	-957,37	121,02	975,26
4	0,07	0,0070	-4,37	20,54	26,68	19,27	0,07	0,0239	-3105,51	304,11	3112,29
5	0,12	0,0142	-4,50	20,66	21,98	21,51	0,49	0,0726	-780,87	111,93	797,44
6	0,12	0,0134	-4,51	20,42	21,88	23,24	0,04	0,0035	-877,53	113,69	895,73
7	0,06	0,0042	-4,65	19,71	6,62	28,78	0,05	0,0052	-3004,90	285,93	3020,77
8	0,03	0,0009	-4,34	15,09	67,96	23,99	-	-	-	-	-

Although some of the models might be not precisely fitted to the data points, their expansion into space allows a prediction of where the power lines are placed in the global frame. Exporting the values of the power line parameters obtained, higher-level algorithms can evaluate the reliability of the models and expand them accordingly to their needs. In figure 7.13 are represented the obtained power line models in the Google Earth. It can be seen that the lines lie between the two associated pylons, connecting to them when the models are expanded. This confirms the validity of the modeling.

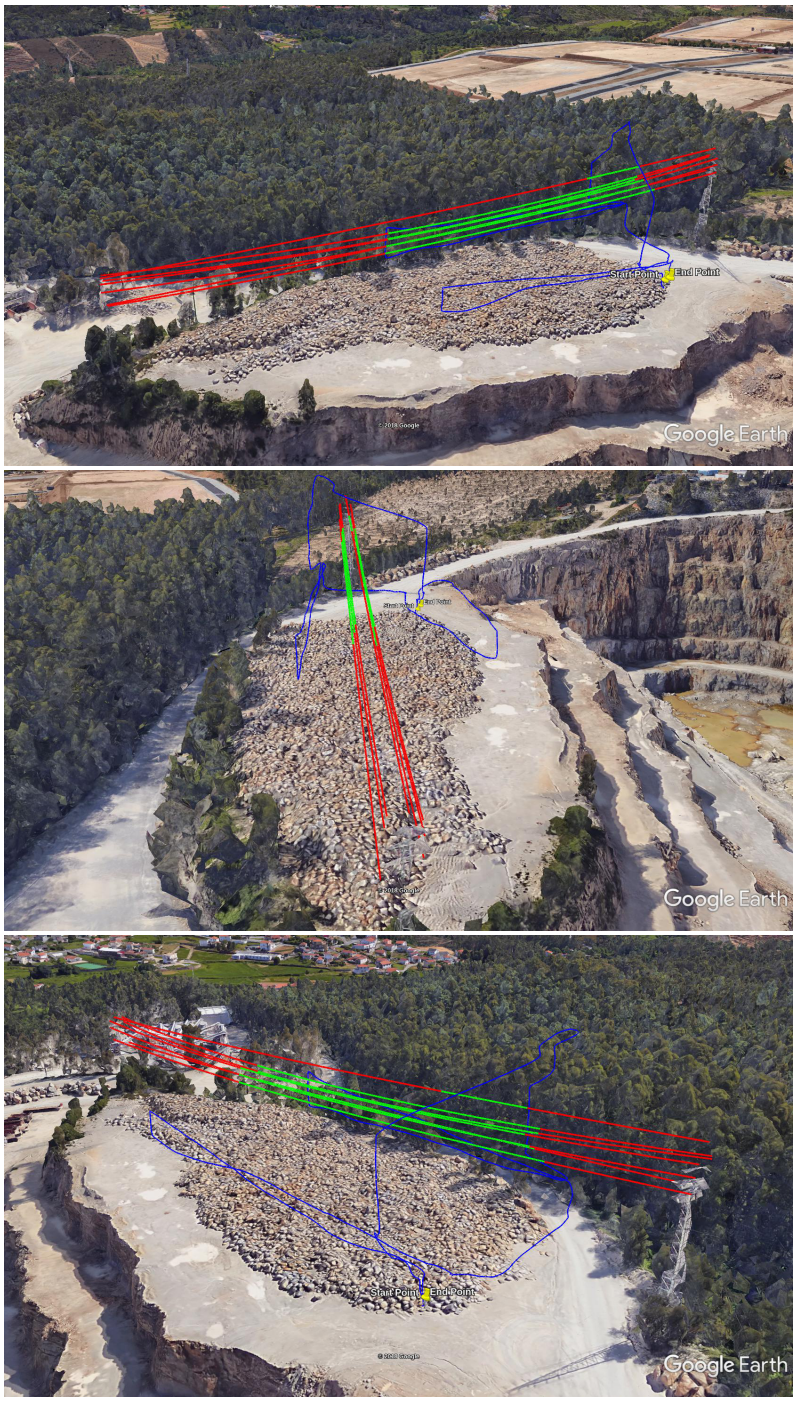


Figure 7.13: Power line models represented in Google Earth. Estimated models, in green, with their expansion in space, in red. UAV trajectory in blue.

7.3.4 Performance Evaluation

In the previous sections of this chapter, the analysis was focused on the outputs generated by the several steps of the algorithm. Here is evaluated the quality of the power line points classification and the processing time of each step.

7.3.4.1 Line Points Classification

Running the complete dataset, the scene perceived by the LiDAR sensor was composed by almost 35 millions of points. From those, 86 887 points corresponded to power lines (value obtained by manual labeling). In figure 7.14 is represented the whole point cloud, colored by point type, based in the output of the segmentation step of the algorithm. It can be seen that the area near to the UAV's takeoff and landing spots is mainly planar. In their majority, the points correspondent to the stockpiles and vegetation are labeled as *undefined*.

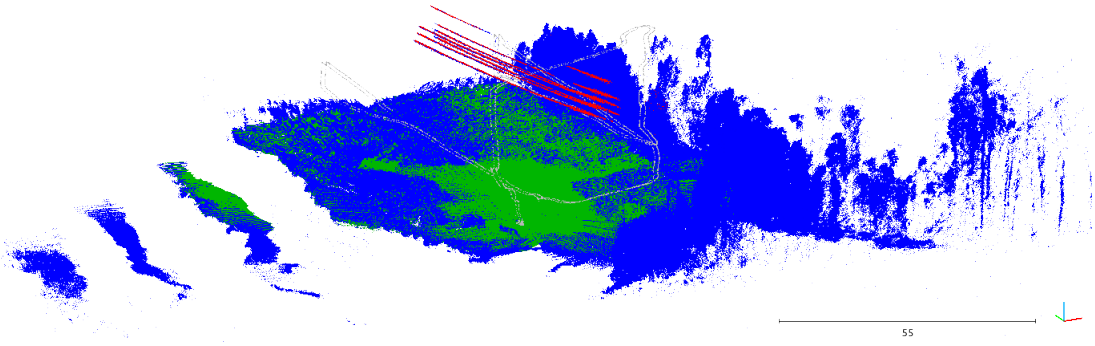


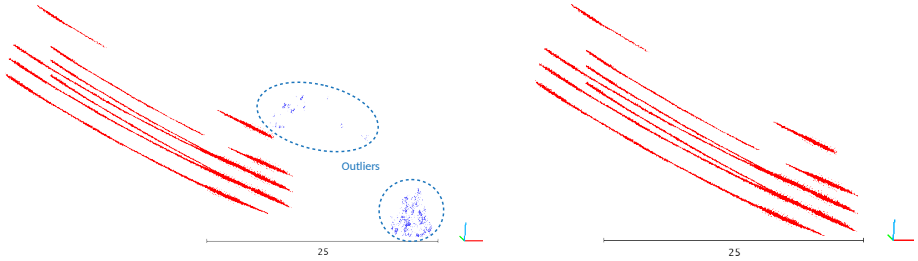
Figure 7.14: Complete dataset point cloud colored by point type. In blue are the *undefined* points, green the *planar* structures, and the *potential lines* are represented in red. The UAV trajectory is depicted in light gray.

The performance evaluation of the PL^{2}DM considers only the points correspondent to power lines, once that their detection is the main objective of this dissertation. Following the strategy in [130], the completeness (C_m) and correctness (C_r) are given by:

$$C_m = \frac{TP}{TP + FN} \tag{7.1}$$

$$C_r = \frac{TP}{TP + FP} \tag{7.2}$$

Being TP the true positive results, i.e., the points correctly labeled as *potential lines*, FN the false negatives (true power line points not classified as *potential lines*), and FP the false positives or the points wrongly labeled as *potential lines*.



(a) *Potential line* points. (b) Used *potential line* points.

Figure 7.15: *Potential line* type points. Outliers depicted in blue.

Regarding the outputs of the segmentation step, the algorithm has classified 69 914 points as *potential lines*. However, as depicted in figure 7.15(a), some FP points were generated (1 038 points). For this case, the number of the manually labeled power line points corresponds to the $TP + FN$ value. The values of correctness (C_{r_class}) and completeness (C_{m_class}) of the point classification are:

$$C_{m\_class} = \frac{69\,914 - 1\,038}{86\,887} = 79,27\% \tag{7.3}$$

$$C_{r\_class} = \frac{69\,914 - 1\,038}{69\,914} = 98,52\% \tag{7.4}$$

During the line fitting process, some of the *potential line* points are not considered. The figure 7.15(b) contains the valid *potential line* points used to construct the power line models. Thus, its completeness (C_{m_fit}) and correctness (C_{r_fit}) can be evaluated against the received points. Here, the value of $TP + FN$ corresponds to the inliers of figure 7.15(a) (68 876 points). For this case, the generated power line points (figure 7.15(b)) has no FN points and 66 624 TP ones.

$$C_{m\_fit} = \frac{66\,624}{68\,876} = 96,73\% \tag{7.5}$$

$$C_{r\_fit} = \frac{66\,624}{66\,624} = 100\% \tag{7.6}$$

From the values of C_{m_class}, C_{r_class}, C_{m_fit}, and C_{r_fit}, is possible to observe that the main limitation of the algorithm is related with the point classification. More than 20% of the available power line points were wrongly classified and not used to create the models of the lines. Even without those points, the majority of the lines were modeled with an error below 10 centimeters (table 7.2). Other noticeable property of the line fitting and modeling steps is the fact of being able to reject outliers.

In order to evaluate the advantage of using the power line model feedback for the segmentation (figure 7.7), is made a comparison between this method and the one with no feedback. The completeness ($C_{m\_no\_feed}$) and correctness ($C_{r\_no\_feed}$) are calculated and compared with C_{m_class} and C_{r_class}. Without feedback, the algorithm has generated 1 022 FP points and 27 091 TP. Calculating the values of $C_{m\_no\_feed}$ and $C_{r\_no\_feed}$:

$$C_{m\_no\_feed} = \frac{27\,091}{86\,887} = 31,18\% \tag{7.7}$$

$$C_{r\_no\_feed} = \frac{27\,091}{27\,091 + 1\,022} = 96,36\% \tag{7.8}$$

Comparing the values of $C_{m\_no\_feed}$ and C_{m_class} is noticeable a huge difference on the completeness value when using the feedback of the models. The value increases more than 48% when the previous power lines detected are considered. For the correctness, the difference is not so evident. The value of $C_{r\_no\_feed}$ is slightly smaller than C_{r_class}. The number of FP is almost constant for both conditions, being the difference in the correctness caused by the great diminishing of the TP points. In table 7.3 are listed all the obtained values for the completeness and correctness.

Table 7.3: Point classification performance

Classification				*Line Fit*	
$C_{m\_no\_feed}$	$C_{r\_no\_feed}$	C_{m_class}	C_{r_class}	C_{m_fit}	C_{r_fit}
$31,18\%$	$96,36\%$	$79,27\%$	$98,52\%$	$96,73\%$	100%

7.3.4.2 Processing Time

In order to achieve an online power line detection algorithm, the processing time is a crucial property. During the used dataset, Velodyne VLP-16 was providing data at an approximated rate of 10 Hz and working in the single *Strongest* return mode. Knowing this, the algorithm will be real-time capable if processes the data and generates outputs in less than 100 ms.

Due to the quantity of data involved, the segmentation step is the most crucial in terms of the overall processing time of the algorithm. In figure 7.16 is presented the processing time of the segmentation layer in terms of the number of data points. As the LiDAR was mounted below the UAV's frame, almost half of its data points is neither valid nor useful. Returning a maximum of 30 000 points per scan, with this mounting configuration, this mean that, from the outset, having a number of points above 15 000 in one scan is almost impossible.

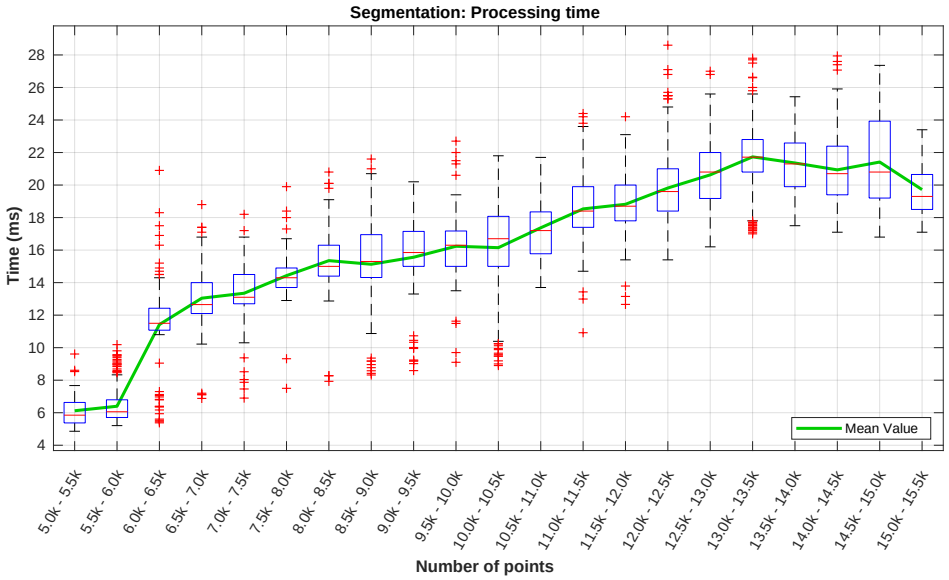


Figure 7.16: Processing time of the segmentation step.

Figures 7.16 and 7.17 support what was said above. There were no occurrences of scans with more than 15 500 points. Regarding the processing time (figure 7.16), it is visible an increase with the number of data points. This was expected due to the fact of the segmentation be a loop-based algorithm. The highest processing time registered was near the 29 *ms*, while the highest mean time (in green) is below 22 *ms*. For the same number of points, the variation of the processing time values is closely related with the number of neighbors associated.

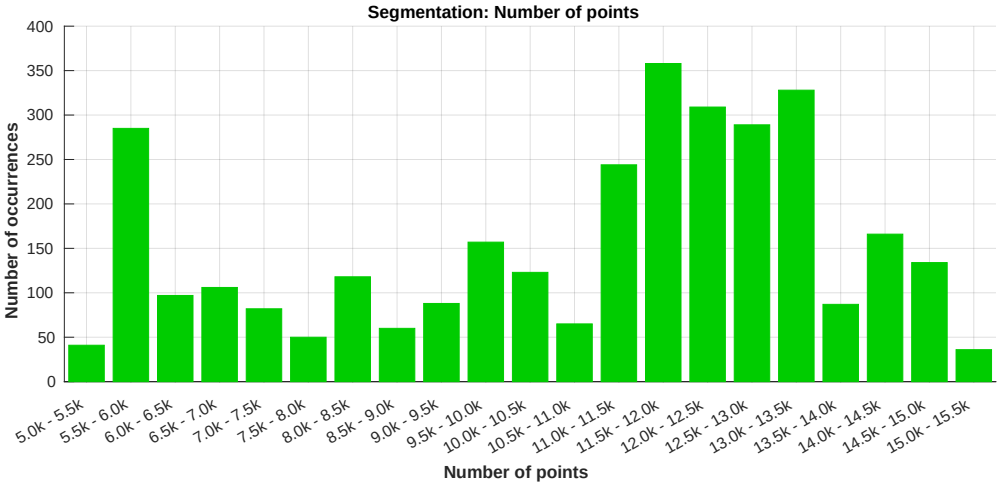


Figure 7.17: Number of points occurrence in the segmentation step.

In figure 7.17 can be observed that the LiDAR was returning a number of points between 11 000 and 13 500 in most of the time. The peak registered in the $5 - 6\,000$ points is related to the time while the UAV was on the ground.

The results obtained for the line detection step are depicted in figure 7.18. The number of points that are passed to this step is almost always under a hundred, being noticed an increase of the processing time with the incoming points. In the left graph can be seen that the processing time is almost negligible when compared to the segmentation processing times. The top time registered was under the 160 μs.

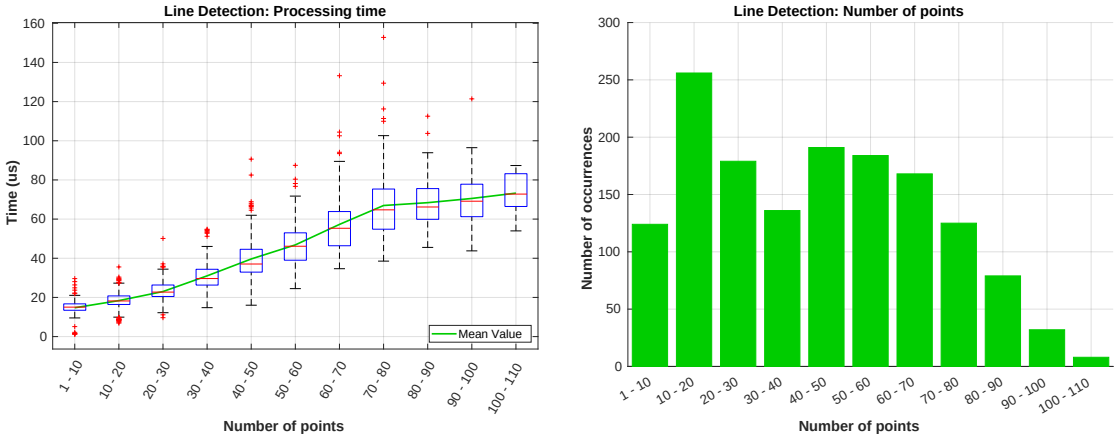


Figure 7.18: Processing time and number of points occurrences in the line detecting.

In the last step of the algorithm, the power lines modeling, were obtained the results exposed in figure 7.19.

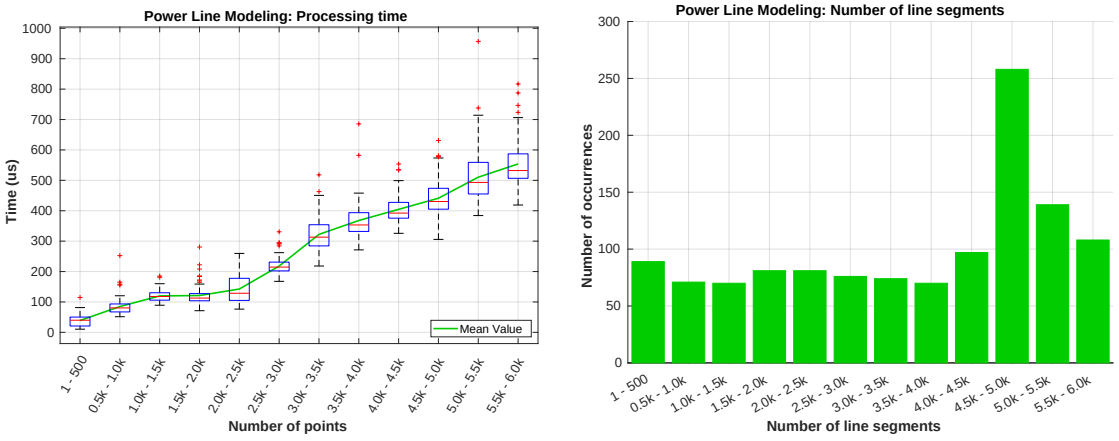


Figure 7.19: Processing time and number of line segments for power line modeling.

After being associated, the line segments of the power line clusters are not removed. This makes the algorithm having a growing processing time over the flight duration. In the left graph can be verified that, during all the dataset, the processing time has never overpassed the 1 ms. This constant increase in the time value is related to the evaluation of the vertical error whenever some new valid parameters a, b, and c are found. In the right graph is observable a dominance in the occurrence of a number of line segments between 4 500 and 5 000. This may refer to a period of the flight where few new line segments were added to the power line models at each iteration. In the rest of the flight, the line segments number had an almost constant growth.

From the obtained results for all the algorithm's steps (figure 7.20), with respect to processing times, it can be concluded that the current implementation is able of processing up to nearly 500 000 points per second in real-time. This allows its application with other LiDAR sensors that provide more data or using the VLP-16 in dual-return mode. Using the dual-return mode can be useful for detecting the power lines.

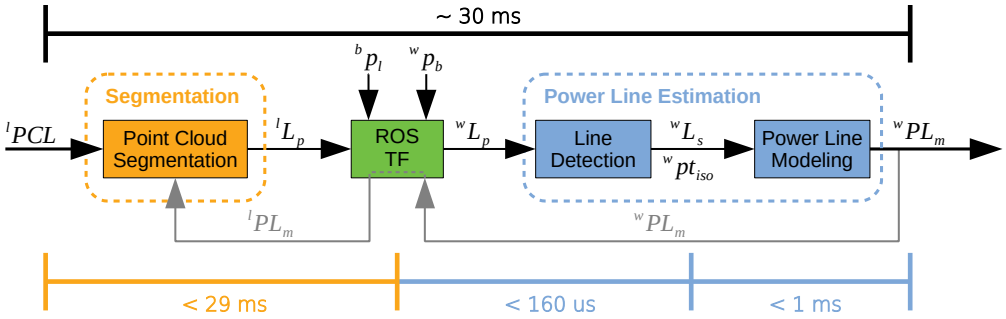


Figure 7.20: Total processing times of the PL^{2}DM.

In a long flight or in a scenario with several power lines, the constant increase of the processing time corresponding to the power line modeling might be limiting. To overcome this, some down-sampling strategy of the line segments can be applied when the confidence in the estimated model is high.

For running the dataset and the developed algorithm was used a computer with an Intel Core i7 4720HQ processor and 8 GB of RAM. The processing was made in the Ubuntu 18.04 LTS operating system, inside the ROS Melodic Morenia environment. Migrating this code to the STORK's onboard computer may cause an increase of the processing times, however, the developed code has room for improvements and optimizations.

When using the proposed algorithm in a situation where the processing time is close to the available time, there are some considerations that must be taken. The algorithm was designed for having a parallel processing of the segmentation and line detection and

modeling (figure 7.21). Whenever the segmentation is performed, if some *potential line* points are found, the other thread is triggered (blue dashed line in figure 7.21). From the line detection and modeling thread are returned the line models to the next scan to aid the segmentation (green dashed line in figure 7.21).

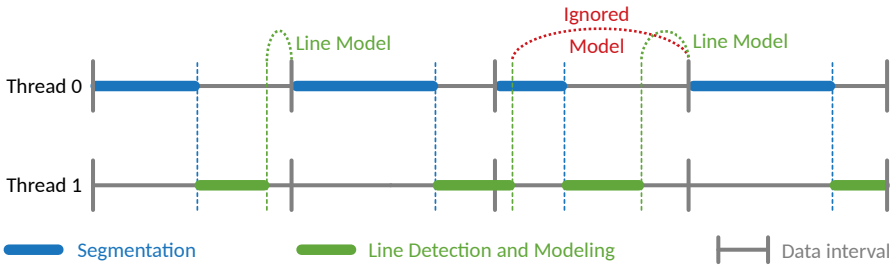


Figure 7.21: Effect of slow processing.

If the combined time of the segmentation and the line detecting is greater than the available time (imposed by the sensor's data rate), the data may desynchronize. The next scan will use the same line model as the previous and the new line model can be either ignored or delayed for the subsequent scans. In the case depicted at figure 7.21, in red, the generated line model is ignored by the next scan due to the existence of newer data. However, if no new model is generated, is used the one obtained from the two (or more) previous scans data.

Chapter 8

Conclusions and Future Work

This dissertation has focused on the development of an algorithm capable of detecting and modeling surrounding power lines in real-time, the Power Line LiDAR-based Detection and Modeling (PL^2DM). The incoming data for processing is provided by a LiDAR sensor mounted in a multirotor Unmanned Aerial Vehicle (UAV). Its outputs and performance were evaluated using a dataset containing several types of structures, acquired during a mapping survey in the Malaposta quarry.

From the results presented in chapter 7, the main objectives of this dissertation were fulfilled. The algorithm is capable of segmenting an input point cloud, detect power lines points and, from there, generate line segments that are merged to construct the final power line model. The whole process has shown to be suitable for being applied in tasks with real-time requirements.

In the classification of power line points, the algorithm fails to label some of them as *potential line*. This is due to the difficulty in discerning between power lines and vegetation points, labeling them as *undefined*. Nonetheless, the power line modeling is still reached with a low error associated in most of the cases. The knowledge of those line models for the next scans segmentation has turned out to be very useful in helping the point classification, increasing in 48% the number of power line points used for modeling.

To the author's knowledge until the writing of this document, the contribution to the segmentation's methodology is derived from a novel approach based on planar properties of the structures. In the line detection and modeling steps, there is also a contribution to the existent methods. It applies a methodology that does not rely on the Hough Transform (HT) to detect lines nor assumes a flight direction parallel to the power lines. Instead, it uses collinearity properties and accepts lines from any direction. Another advantage of the PL^2DM is using a low number of thresholds for tunning (table 7.1).

Although the concept of the algorithm was validated with the used dataset, it still needs to be submitted to others with different conditions. One of those conditions may be the detection of multiple spans of a power line set, or even in scenarios with multiple distinct lines. Having different types of backgrounds (vegetation, rocks, buildings, ...) or lines with various slopes are other interesting conditions to validate the method. Its online operation during a real flight shall also be evaluated. The outputs generated by the PL^{2}DM can then be compared to a ground truth geo-referenced point cloud generated, for example, by a FARO Laser Scanner[1].

For trying to improve and refine the point classification, the use of the LiDAR in dual-return mode can be tested. However, the algorithm would need adaptations to consider the double point existence for the same direction.

When migrating the algorithm to new processing units, the running time needs to be re-evaluated to either consider or not the effect of slow processing, depicted in figure 7.21. The increase of the time expended in the line modeling needs to be handled with care. The development of a down-sampling of the line segments has to be pondered if that increase turns out to be limitative to the algorithm's performance. An alternative implementation in a Graphics Processing Unit (GPU) can be advantageous to the performance of the algorithm, specially in the segmentation step (see figure 7.20). The scan could be divided into several parts and analyzed in parallel, being then merged by processing the contiguous limits of each part.

On top of this work, several higher-level algorithms can be developed using the data provided. The power line model is useful to obstacle detection, collision avoidance, or line following algorithms. At the height of the power line, the *undefined* points usually correspond to vegetation. Associating this with the obtained models allows the application of vegetation clearance anomalies detectors. The horizontal planar structures are useful to detect possible safe landing spots for the UAV.

The sequence of developments that led to this dissertation's work has resulted in a scientific publication of the paper "Collision avoidance for safe structure inspection with multirotor UAV" on the European Conference on Mobile Robots (ECMR) 2017, in Paris [40]. The mapping survey used to test the PL^{2}DM is part of the scientific work "Unmanned geo-technology systems: aerial imagery, survey and mapping of georesources and maritime structures", submitted for review on the 3rd International Conference on Information Technology in Geo-Engineering (ICITG) 2019, in Guimarães [163].

[1] `https://www.faro.com/products/construction-bim-cim/faro-focus/`

Bibliography

[1] Chun Fui Liew, Danielle DeLatte, Naoya Takeishi, and Takehisa Yairi. Recent Developments in Aerial Robotics: {A} Survey and Prototypes Overview. *CoRR*, abs/1711.10085, 2017.

[2] Bas Vergouw, Huub Nagel, Geert Bondt, and Bart Custers. *Drone Technology: Types, Payloads, Applications, Frequency Spectrum Issues and Future Developments*, pages 21–45. T.M.C. Asser Press, The Hague, 2016.

[3] Autonomous Systems Laboratory. "FALCOS". [Online]. Available: `http://archive.osvaldosousa.com/lsa/falcos/images/falcos{_}div/Dsc02308b.jpg`. [Accessed: 28-Aug-2018].

[4] NASA. "Greased Lightning". [Online]. Available: `https://technology.nasa.gov/t2media/tops/img/LAR-TOPS-241/GLback.jpg`. [Accessed: 28-Aug-2018].

[5] DelFly. "DelFly Micro". [Online]. Available: `http://www.delfly.nl/images/modified_micro.jpg`. [Accessed: 28-Aug-2018].

[6] Zujian Xu, Feng Yang, Yong Huang, Zizheng Wang, and Yanjing Liu. Lidar applications in the electrical power industry. *Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci*, 36:137–140, 2008.

[7] ULC Robotics. "Unmanned Aerial Electric Utility Inspection Services - ULC Robotics". [Online]. Available: `http://ulcrobotics.com/services/electric-utility-unmanned-aerial-inspection/`, 2017. [Accessed: 05-Jul-2018].

[8] Sonal Patel. "A Bird's-Eye View: Drones in the Power Sector". [Online]. Available: `http://www.powermag.com/a-birds-eye-view-drones-in-the-power-sector/`, 2018. [Accessed: 05-Jul-2018].

[9] Joel Pulkkinen. Laserkeilauksen soveltaminen sähköverkon kunnossapidossa; Laser scanning in electricity network maintenance. G2 pro gradu, diplomityö, 2015.

[10] A Pagnano, M Höpf, and R Teti. A Roadmap for Automated Power Line Inspection. Maintenance and Repair. *Procedia CIRP*, 12:234–239, 2013.

[11] Leena Matikainen, Matti Lehtomäki, Eero Ahokas, Juha Hyyppä, Mika Karjalainen, Anttoni Jaakkola, Antero Kukko, and Tero Heinonen. Remote sensing methods for power line corridor surveys. *ISPRS Journal of Photogrammetry and Remote Sensing*, 119:10–31, 2016.

[12] Chi Chen, Bisheng Yang, Shuang Song, Xiangyang Peng, and Ronggang Huang. Automatic Clearance Anomaly Detection for Transmission Line Corridors Utilizing UAV-Borne LIDAR Data. *Remote Sensing*, 10(4), 2018.

[13] S J Mills, M P Gerardo Castro, Z Li, J Cai, R Hayward, L Mejias, and R A Walker. Evaluation of Aerial Remote Sensing Techniques for Vegetation Management in Power-Line Corridors. *IEEE Transactions on Geoscience and Remote Sensing*, 48(9):3379–3390, 2010.

[14] Junaid Ahmad, Aamir Saeed Malik, Mohd Faris Abdullah, Nidal Kamel, and Likun Xia. A novel method for vegetation encroachment monitoring of transmission lines using a single 2D camera. *Pattern Analysis and Applications*, 18(2):419–440, 2015.

[15] D W Wanik, J R Parent, E N Anagnostou, and B M Hartman. Using vegetation management and LiDAR-derived tree height data to improve outage predictions for electric utilities. *Electric Power Systems Research*, 146:236–245, 2017.

[16] Yoonseok Jaw and Gunho Sohn. Wind adaptive modeling of transmission lines using minimum description length. *ISPRS Journal of Photogrammetry and Remote Sensing*, 125:193–206, 2017.

[17] Junaid Ahmad, Aamir Saeed Malik, Likun Xia, and Nadia Ashikin. Vegetation encroachment monitoring for transmission lines right-of-ways: A survey. *Electric Power Systems Research*, 95:339–352, 2013.

[18] I Ituen and Gunho Sohn. The way forward: Advances in maintaining right-of-way of transmission lines. *Geomatica*, 64:451–462, 2010.

[19] J Katrasnik, F Pernus, and B Likar. A Survey of Mobile Robots for Distribution Power Line Inspection. *IEEE Transactions on Power Delivery*, 25(1):485–493, jan 2010.

[20] R K Aggarwal, A T Johns, J.A.S.B Jayasinghe, and W Su. An overview of the condition monitoring of overhead lines. *Electric Power Systems Research*, 53(1):15–22, 2000.

[21] Friedrich Kiessling, Peter Nefzger, João Felix Nolasco, and Ulf Kaintzyk. *Overhead Power Lines*. Power Systems. Springer Berlin Heidelberg, Berlin, Heidelberg, 2003.

[22] T Santos, M Moreira, J Almeida, A Dias, A Martins, J Dinis, J Formiga, and E Silva. PLineD: Vision-based power lines detection for Unmanned Aerial Vehicles. In *2017 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, pages 253–259, 2017.

[23] Higgins Fire District. "Vegetation Encroachment". [Online]. Available: `http://www.higginsfire.org/images/PowerlinesTreeDown.jpg`. [Accessed: 29-Aug-2018].

[24] Edvard. "Defective Insulator". [Online]. Available: `https://electrical-engineering-portal.com/wp-content/uploads/2015/10/flashed-230-kv-porcelain-insulator.jpg`. [Accessed: 29-Aug-2018].

[25] C C Whitworth, A W G Duller, D I Jones, and G K Earp. Aerial video inspection of overhead power lines. *Power Engineering Journal*, 15(1):25–32, 2001.

[26] L E B Eriksson, J E S Fransson, M J Soja, and M Santoro. Backscatter signatures of wind-thrown forest in satellite SAR images. In *2012 IEEE International Geoscience and Remote Sensing Symposium*, pages 6435–6438, 2012.

[27] Zhang JiXian, HUANG Guoman, and Liu Jiping. SAR remote sensing monitoring of the Yushu Earthquake disaster situation and the information service system. 2010.

[28] Y Kobayashi, G G Karady, G T Heydt, and R G Olsen. The Utilization of Satellite Images to Identify Trees Endangering Transmission Lines. *IEEE Transactions on Power Delivery*, 24(3):1703–1709, 2009.

[29] R A McLaughlin. Extracting transmission lines from airborne LIDAR data. *IEEE Geoscience and Remote Sensing Letters*, 3(2):222–226, 2006.

[30] Lingli Zhu and Juha Hyyppä. Fully-Automated Power Line Extraction from Airborne Laser Scanning Point Clouds in Forest Areas. *Remote Sensing*, 6(11):11267–11282, 2014.

[31] Bo Guo, Qingquan Li, Xianfeng Huang, and Chisheng Wang. An Improved Method for Power-Line Reconstruction from Point Cloud Data. *Remote Sensing*, 8(1), 2016.

[32] Changming Sun, Ronald Jones, Hugues Talbot, Xiaoliang Wu, Kevin Cheong, Richard Beare, Michael Buckley, and Mark Berman. Measuring the distance of vegetation from powerlines using stereo vision. *ISPRS Journal of Photogrammetry and Remote Sensing*, 60(4):269–283, 2006.

[33] Angelika Wronkowicz. Automatic fusion of visible and infrared images taken from different perspectives for diagnostics of power lines. *Quantitative InfraRed Thermography Journal*, 13(2):155–169, 2016.

[34] Alejandro Tache Gregory R. Stockton. Advances in applications for aerial infrared thermography, 2006.

[35] Markus Ax, Stefan Thamke, Lars Kuhnert, and Klaus Dieter Kuhnert. UAV Based Laser Measurement for Vegetation Control at High-Voltage Transmission Lines. In *Advances in Power and Electrical Engineering*, volume 614 of *Advanced Materials Research*, pages 1147–1152. Trans Tech Publications, 2013.

[36] D Jones. Power line inspection - a UAV concept. In *2005 The IEE Forum on Autonomous Systems (Ref. No. 2005/11271)*, pages 8 pp.–, nov 2005.

[37] Toussaint Kristopher, Pouliot Nicolas, and Montambault Serge. Transmission line maintenance robots capable of crossing obstacles: State-of-the-art review and challenges ahead. *Journal of Field Robotics*, 26(5):477–499, 2009.

[38] Haiyan Guan, Yongtao Yu, Jonathan Li, Zheng Ji, and Qi Zhang. Extraction of power-transmission lines from vehicle-borne lidar data. *International Journal of Remote Sensing*, 37(1):229–247, 2016.

[39] Hydro-Québec. "LineScout". [Online]. Available: `http://news.hydroquebec.com/media/cache/b2/5a/b25a385b7d8c63c2bbf38958f48abb7a.jpg`. [Accessed: 29-Aug-2018].

[40] F Azevedo, A Oliveira, A Dias, J Almeida, M Moreira, T Santos, A Ferreira, A Martins, and E Silva. Collision avoidance for safe structure inspection with

multirotor UAV. In *2017 European Conference on Mobile Robots (ECMR)*, pages 1–7, 2017.

[41] INESC TEC. "OTUS". [Online]. Available: `http://www.strongmar.eu/site/otus-91`. [Accessed: 31-Aug-2018].

[42] INESC TEC. "STORK I". [Online]. Available: `http://www.strongmar.eu/site/stork-i-106`. [Accessed: 31-Aug-2018].

[43] INESC TEC. "TURTLE". [Online]. Available: `http://www.strongmar.eu/site/album/turtle-819`. [Accessed: 29-Aug-2018].

[44] INESC TEC. "ROAZ II". [Online]. Available: `http://www.strongmar.eu/site/album/roaz-ii-783`. [Accessed: 29-Aug-2018].

[45] Diogo Coelho Pedrosa. Control-law for oil spill mitigation with a team of heterogeneous autonomous vehicles. Master's thesis, Instituto Superior de Engenharia do Porto, Departamento de Engenharia Eletrotécnica, Porto, Portugal, 7 2018.

[46] Diogo Pedrosa, André Dias, Alfredo Martins, José Almeida, and Eduardo Silva. Control-law for Oil Spill Mitigation with an Autonomous Surface Vehicle. In *OCEANS'18 MTS/IEEE Kobe / Techno-Ocean 2018*, 2018.

[47] J Formiga, J Dinis, J Fialho, F Moreira, J Almeida, A Dias, E Silva, M Moreira, and T Santos. Field Experiments in Power Line Inspection with an Unmanned Aerial Vehicle. In *24th International Conference on Electricity Distribution (CIRED)*, 2017.

[48] P Sousa, A Ferreira, M Moreira, T Santos, A Martins, A Dias, J Almeida, and E Silva. ISEP/INESC TEC Aerial Robotics Team for Search and Rescue Operations at the EuRathlon Challenge 2015. In *2016 International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, pages 156–161, 2016.

[49] Alan F T Winfield, Marta Palau Franco, Bernd Brueggemann, Ayoze Castro, Miguel Cordero Limon, Gabriele Ferri, Fausto Ferreira, Xingkun Liu, Yvan Petillot, Juha Roning, Frank Schneider, Erik Stengler, Dario Sosa, and Antidio Viguria. euRathlon 2015: A Multi-domain Multi-robot Grand Challenge for Search and Rescue Robots. In Lyuba Alboul, Dana Damian, and Jonathan M Aitken, editors, *Towards Autonomous Robotic Systems*, pages 351–363, Cham, 2016. Springer International Publishing.

[50] Alan F T Winfield, Marta Palau Franco, Bernd Brueggemann, Ayoze Castro, Gabriele Ferri, Fausto Ferreira, Xingcun Liu, Yvan Petillot, Juha Roning, Frank Schneider, Erik Stengler, Dario Sosa, and Antidio Viguria. euRathlon and ERL Emergency: A Multi-domain Multi-robot Grand Challenge for Search and Rescue Robots. In Anibal Ollero, Alberto Sanfeliu, Luis Montano, Nuno Lau, and Carlos Cardeira, editors, *ROBOT 2017: Third Iberian Robotics Conference*, pages 263–271, Cham, 2018. Springer International Publishing.

[51] Yoonseok Jwa and Gunho Sohn. A Piecewise Catenary Curve Model Growing for 3D Power Line Reconstruction. *Photogrammetric Engineering & Remote Sensing*, 78(12):1227–1240, 2012.

[52] Matthew J. McGill and David OC. Starr. Lidar Remote Sensing. jan 2002.

[53] Brian M. Walsh and Samuel W. Spelsberg. "LIDAR — Remote Sensing Branch". [Online]. Available: `https://lasersdbw.larc.nasa.gov/tutorials/lidar/`, 2016. [Accessed: 15-Jul-2018].

[54] Dr. Barbara Mattson, J.D. Myers, and Phil Newman. "Doppler Shift". [Online]. Available: `https://imagine.gsfc.nasa.gov/features/yba/M31_velocity/spectrum/doppler_more.html`, 2016. [Accessed: 15-Jul-2018].

[55] VectorStock. "Electromagnetic Spectrum Infographic Diagram". [Online]. Available: `https://cdn.vectorstock.com/i/1000x1000/32/23/electromagnetic-spectrum-infographic-diagram-vector-19633223.jpg`. [Accessed: 28-Aug-2018].

[56] Bob Allen. Knowing Earth. 2017.

[57] G FIOCCO and L D SMULLIN. Detection of Scattering Layers in the Upper Atmosphere (60–140 km) by Optical Radar. *Nature*, 199:1275, sep 1963.

[58] Collis R T H. Lidar: A new atmospheric probe. *Quarterly Journal of the Royal Meteorological Society*, 92(392):220–230, 1966.

[59] Earl W Barrett and Oded Ben-Dov. Application of the Lidar to Air Pollution Measurements. *Journal of Applied Meteorology*, 6(3):500–515, 1967.

[60] Warren B Johnson. Lidar Applications in Air Pollution Research and Control. *Journal of the Air Pollution Control Association*, 19(3):176–180, 1969.

[61] P M Hamilton. The Application of a Pulsed-Light Rangefinder (Lidar) to the Study of Chimney Plumes. *Philosophical Transactions of the Royal Society of London. Series A, Mathematical and Physical Sciences*, 265(1161):153–172, 1969.

[62] James B. Abshire and David E. Smith. Space Lidar and Applications. jan 2001.

[63] C Hug. The scanning laser altitude and reflectance sensor, an instrument for efficient 3D terrain survey–ISPRS Comm. In *I Symposium" Primary data acquisition and evaluation", Como, Italia*, pages 12–16, 1994.

[64] W Sohne, O Heinze, C Hug, and U Kalberer. Positioning and orientation of a laser/radar-altimeter survey flight with GPS and INS. In *Proceedings of the Gyro Symposium*, pages 203–212, 1993.

[65] Christoph Hug. Combined use of laser scanner geometry and reflectance data to identify surface objects. In *Proceedings of the OEEPE Workshop*, pages 9–11, 1996.

[66] Christoph Hug. Extracting Artificial Surface Objects from Airborne Laser Scanner Data. In *Automatic Extraction of Man-Made Objects from Aerial and Space Images (II)*, pages 203–212. Birkhäuser Basel, Basel, 1997.

[67] Norbert Haala and Claus Brenner. Extraction of buildings and trees in urban environments. *ISPRS Journal of Photogrammetry and Remote Sensing*, 54(2-3):130–137, jul 1999.

[68] Hans-Gerd Maas. Fast determination of parametric house models from dense airborne laserscanner data. In *International Workshop on Mobile Mapping Technology*, volume 32, pages 1–6, 1999.

[69] Hans-Gerd Maas and George Vosselman. Two algorithms for extracting building models from raw laser altimetry data. *ISPRS Journal of Photogrammetry and Remote Sensing*, 54(2):153–163, 1999.

[70] Peter Axelsson. Processing of laser scanner data—algorithms and applications. *ISPRS Journal of Photogrammetry and Remote Sensing*, 54(2):138–147, 1999.

[71] Wolfgang Förstner. 3D-City Models : Automatic and Semiautomatic Acquisition Methods. 1999.

[72] Norbert Haala, Claus Brenner, and Karl-heinrich Anders. 3D Urban GIS From Laser Altimeter And 2D Map Data, 1998.

[73] Hiroshi Masaharu and Hiroyuki Hasegawa. Three-dimensional city modeling from laser scanner data by extracting building polygons using region segmentation method. *International Archives of Photogrammetry and Remote Sensing*, 33(B3/1; PART 3):556–562, 2000.

[74] Roland Geibel and Uwe Stilla. Segmentation of Laser Altimeter Data for Building Reconstruction : Different Procedures and Comparison. 2001.

[75] Paolo Gamba and Vittorio Casella. Model independent object extraction from digital surface models. *International Archives of Photogrammetry and Remote Sensing*, 33(B3/1; PART 3):312–319, 2000.

[76] Miao Wang and Yi-Hsing Tseng. Lidar data segmentation and classification based on octree structure. volume 2, pages 1–6, 2004.

[77] Miao Wang and Yi Hsing Tseng. Automatic 3D feature extraction from structuralized LIDAR data. In *Asian Association on Remote Sensing - 26th Asian Conference on Remote Sensing and 2nd Asian Space Conference, ACRS 2005*, volume 3, pages 1421–1429. 2005.

[78] Saja Abdul Rahman Kutty and Sonal Ayyappan. A novel technique for LiDAR data segmentation and three-dimensional space projection. In *2015 Annual IEEE India Conference (INDICON)*, pages 1–5. IEEE, dec 2015.

[79] Jie Shan and Aparajithan Sampath. Urban DEM Generation from Raw Lidar Data: A Labeling Algorithm and its Performance. 2051(February):217–226, 2005.

[80] D Steinhauser, O Ruepp, and D Burschka. Motion segmentation and scene classification from 3D LIDAR data. In *2008 IEEE Intelligent Vehicles Symposium*, pages 398–403, 2008.

[81] Frank Moosmann, Oliver Pink, and Christoph Stiller. Segmentation of 3D lidar data in non-flat urban environments using a local convexity criterion. In *2009 IEEE Intelligent Vehicles Symposium*, pages 215–220. IEEE, jun 2009.

[82] L Spinello, R Triebel, and R Siegwart. Multimodal detection and tracking of pedestrians in urban environments with explicit ground plane extraction. In *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1823–1829, 2008.

[83] Klaas Klasing, Dirk Wollherr, and Martin Buss. A clustering method for efficient segmentation of 3D laser data. In *2008 IEEE International Conference on Robotics and Automation*, pages 4043–4048. IEEE, may 2008.

[84] K. Klasing, D. Wollherr, and M. Buss. Realtime segmentation of range data using continuous nearest neighbors. In *2009 IEEE International Conference on Robotics and Automation*, pages 2431–2436. IEEE, may 2009.

[85] Rodrigo Paredes and Edgar Chávez. Using the k-Nearest Neighbor Graph for Proximity Searching in Metric Spaces. In Mariano Consens and Gonzalo Navarro, editors, *String Processing and Information Retrieval*, pages 127–138, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[86] Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, and Werner Stuetzle. *Surface reconstruction from unorganized points*, volume 26. ACM, 1992.

[87] K Klasing, D Althoff, D Wollherr, and M Buss. Comparison of surface normal estimation methods for range sensing applications. In *2009 IEEE International Conference on Robotics and Automation*, pages 3206–3211, 2009.

[88] M Pauly, M Gross, and L P Kobbelt. Efficient simplification of point-sampled surfaces. In *IEEE Visualization, 2002. VIS 2002.*, pages 163–170, nov 2002.

[89] Pauly Mark, Keiser Richard, and Gross Markus. Multi-scale Feature Extraction on Point-Sampled Surfaces. *Computer Graphics Forum*, 22(3):281–289, 2003.

[90] T Rabbani, F A van den Heuvel, and G Vosselman. Segmentation of point clouds using smoothness constraints. In H G Maas and D Schneider, editors, *ISPRS 2006 : Proceedings of the ISPRS commission V symposium Vol. 35, part 6 : image engineering and vision metrology, Dresden, Germany 25-27 September 2006*, volume 35, pages 248–253. International Society for Photogrammetry and Remote Sensing (ISPRS), 2006.

[91] Bastian Oehler, Joerg Stueckler, Jochen Welle, Dirk Schulz, and Sven Behnke. Efficient Multi-resolution Plane Segmentation of 3D Point Clouds. In Sabina Jeschke, Honghai Liu, and Daniel Schilberg, editors, *Intelligent Robotics and Applications*, pages 145–156, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[92] N J Mitra, A Nguyen, and L Guibas. Estimating Surface Normals in Noisy Point Cloud Data. In *special issue of International Journal of Computational Geometry and Applications*, volume 14, pages 261–276, 2004.

[93] Mincheol Yoon, Yunjin Lee, Seungyong Lee, Ioannis Ivrissimtzis, and Hans-Peter Seidel. Surface and normal ensembles for surface reconstruction. *Computer-Aided Design*, 39(5):408–420, 2007.

[94] Bao Li, Ruwen Schnabel, Reinhard Klein, Zhiquan Cheng, Gang Dang, and Shiyao Jin. Robust normal estimation for point clouds with sharp features. *Computers & Graphics*, 34(2):94–106, 2010.

[95] A Nurunnabi, D Belton, and G West. Robust segmentation for multiple planar surface extraction in laser scanning 3D point cloud data. In *Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012)*, pages 1367–1370, nov 2012.

[96] Abdul Nurunnabi, David Belton, and Geoff West. Robust Segmentation in Laser Scanning 3D Point Cloud Data. In *2012 International Conference on Digital Image Computing Techniques and Applications (DICTA)*, pages 1–8. IEEE, dec 2012.

[97] M. Himmelsbach, Felix v. Hundelshausen, and H.-J. Wuensche. Fast segmentation of 3D point clouds for ground vehicles. In *2010 IEEE Intelligent Vehicles Symposium*, pages 560–565. IEEE, jun 2010.

[98] M. Himmelsbach, T. Luettel, and H.-J. Wuensche. Real-time object classification in 3D point clouds using point feature histograms. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 994–1000. IEEE, oct 2009.

[99] Sebastian Thrun, Michael Montemerlo, and Andrei Aron. Probabilistic Terrain Analysis For High-Speed Desert Driving. In *Robotics: Science and Systems*, 2006.

[100] Viet Nguyen, Stefan Gächter, Agostino Martinelli, Nicola Tomatis, and Roland Siegwart. A comparison of line extraction algorithms using 2D range data for indoor mobile robotics. *Autonomous Robots*, 23(2):97–111, 2007.

[101] B. Douillard, J. Underwood, N. Kuntz, V. Vlaskine, A. Quadros, P. Morton, and A. Frenkel. On the segmentation of 3D LIDAR point clouds. In *2011 IEEE International Conference on Robotics and Automation*, pages 2798–2805. IEEE, may 2011.

[102] S Vasudevan, F Ramos, E Nettleton, H Durrant-Whyte, and A Blair. Gaussian Process modeling of large scale terrain. In *2009 IEEE International Conference on Robotics and Automation*, pages 1047–1053, 2009.

[103] Y Owechko, S Medasani, and T Korah. Automatic recognition of diverse 3-D objects and analysis of large urban scenes using ground and aerial LIDAR sensors. In *CLEO/QELS: 2010 Laser Science to Photonic Applications*, pages 1–2, 2010.

[104] T Korah, S Medasani, and Y Owechko. Strip Histogram Grid for efficient LIDAR segmentation from urban environments. In *CVPR 2011 WORKSHOPS*, pages 74–81, 2011.

[105] Dmitriy Korchev, Shinko Y Cheng, and Yuri Owechko. On Real-Time LIDAR Data Segmentation and Classification. In *IPCV'13 The 2013 International Conference on Image Processing, Computer Vision, and Pattern Recognition*, 2013.

[106] Quanshi Zhang, Xuan Song, Xiaowei Shao, Huijing Zhao, and Ryosuke Shibasaki. Start from minimum labeling: Learning of 3D object models and point labeling from a large and complex environment. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3082–3089. IEEE, may 2014.

[107] Mingfang Zhang, Daniel D. Morris, and Rui Fu. Ground Segmentation Based on Loopy Belief Propagation for Sparse 3D Point Clouds. In *2015 International Conference on 3D Vision*, pages 615–622. IEEE, oct 2015.

[108] Yungeun Choe, Seunguk Ahn, and Myung Jin Chung. Online urban object recognition in point clouds using consecutive point information for urban robotic missions. *Robotics and Autonomous Systems*, 62(8):1130–1152, aug 2014.

[109] Yungeun Choe, Seunguk Ahn, and Myung Jin Chung. Fast point cloud segmentation for an intelligent vehicle using sweeping 2D laser scanners. In *2012 9th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*, pages 38–43, nov 2012.

[110] Olympia Hadjiliadis and Ioannis Stamos. Sequential classification in point clouds of urban scenes. In *Proc. 3DPVT*. Citeseer, 2010.

[111] Zoubin Ghahramani. An Introduction to Hidden Markov Models and Bayesian Networks. *International Journal of Pattern Recognition and Artificial Intelligence*, 15(01):9–42, 2001.

[112] R Adams and L Bischof. Seeded region growing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(6):641–647, jun 1994.

[113] Satish K Reddy and Prabir K. Pal. Segmentation of point cloud from a 3D LIDAR using range difference between neighbouring beams. In *Proceedings of the 2015 Conference on Advances In Robotics*, AIR '15, pages 1–6, New York, NY, USA, 2015. ACM.

[114] Dimitris Zermas, Izzat Izzat, and Nikolaos Papanikolopoulos. Fast segmentation of 3D point clouds: A paradigm on LiDAR data for autonomous vehicle applications. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5067–5073. IEEE, may 2017.

[115] L He, Y Chao, and K Suzuki. A Run-Based Two-Scan Labeling Algorithm. *IEEE Transactions on Image Processing*, 17(5):749–756, 2008.

[116] J Strom, A Richardson, and E Olson. Graph-based segmentation for colored 3D laser point clouds. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2131–2136, 2010.

[117] Hitoshi Niigaki, Jun Shimamura, and Akira Kojima. Segmentation of 3D Lidar Points Using Extruded Surface of Cross Section. In *2015 International Conference on 3D Vision*, pages 109–117. IEEE, oct 2015.

[118] S Ashidate, S Murashima, and N Fujii. Development of a helicopter-mounted eye-safe laser radar system for distance measurement between power transmission lines and nearby trees. *IEEE Transactions on Power Delivery*, 17(2):644–648, 2002.

[119] L J Chaput. Understanding LiDAR data-How utilities can get maximum benefits from 3D modeling. In *Proceedings of the International LiDAR Mapping Forum (ILMF)*, pages 21–22, 2008.

[120] Thomas Melzer and Christian Briese. Extraction and Modeling of Power Lines from ALS Point Clouds. 2004.

[121] Paul V C Hough. Method and means for recognizing complex patterns, 1962.

[122] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2000.

[123] Simon Clode and Franz Rottensteiner. Classification of trees and powerlines from medium resolution airborne laserscanner data in urban environments. In *Proceedings of the APRS Workshop on Digital Image Computing (WDIC), Brisbane, Australia*, pages 191–196, 2005.

[124] Alberto Vale and João Gomes-Mota. Detection and Classification of Clearance Anomalies on Over-Head Power Lines. 2006.

[125] Alberto Vale and João Gomes-Mota. LIDAR data segmentation for track clearance anomaly detection on over-head power lines. 2007.

[126] Y Jwa, G Sohn, and H B Kim. Automatic 3D Powerline Reconstruction Using Airborne Lidar Data. *International Archives of Photogrammetry and Remote Sensing*, XXXVIII(2004):105–110, 2009.

[127] Jing Liang and Jixian Zhang. A New Power-line Extraction Method Based on Airborne LiDAR Point Cloud Data. *International Symposium on Image and Data Fusion (ISIDF)*, pages 2–5, 2011.

[128] Qing Xiang. 3D Reconstruction of 138 KV Power-lines from Airborne LiDAR Data. 2014.

[129] Liang Cheng, Lihua Tong, Yu Wang, and Manchun Li. Extraction of Urban Power Lines from Vehicle-Borne LiDAR Data. *Remote Sensing*, 6(4):3302–3320, 2014.

[130] Mohammad Awrangjeb, David Jonas, and Jun Zhou. An Automatic Technique for Power Line Pylon Detection from Point Cloud Data. *2017 International Conference on Digital Image Computing: Techniques and Applications (DICTA)*, pages 1–8, 2017.

[131] Bo Guo, Xianfeng Huang, Fan Zhang, and Gunho Sohn. Classification of airborne laser scanning data using JointBoost. *ISPRS Journal of Photogrammetry and Remote Sensing*, 100:71–83, 2015.

[132] Nick Mokey. "Solid-State Lidar: The Key to Cheap Self-Driving Cars — Digital Trends". [Online]. Available: `https://www.digitaltrends.com/cars/solid-state-lidar-for-self-driving-cars/`, 2018. [Accessed: 29-Aug-2018].

[133] LeddarTech. "Leddar Solid-State LiDAR Technology Fundamentals". [Online]. Available: `https://leddartech.com/technology-fundamentals/`. [Accessed: 29-Aug-2018].

[134] Timothy B. Lee. "Why spinning lidar sensors might be around for another decade — Ars Technica". [Online]. Available: `https://arstechnica.com/cars/2018/05/`

why-bulky-spinning-lidar-sensors-might-be-around-for-another-decade/, 2018. [Accessed: 29-Aug-2018].

[135] Renishaw. "Spinning LiDAR". [Online]. Available: http://www.renishaw.com/media/img/en/d5f2c0bc53c2419d850b1b979662efe5.jpg. [Accessed: 29-Aug-2018].

[136] Velodyne LiDAR. *VLP-16 User Manual*.

[137] Andy Mohd. "Does Infineon's MEMS lidar portend a quantum leap for ADAS?". [Online]. Available: https://medium.com/futuremobile2025/does-infineons-mems-lidar-portend-a-quantum-leap-for-adas-c14eb939545a, 2017. [Accessed: 29-Aug-2018].

[138] Taku Komura. Transformations.

[139] Gregory G. Slabaugh. Computing Euler angles from a rotation matrix.

[140] Mark Vallis. "Euler Angles". [Online]. Available: https://mveeprojects.files.wordpress.com/2016/10/pitchrollyawimage.gif, 2016. [Accessed: 30-Aug-2018].

[141] Eric W. Weisstein. Eigenvalue. From MathWorld–A Wolfram Web Resource. [Online]. Available: http://mathworld.wolfram.com/Eigenvalue.html. [Accessed: 10-Oct-2018].

[142] Erik Cheever. Eigenvalues and Eigenvectors. [Online]. Available: http://lpsa.swarthmore.edu/MtrxVibe/EigMat/MatrixEigen.html. [Accessed: 10-Oct-2018].

[143] Will Boney. Lines and planes in $\mathbb{R}^3$. In *Math 21a Multivariable Calculus*. Cambridge, MA, 2015.

[144] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer New York, 2009.

[145] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An Introduction to Statistical Learning*. Springer New York, 2013.

[146] Matt Brems. A One-Stop Shop for Principal Component Analysis – Towards Data Science. [Online]. Available: https://towardsdatascience.com/

a-one-stop-shop-for-principal-component-analysis-5582fb7e0a9c, 2017. [Accessed: 10-Oct-2018].

[147] Lazy Programmer. PCA. [Online]. Available: https://lazyprogrammer.me/wp-content/uploads/2015/11/PCA.jpg, 2015. [Accessed: 11-Oct-2018].

[148] W Gellert, S Gottwald, M Hellwich, H Kästner, and H Küstner, editors. *The {VNR} Concise Encyclopedia of Mathematics*. Springer Netherlands, 1990.

[149] Eric W. Weisstein. Line-line distance. From MathWorld–A Wolfram Web Resource. [Online]. Available: http://mathworld.wolfram.com/Line-LineDistance.html. [Accessed: 11-Oct-2018].

[150] Edward Parker. A Property Characterizing the Catenary. *Mathematics Magazine*, 83(1):63–64, 2010.

[151] Morgan Quigley, Ken Conley, Brian P Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y Ng. ROS: an open-source Robot Operating System. In *ICRA Workshop on Open Source Software*, 2009.

[152] W Curran, T Thornton, B Arvey, and W D Smart. Evaluating impact in the ROS ecosystem. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6213–6219, 2015.

[153] Clearpath Robotics. "ROS High-Level Architecture using a Local Access Network". [Online]. Available: https://robohub.org/wp-content/uploads/2014/01/ros101-3.png, 2014. [Accessed: 31-Aug-2018].

[154] Christian Damm. Object detection in 3D point clouds. Master's thesis, Institut für Informatik der Freien Universität Berlin, 2016.

[155] DNS. ODROID-XU4. [Online]. Available: https://s.technopoint.ru/thumb/st1/fit/wm/800/650/d299073506ab59f2a7dc1cdf6c3a1316/266198485396814a37d5b796d6d35c9ed6a8ec1c4ab69ccc1199485cac222660.jpg. [Accessed: 04-Sep-2018].

[156] Freepik. Flat Drone. [Online]. Available: https://www.freepik.com/free-vector/basic-variety-of-flat-drones_1348538.htm, 2017. [Accessed: 04-Sep-2018].

[157] Tsukasa Sugiura. VLP-16 Laser Beams. [Online]. Available: http://unanancyowen.azurewebsites.net/wp-content/uploads/vlp16_laser.png, 2017. [Accessed: 04-Sep-2018].

[158] Tobias Nyström Johansson and Oscar Wellenstam. Lidar clustering and shape extraction for automotive applications. Master's thesis, Chalmers University of Technology, Department of Electrical Engineering, Gothenburg, Sweden, 2017.

[159] Geovany Araujo Borges and Marie-José Aldon. Line extraction in 2d range images for mobile robotics. *Journal of Intelligent and Robotic Systems*, 40(3):267–297, jul 2004.

[160] Eric W. Weisstein. Law of sines. From MathWorld–A Wolfram Web Resource. [Online]. Available: http://mathworld.wolfram.com/LawofSines.html. [Accessed: 15-Oct-2018].

[161] NovAtel. "Real-Time Kinematic (RTK) — An Introduction to GNSS — NovAtel". [Online]. Available: https://www.novatel.com/an-introduction-to-gnss/chapter-5-resolving-errors/real-time-kinematic-rtk/. [Accessed: 31-Aug-2018].

[162] Velodyne LiDAR. VLP-16. [Online]. Available: https://velodynelidar.com/vlp-16.html. [Accessed: 15-Oct-2018].

[163] Pires A., Dias A., Santos T., Oliveira A., Azevedo F., Ferreira A., Almeida J., Ferreira A., Chaminé H.I., and Silva E. Unmanned geo-technology systems: aerial imagery, survey and mapping of georesources and maritime structures. In *Proceedings of 3rd International Conference on Information Technology in Geo-Engineering*. Geomechanics and Geoengineering Journal, Springer Series, 2019. Submitted.