TÉCNICO LISBOA

TU Delft

LASER-BASED CONTROL OF ROTARY-WING UAVS

Alexandre Luís Cordeiro Gomes

Thesis to obtain the Master of Science Degree in

Aerospace Engineering

Supervisors: Dr. Rita Maria Mendes de Almeida Correia da Cunha
Dr. Bruno João Nogueira Guerreiro

Examination Committee

Chairperson: Dr. João Manuel Lage de Miranda Lemos
Supervisor: Dr. Bruno João Nogueira Guerreiro
Member of the Committee: Dr. Alexandra Bento Moutinho

December 2015

To my father.

Acknowledgments

I would like to begin by expressing my deepest gratitude to my supervisors, Drs. Rita Cunha and Bruno Guerreiro. In our early contacts, they always presented themselves available to discuss new ideas for a thesis, not limiting themselves to the topics that were being planned for the current year and wanting to hear what I might find interesting to work with. Both of them knew not only how to keep me motivated but also when to give me the short answer and when to make me work harder and push myself to strive for the solution on my own. Even with many other responsibilities, I can never say I felt unsupervised or at drift during that time. Dr. Bruno Guerreiro spent one month away across the world, but that fact did not prevent him from keeping up with my progress and providing feedback on the work I was producing. Dr. Rita Cunha was always one step ahead of what I was developing, conceiving alternatives to accomplish the goals we had set to achieve at the end. A fulfilling thesis needs not only a topic with potential that truly interests the student, but also a solid coordination from qualified and experienced professionals, both of which I believe I got.

Because this thesis had a very important experimental component, I could not go by without mentioning those who had shared their hands-on expertise and devoted their time to help me take that leap into experimenting and gathering data in the real world. One of those people was Bruno Gomes, someone with an on point and clear answer to every question I posed, always willing to give friendly advices that would reveal themselves essential along the road. He repeatedly put his own work on hold, to assist me when I truly needed, and gave all the necessary explanations for me to delve into what had been his previous work on the quadcopter that I had available for tests. This quadcopter had been used within this research team before, to which Dr. Bruno Guerreiro also belongs. Same as Bruno Gomes, he gave me invaluable insights into the code that he had designed and that I ended up building upon during my thesis. Last but not least, José Tojeira was also a key individual in my work, helping whenever an hardware upgrade was necessary on the quadcopter, for testing an hypothesis or even piloting the vehicle in more realistic experimental sessions.

Even indirectly, the people that surrounds us during a project such as this have an impact on the end product. This is why I want to thank my friends at the laboratory, always present during those months, allowing me to take a break when I really needed it and being available to discuss the small roadblocks we all hit sometimes, which often only require someone there to listen for our own mind to solve them.

Lastly, I need to thank my family for all the support they have given me along this journey, because without them I would not be here right now. Specially to my father, who has made more sacrifices than I could ever imagine, to give me all the opportunities I have had, and has always revealed interest in my work no matter what.

Resumo

Veículos Aéreos Não Tripulados (VANTs) apresentam uma elevada taxa de desenvolvimento tecnológico actualmente. Estes veículos podem ser utilizados para executar procedimentos de inspecção perigosos e caros em estruturas de difícil acesso em vez de operadores humanos, mas ainda necessitam de uma monitorização próxima. Esta tese aborda o problema de utilizar exclusivamente sensores a bordo de VANTs para criar ferramentas de determinação de atitude e estratégias de seguimento de trajectória. Primeiramente, este trabalho discute a percepção do mundo exterior pelo veículo e a formulação de uma descrição matemática contendo informações sobre a sua posição e atitude relativos à estrutura. Para este efeito, uma geometria é definida e o melhor ajuste aos dados fornecidos por um sensor LiDAR é seleccionado, após um processo robusto de filtragem de pontos. Com esta informação, são propostos vários métodos para obter a atitude. Estes incluem um estimador de guinada rápido e completo, baseado em continuidade, e uma solução em forma fechada para o problema de Wahba e um filtro não linear para a estimação de atitude completa, ambos no grupo das matrizes de rotação. Foi dedicado um esforço significativo à análise de todo o procedimento através de simulação, da criação dos dados laser à aplicação dos métodos, para fins de validação. Resultados simulados e experimentais são fornecidos para a avaliação de desempenho dos algoritmos de percepção. Com base nesses resultados, uma estratégia de controlo não linear é projectada com o objectivo de proporcionar um controlo de trajectória preciso relativo à estrutura, com estabilidade assimptótica garantida.

Palavras-chave: VANTs, sensores LiDAR, ajuste geométrico, determinação de atitude, estabilidade de Lyapunov, controlo de trajectória.

Abstract

Unmanned Aerial Vehicles (UAVs) present a high technological development rate nowadays. These vehicles can be used to perform dangerous and costly inspection procedures in structures with difficult access instead of human operators, but they still need a close monitoring. This thesis addresses the problem of using exclusively sensors on board UAVs to derive attitude determination tools and trajectory tracking strategies. Firstly, this work discusses the perception of the outside world by the vehicle and the formulation of a mathematical description containing information regarding its position and attitude relative to the structure. For this purpose, a geometry is set and the best fit to the data provided by a LiDAR sensor is selected, after a robust outlier filtering process. With this information, several methods for obtaining the attitude are proposed. These include a fast and comprehensive yaw estimator, based on continuity, and a closed-form solution for the Wahba's problem and a nonlinear filter for a full attitude estimation, both on the group of rotation matrices. A significant effort was devoted to the analysis of the entire procedure through simulation, from the creation of the LiDAR data to the application of the methods, for validation purposes. Both simulated and experimental results are provided for the performance evaluation of the perception algorithms. Building on these results, a nonlinear control strategy is designed with the objective of providing an accurate trajectory tracking control relative to the structure, with guaranteed asymptotic stability.

Keywords: UAVs, LiDAR sensor, geometry fitting, attitude determination, Lyapunov stability, trajectory tracking.

Contents

List of Tables

List of Figures

Nomenclature

Greek symbols

ϕ	Roll angle.
θ	Pitch angle.
ψ	Yaw angle.
$\boldsymbol{\lambda}$	Attitude vector.
$\dot{\boldsymbol{\lambda}}$	Derivative of the attitude vector.
ρ	Perpendicular range from a line to the origin.
γ	Angle to the positive X axis.
$\delta_{(.)}$	Interval range related to its subscript.
$\epsilon_{(.)}$	Contiguous spacing related to its subscript.
ε	Error measure between the edges in $\{B\}$ and $\{I\}$.
$\boldsymbol{\varepsilon}$	Vector of the error measures between the edges in $\{B\}$ and $\{I\}$.
$\mathbf{\Pi}_{(.)}$	Projection in the plane normal to the vector in its subscript.
$\boldsymbol{\omega}$	Angular velocity vector.
$\hat{\boldsymbol{\omega}}$	Estimation of the angular velocity vector.
$\boldsymbol{\delta\omega}$	Angular velocity bias vector.
$\boldsymbol{\nu}$	Generic vector.
$\boldsymbol{\eta}$	Generic unitary vector.
ξ	Generic angle.
$\boldsymbol{\zeta}$	Generic symmetric matrix.
$\lambda(.)$	Eigenvalue operator.
α	Heading angle.

$\boldsymbol{\alpha}$ Laser beam's heading angles vector.

$\boldsymbol{\Theta}$ Generic positive definite weighting matrix.

Roman symbols

$\{.\}$ Reference frame.

X Longitudinal axis.

Y Lateral axis.

Z Vertical axis.

XY Horizontal plane.

$\boldsymbol{M}$ Measurements matrix in the XY plane.

$\mathbb{R}$ Set of real numbers.

$\mathbb{R}^{(.)}$ Set of $(.)$-dimensional vectors with real entries.

$\boldsymbol{p}$ Position vector.

$\boldsymbol{n}$ Normal vector of a line.

c Perpendicular offset from a line to the origin.

e Perpendicular offset from a point to a line.

$\boldsymbol{e}$ Vector of the perpendicular offset from points to a line.

L Dimension of a pier's edge.

$n_{(.)}$ Quantity related to its subscript.

$f_{(.)}$ Threshold factor related to its subscript.

$m_{(.)}$ Slope of the linear equation related to its subscript.

$b_{(.)}$ Y-intercept of the linear equation related to its subscript.

$\mathbf{0}$ Vector of zeros with dimensions obvious from context.

$\mathbf{0}_{N\times M}$ Vector of zeros with dimensions $N \times M$.

$\mathbf{1}_{N\times M}$ Vector of ones with dimensions $N \times M$.

$\boldsymbol{I}$ Identity matrix with dimensions obvious from context.

$\boldsymbol{d}$ Euclidean distances vector.

i, j Index of a vector or matrix dimension obvious from context.

$\boldsymbol{i}$ Indexes vector of a matrix dimension obvious from context.

$\boldsymbol{w}$ Weights vector.

$\boldsymbol{Q}$ Edges matrix.

$\boldsymbol{l}$ Projection in the XY plane of an edge in $\{I\}$.

h Projection in the Z axis of an edge in $\{I\}$.

$\boldsymbol{e}_{(.)}$ Versor of the axis numbered in its subscript.

$s_{(.)}$ Sine of the angle in its subscript.

$c_{(.)}$ Cosine of the angle in its subscript.

$r_{(.)(.)}$ Rotation matrix's element corresponding to its subscript.

$\boldsymbol{r}_{(.)}$ Rotation matrix's column corresponding to its subscript.

$\boldsymbol{R}$ Rotation matrix.

$\dot{\boldsymbol{R}}$ Derivative of the rotation matrix.

$\hat{\boldsymbol{R}}$ Estimation of the rotation matrix.

$\tilde{\boldsymbol{R}}$ Error in the estimation of the rotation matrix.

$L(.)$ Loss function.

$\boldsymbol{a}$ Acceleration vector.

$\boldsymbol{O}$ Observations matrix.

$\boldsymbol{H}$ Generic matrix.

$\boldsymbol{A}$ Kinematics matrix.

$\mathcal{SO}(.)$ Special orthogonal group with . $\times$. proper rotation matrices.

$K_{(.)}$ Gain related to its subscript.

t Period of time.

dt Infinitesimal period of time.

$h_{(.)}$ Sampling period of the sensor in its subscript.

r Radial distance to the origin.

$\boldsymbol{u}_{(.)}$ Direction vector related to its subscript.

$\boldsymbol{f}$ Vector alongside a pier's face.

$T_{(.)}$ Time constant related to its subscript.

m Mass.

g Gravitational acceleration.

T Thrust.

$\boldsymbol{v}$ Linear velocity vector.

$\boldsymbol{j}$ Jerk vector.

$\boldsymbol{x}$ State vector.

$\boldsymbol{u}$ Input vector.

$\boldsymbol{B}$ Input matrix.

$\boldsymbol{P}$ Generic symmetric weighting matrix.

$|.|$ Absolute value operator for scalars.

$||.||$ 2-norm operator for vectors.

$. \times .$ Cross or external product operator.

$S(.)$ Skew-symmetric matrix function.

$S^{-1}(.)$ Skew-symmetric matrix inverse function.

$\mathrm{tr}(.)$ Trace operator.

$\mathrm{diag}(.)$ Block diagonal matrix operator.

$\mathrm{length}(.)$ Largest dimension measurement function.

$\mathrm{sort}(.)$ Increasing sorting function.

$\mathrm{atan2}(.)$ Four-quadrant inverse tangent function.

Subscripts

x Longitudinal position.

y Lateral position.

z Vertical position.

min Minimum value.

max Maximum value.

new Current value.

$corner$ Rectangle corner.

$edge$ Rectangle edge.

$point$ Data point.

$cluster$ Data point cluster.

$outliers$ Outlier data points.

set Data set.

$line$ Straight line.

$\boldsymbol{R}_{abc}$ Rotation matrix about the A, B and C axes in that order.

obs Observations.

$pier$ Structure.

$traj$ Body's trajectory.

$face$ Pier's face.

c Center.

$beam$ Laser beam.

d Desired.

Superscripts

$(.)^2$ Scalar or matrix square operator.

$(.)^{-1}$ Matrix inverse operator.

$(.)^T$ Vector or matrix transpose operator.

${}^A(.)$ Represented in $\{A\}$.

${}^C_B\boldsymbol{R}$ Rotation matrix from $\{B\}$ to $\{C\}$.

Glossary

2D	two Dimensional
3D	three Dimensional
DOF	Degree Of Freedom
GPS	Global Positioning System
GUI	Guest User Interface
ICAO	International Civil Aviation Organization
IMU	Inertial Measurement Unit
ISR	Institute for Systems and Robotics
IST	Instituto Superior Técnico
LTV	Linear Time-Varying
LiDAR	Light Detection And Ranging
MAD	Median Absolute Deviation
NED	North East Down
POV	Point Of View
RADAR	Radio Detection And Ranging
ROS	Robot Operating System
SVD	Singular Value Decomposition
UAS	Unmanned Aircraft System
UAV	Unmanned Aerial Vehicle

Chapter 1

Introduction

An Unmanned Aerial Vehicle (UAV), also known as a drone, is an aircraft without a human pilot aboard. There are two types of UAVs, the autonomous and the remotely piloted aircrafts. The former is characterized by an autonomous flight control through on-board computers without human interference, currently not suited for regulation due to legal and liability issues. On the other hand, the latter depends on a human pilot, either on ground or inside another vehicle, and it is subject to civil regulation under the International Civil Aviation Organization (ICAO) and the relevant national aviation authority. [1]

Besides the aircraft itself, there are other important elements related with the operation of UAVs that together form an Unmanned Aircraft System (UAS). Those include the associated support equipment, control station, data links, telemetry, communications, and navigation equipment. A UAS is required in order to have safe and reliable operations with these vehicles. [2]

UAVs were initially developed with military purposes, namely for situations where manned flight was considered too risky or difficult [3, 4]. During these high precision tasks, they required a set of sensors that would allow them to maneuver in the world as accurately as possible. As with other technology advancements, the world soon realized that these small vehicles, equipped with a great variety of sensors, could be used in tasks other than warfare.

Meanwhile, numerous civil aviation uses have been developed, including surveying of crops for agriculture, footage capturing for film making, search and rescue operations, power line and pipeline inspections, wildlife counting, and many others. These vehicles have been becoming increasingly meaningful in today's society, by making several previously dangerous and costly tasks much more effective.

Regarding the sensors on board, they need to be able to create a digital awareness of the vehicle's movement in the world. The world as we know it is a three Dimensional (3D) space, where any movement of a rigid body can be decomposed into a combination of translations and rotations along three different axes. In an aircraft, the origin is usually at the vehicle's center of mass and a rotation about its longitudinal, lateral, and vertical axes is called roll, pitch, and yaw respectively. Usually, UAVs contain an Inertial Measurement Unit (IMU) and a Global Positioning System (GPS), which are used to estimate the information regarding the acceleration, velocity and position of the vehicle.

After covering the current status of this industry, Section 1.1 goes through some aspects concerning the improvement of the systems that were described and are currently in place. Having a clear idea about the relevance of their components and how they can be combined to generate enhanced alternatives, an application and the motivations behind that choice are discussed in Section 1.2. Lastly, Sections 1.3 to 1.5 establish the main goals for the work developed from hereon and describe the contributions and general structure of the thesis respectively.

1.1 Alternative Navigation Solutions

A Light Detection And Ranging (LiDAR) system is a sensor that measures distance by illuminating a target with a laser and analyzing the reflected light. It can be seen as a combination of laser-focused imaging with the ability of the Radio Detection And Ranging (RADAR) system to calculate a distance by measuring the time-of-flight of a signal [5]. This sensor can target a wide range of materials, including non-metallic objects, rocks, rain, chemical compounds, aerosols, clouds, and even single molecules [6].

Redundancy is the duplication of critical components or functions of a system, with the intention of increasing reliability of the system, usually in the form of a backup or fail-safe [7]. During navigation tasks, there is always the possibility of sensor failure due to unforeseen events. Moreover, given that the accuracy of a sensor is never 100%, having different mechanisms to obtain a certain quantity may prove itself a valuable feature in a control system, as it allows for data fusion in situations suspected to cause ambiguous readings. Having an on-board GPS is one of the main reasons these vehicles are able to fly autonomously, a feature that can become compromised in the vicinity of large infrastructures or under bridges. This happens since the GPS signal can be easily occluded by these structures. The LiDAR can be an interesting alternative, from the redundancy point of view, that may be used to overcome signal propagation issues and even calculate relevant data in a more accurate way.

UAVs are rapidly evolving to become highly capable sensing platforms, able to navigate and track trajectories with great accuracy. While the motion control of aerial vehicles in free flight is reaching its maturity, new challenges that involve interaction with the environment are being embraced. Using local sensors, such as IMUs, some quantities required for control tasks can be obtained depending entirely on the vehicle. As controlling the full motion of an aerial vehicle is not usually viable with only an IMU, this thesis aims to take that interaction with the environment one step further and obtain additional measurements, from what the vehicle is perceiving of the world.

1.2 Innovative Structure Inspections

The technological evolution has led to an increase in the demand for more and larger wind turbines, cellphone towers, and power lines, to name a few. All these large buildings and facilities are critical infrastructures that require maintenance through structural inspections and health monitoring, which in general requires complex and expensive routine inspections and monitoring procedures.

Those processes can become inefficient in situations where the access is difficult, time-consuming, and often dangerous. Some of these structures are hundreds of feet tall, making it difficult to get human operators at the scene to conduct an inspection. Large structures also present a significant challenge, since the process is extremely labor-intensive, due to the amount of area to cover. Structurally unsound towers, buildings, or bridges can also be extremely hazardous. Information management may also be a challenge since the inspection data may not have the proper detail, i.e. a wrong angle used when an image was taken, and they maybe scattered into hundreds of files, which is difficult to manage. [8]

Accurate health monitoring and diagnosis of these infrastructures will increase the efficiency of maintenance and repair plans, with inherent benefits in terms of cost reduction and damage minimization in case of disaster. Robotic surveying tools can be used to address this issue, namely UAVs equipped with state-of-the-art sensors. Small vehicles such as these constitute a tailor-made solution, since they should be able to perform high accuracy three-dimensional surveys of structures, with the objective of producing accurate data sets, in real time, with the required spatial and temporal resolutions and thereby providing quantitative information, vital for a well-founded diagnosis. Therefore, UAVs can be used to perform structural inspections by looking for cracks, structural defects, and other visual information from a bird's eye, taking the skill out of flying the vehicle and allowing the inspector to focus on interpreting the data.

The UAV senses its environment and maneuvers around the structure, while the operator determines what to inspect and uses his or her knowledge to determine the results of the inspection. It can gather data objectively from the same locations and precisely quantify it, meaning there will be repeatable, reliable information. The inspection results are still interpreted by a qualified technician, but the time invested in rigging and climbing the structure is eliminated from the routine inspection. The inspection can be done faster, which means that more structures can be processed, and overall safety and reliability can be increased. [8]

As critical structures increase in number and size, there must be a corresponding increase in structural assessment. Timely and accurate information can be costly and at times dangerous, due to the complexity of accessing large structures. To increase safety and efficiency in structure inspections, by using UAVs, is an important step forward. Moreover, the use of a LiDAR may present an interesting enhancement to structural inspections, since it provides a way to maintain a lock on the target. This relative positioning method has the potential to be more effective and subject to less drift and error accumulation over time than traditional absolute positioning systems, as it depends directly on LiDAR scans acquired at a high frequency.

The basis for this thesis can be found in [9], where a LiDAR sensor was used to provide a relative positioning solution to a structure and then to build the appropriated controllers. The idea of the current proposal is to extend that relative description to the vehicle's attitude, to allow a structure dependent trajectory. Furthermore, the control techniques investigated in the aforementioned work rely on the local linearization of a nonlinear system, used afterwards within a gain scheduling strategy. The present work intends to compute a nonlinear controller, for the original system, that ensures asymptotic stability with a large region of attraction.

1.3 Objectives

There are three main goals to this work, the first of which is to develop laser-based methods for landmark detection, creating the necessary algorithms and testing several alternatives, in order to obtain a framework for perceiving the environment. The following objective is to estimate the attitude of the vehicle, striving for speed of execution, ease of implementation and stability guarantees. The final aim is to design the motion control of the vehicle in dynamic environments, such as the inspection of buildings and industrial facilities. The idea is to build upon the previous achievements and provide a stability proof, in order to enable the execution of maneuvers that involve close interaction with the environment and require a reactive and compliant behavior.

1.4 Contributions

Several contributions can be derived from the objectives set in the previous section. Regarding the landmark detection, the framework for processing data from a laser sensor is the first, which besides being invaluable for this application it can be simply adjusted to work with different goals. It should be noted that these tools will allow for a 3D attitude to be retrieved from a sensor measuring in a 2D space. When it comes to the attitude estimation, the most relevant contribution is a nonlinear filter, that computes the rotation matrix describing the motion of a vehicle based on the fusion of measurements from several local sensors, including a LiDAR and separate components from an IMU. Finally, the motion control design yields a trajectory tracking controller solely based on local sensory information, therefore providing a relative positioning solution.

1.5 Outline

Concerning the organization of the thesis, firstly Chapter 2 discusses the detection part of the approach, more specifically how the vehicle perceives the world and translates it to an interpretation intuitive and accurate enough to be manipulated. For this purpose, the LiDAR sensor used for the development of this work was investigated, to acquire a sense about its features and draw some conclusions regarding its suitability for this application. After establishing the characteristics of the source information, the pier detection consists of building a strategy based on the Split & Merge algorithm for the extraction of landmark features. This procedure is paired with a robust outlier identification and removal algorithm that intends to improve the performance of the detection stage. Then, a least square estimation is compared with a reduced space Hough transform algorithm in the process of fitting the data to build a description of the landmarks.

The following step is to actually process the data and derive several methods to obtain an attitude description consistent with the movement of the vehicle in Chapter 3, so that it can be used for establishing control strategies later on. Among them, there is a history-based yaw tracker, a solution to the Wahba's problem on the group of rotation matrices, and a nonlinear observer with filtering capabilities and a proof

of stability. Afterwards, Chapter 4 goes through an artificial simulated environment created especially for testing these methods and visualize the behavior of the vehicle in a wide range of scenarios that may be encountered. The following logical step towards validation is a proper evaluation of the previously developed work, in Chapter 5, both in simulation and the real world.

Next, Chapter 6 discusses the control strategies developed and their implementation in both simulation and the real world. The focus is on obtaining closed-loop solutions for the problem of trajectory stabilization, with respect to a fixed target, by determining an error variable that adequately represents the vehicle's position and orientation error relative to it, in the inertial space. The concepts from Lyapunov control theory are applied, to ensure the asymptotic stability of the derived control system. In the end, the conclusions obtained from the results presented along the thesis are drawn in Chapter 7.

Chapter 2

Environment Perception

In order to relate one object and its features with another in the world, there must must a frame of reference in which both can be represented. These frames can be chosen arbitrarily, however Section 2.1 describes those deemed necessary to understand and manipulate the elements at hand in this thesis, such as the vehicle and the objects around it. After having the tools to make those connections, the sensor with which the world is going to be observed needs to be analyzed and its accuracy established, so that the precision limits of the developed methods can be known and the expectations towards its performance may be appropriately scaled. Section 2.2 goes through the intuition behind the quality of LiDAR measurements in certain situations and how the existing work on that field corroborates or not those notions, since determining to what extent can the LiDAR data be relied upon is a very important step.

The data points coming from the LiDAR measurements need to be converted into a description that can be used to compute an attitude. Therefore, some assumptions have to be made regarding the geometry of the structures in Section 2.3. With every sensor, there are measurement errors that cause the data set to have outliers, which will then affect the processing stage and cause a deviation in the results. Section 2.4 approaches this highly unpredictable and yet truly important subject and describes the search for the best solution possible. Knowing the reliability of the sensor, the geometry of the chosen structure, and which data points are valid, the final step in reconstructing the structure is to estimate its identifying marks. This topic and two options are discussed in Section 2.5.

At last, all the necessary information before the data processing is available and must be unified. Section 2.6 goes through this final hurdle in the task of providing a full description of what the sensor is perceiving at a given moment, including the characteristic marks of the structure's geometry and a piece of information that will be useful for control purposes, the location of its center.

2.1 Reference Frames

There are several important reference frames during the LiDAR data processing, the first being the Inertial Frame $\{I\}$, which for simplicity is considered to be the local tangent plane with the North East

Down (NED) convention. There is also the Body Frame $\{B\}$, with the origin at the vehicle's center of mass, the X axis pointing forward along the longitudinal axis of the vehicle, the Z axis pointing downward along its vertical, and the Y axis set to make this frame orthonormal. These two reference frames are shown in Figure 2.1. Finally, there is the intermediate Horizontal Frame $\{H\}$, which can be interpreted as a projection of $\{B\}$ on the XY plane of $\{I\}$.

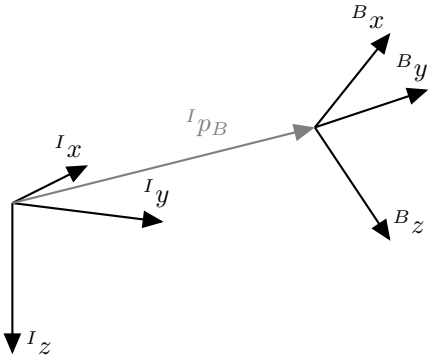


Figure 2.1: Representation of $\{I\}$ and $\{B\}$.

The two main sensors used in this work, a LiDAR and an IMU, are mounted on board the vehicle with an offset in the Z axis of roughly -10 cm and -2 cm respectively. For simplicity purposes, these offsets and their consequent translation resulting from rotating the vehicle around its center of mass are considered to be negligible. This means the LiDAR's reference frame will be coincident with $\{B\}$. However, besides sharing the origin, the axes of the IMU can have a different orientation when compared with $\{B\}$, so a Sensor Frame $\{S\}$ was defined to establish their independence.

2.2 LiDAR Reliability

The sensor's precision, and consequently the quality of the measurements, is a very sensitive matter, since it is one of the most relevant limiting factors in what concerns the accuracy of the data processing and the following results.

In [10], the accuracy of the data points from a two Dimensional (2D) Hokuyo UTM-30LX LiDAR, a sensor quite popular in small vehicles, was analysed while detecting moving cylindrical targets. This sensor, whose scanning plane is illustrated in Figure 2.2, has an angle opening of $270°$, with a blind cone of $90°$ around its 6 o'clock, an angular resolution of $0.25°$, a scanning frequency of 40 Hz, and a range of 30 m, weighting roughly 200 g. The described approach was based on the detection of markers belonging to a circular section from the data points provided by the LiDAR, after undergoing an outlier avoidance method and a least-squares circular fitting.

The experiments carried out in that paper take into account the proximity to the target and its surface's characteristics. The former is rather self explanatory, while the latter is related with the curvature of the object where the beams hit. Intuitively, the closer the target, the better the results. However, one of the findings in the paper is that the error reaches its maximum value for distances between 0 and 2000 mm from the target. Beyond that distance, the error is relatively small and approximately constant.

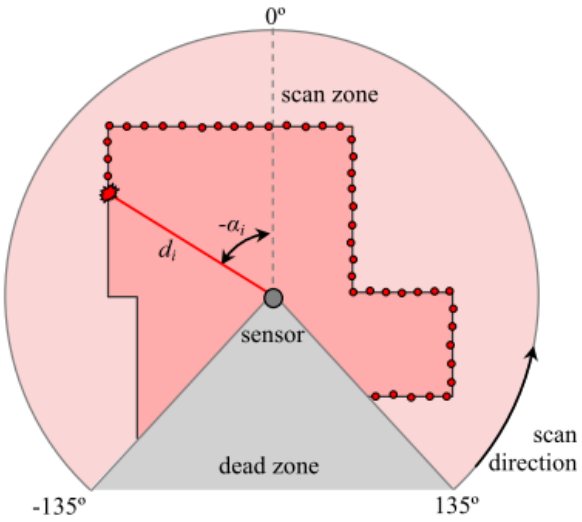


Figure 2.2: LiDAR scanning. [10]

Another interesting observation is that the LiDAR has its highest error in the $90°$ cone around its 12 o'clock. This error is maximum at the center and decreases in an approximately linear trend until the end of the cone, after which that value is significantly lower and constant. As to ensure that the methods developed later in this work are reliable enough, even in the most adverse conditions, the necessary tests were performed preferably within the 2000 mm range and inside the $90°$ frontal cone, where the highest errors are more prone to arise. Regarding the curvature of the target, a round object with a small diameter has a lower probability to be properly detected by the LiDAR, since the abrupt curvature in a smaller space can be incompatible with the LiDAR's resolution. The conversion from an analog to a digital signal implies the association between a continuous and a discrete domain, by rounding a given continuous value to its closest discrete approximation. When the gap between two consecutive steps of the discrete domain is large, i.e. the sampling frequency is not high enough, sudden jumps in the data can be the result of quantization errors, which are one possible source of measurement noise in these sensors. The paper verifies that a wider diameter leads, in fact, to better results and therefore smaller errors. Additionally, the paper proposes a cooling method using an aluminum plate to prevent the LiDAR from overheating and causing the measurements to deteriorate with time. That suggestion was incorporated into the present work after the initial testing and the results in Chapter 5 were obtained with it implemented.

A second LiDAR was briefly used during the testing stage, in order to investigate the effect of a different sensor in the structure detection. The chosen equipment was a Sick LMS100-10000, which has the same angle opening and angular resolution but a lower scanning frequency of 25 Hz and range of 20 m, weighting around 5.5 times more than the previous sensor. Comparing the output of both LiDARs, naturally there is not a significant difference in the data points gathered given the similarity in the specifications, which verifies their functioning and validates the data. Besides that, the lower sampling frequency definitely caused a deterioration in the results, specially with the methods that rely on the history behind the current time step, because this way there is a higher interval between consecutive measurements, in which more changes can occur.

2.3 Detection Strategies

Detecting a structure and obtaining the pertinent information, needed to determine the vehicle's attitude, greatly depends on the knowledge of its geometry. Considering the most common types of infrastructures, first a cylindrical and then a cuboid geometry were analyzed, where the former was discovered to have some limitations. These two different options on how to process the collected data are approached in Sections 2.3.1 and 2.3.2 respectively, with its pros and cons duly weighted.

It should be noted that a raw LiDAR scan contains all the data points within the angle opening of the sensor, however the research team at the Institute for Systems and Robotics (ISR), in Instituto Superior Técnico (IST), already has routines in place to perform the basic clustering of the set and yield the closest cluster at each step, which is assumed to be the starting point of the following work.

2.3.1 Cylindrical Structures

Taking into account that a vast extension of structures, throughout the world, has piers designed with a constant circular section along its height, the cylinder comes as a natural choice concerning the geometry of possible landmarks to inspect. Its geometrical properties are also appealing from a modeling perspective.

2.3.1.1 Geometrical Recognition

While analyzing a cylindrical structure, the LiDAR gathers a set of measurements belonging to either a circle or an ellipse, depending on the current attitude. In level flight, the intersection of the LiDAR scanning plane with the pier is a circle, whereas with a pitch or a roll angle that shape will shift to an ellipse.

With the data points from the LiDAR scan, the parameters that describe the best fitting ellipse can be acquired through its general quadratic equation. In [11], a numerically stable non-iterative algorithm to fit ellipses to data points was developed, inspired and built upon the work presented in [12]. This approach avoids the numerical instability of its predecessor and it is based on a least squares minimization, having been validated with data containing noise.

The idea behind attitude determination is that a specific movement, in roll or pitch, has an independent impact on the length of the ellipse's major or minor semi-axis. According to the magnitude of that movement, these axes can switch, but they are always perpendicular to each other, with one of them being parallel to the vector uniting the center of the vehicle's reference frame and the center of the pier. It can be seen that a movement in roll will have a direct effect on the semi-axis along the perpendicular to that vector and one in pitch on the semi-axis along the parallel.

2.3.1.2 LiDAR Experiment #1

The first experiment was conducted using a cylindrical PVC tube, with a 0.235 m diameter, placed about 0.8 m from a quadcopter. It consisted of maintaining the vehicle in a fixed position, regarding

the tube, and manually performing three different movements, with the rotors off, to evaluate the data resulting from them. The first two were decoupled movements in roll and pitch, along the positive direction of the corresponding axes, whereas the last was the combination of both.

In the beginning of the experiment, a sample corresponding to a leveled state was retrieved and shown in Figure 2.3. After an initial observation, it can be concluded that the LiDAR detects slightly less than $180°$ of the pier most of the times. Additionally, the LiDAR's resolution and inherent noise, together with the fact that the data points at the boundaries of the set suffer from reflections and have an unpredictable erroneous location, cause the fit of the ellipse through the least-squares algorithm not to be consistent with reality, given the absence of maneuvers. On the other hand, the circle obtained by forcing a circle fit with the available data points matches very closely the actual section of the pier, corroborating the results from the paper mentioned in Section 2.2. There is only one ellipse that best fits a given data set even with only almost $180°$, so these however small variations in the data set have a large impact on the fitted ellipse, consequently greatly affecting the lengths of the semi-axes. Besides those lengths, also the orientation of the ellipse regarding the longitudinal axis becomes tilted, making this semi-axis not to be aligned with the vector linking the quadcopter and the pier, which was the basis for the attitude determination.

The sample in Figure 2.3 was chosen to illustrate the most common cases, although there were situations where the fit was much more unrealistic. The scarcity of realistic results indicates that this approach is neither accurate nor stable enough.

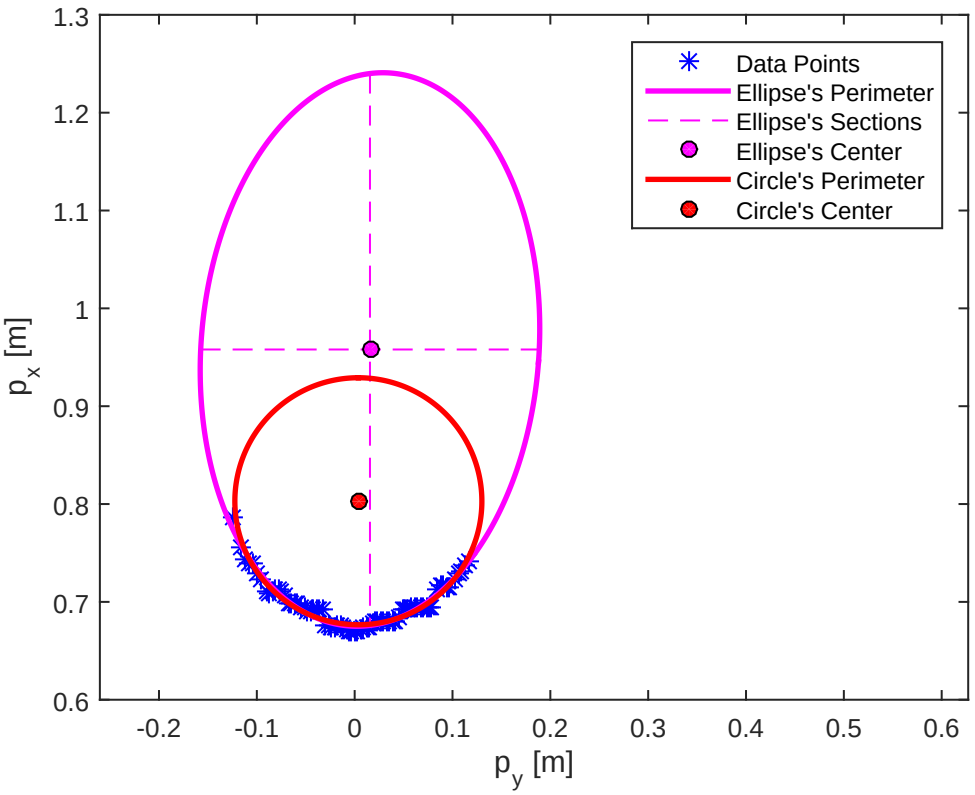


Figure 2.3: Cylindrical pier detection during the first experiment.

2.3.2 Cuboid Structures

Considering piers with a rectangular section is a good alternative for LiDAR-based perception and control, given the wide range of landmarks in which the curvature does not play a role in the fitted shape, as with cylindrical structures.

2.3.2.1 Geometrical Recognition

Facing these structures, the LiDAR will hit one or two faces of the pier, depending on its relative pose, and the intersection of this sensor's plane with these faces will result in two straight lines, which from hereon will be called edges. Similarly to the alternative discussed in Section 2.3.1, when the vehicle performs a movement, in roll or pitch, there is an impact on both the length of the edges and the angle between them. Once again, that effect can be interpreted as the contribution of these two components, but this time not with such a clear separation as with an elliptical section. The first stage is related with identifying how many edges the vehicle is encountering at each moment. For that purpose, a strategy based on the Split & Merge algorithm, which can be found in [13] and was originally proposed in [14], was developed and then presented in Algorithm 1.

Algorithm 1 Split Edges

procedure SPLITEDGES($\boldsymbol{M}, e_{corner}, n_{edge}$)
  $\gamma \leftarrow \tan^{-1}\left(\frac{\boldsymbol{M}(end,1)-\boldsymbol{M}(1,1)}{\boldsymbol{M}(1,2)-\boldsymbol{M}(end,2)}\right)$
  $\boldsymbol{n} \leftarrow [\cos(\gamma), \sin(\gamma)]^T$
  $c \leftarrow -\boldsymbol{M}(1,:) \cdot \boldsymbol{n}$
  $\boldsymbol{e} \leftarrow \boldsymbol{M} \cdot \boldsymbol{n} + c$
  $e_{max} \leftarrow \max(\|\boldsymbol{e}\|)$
  $i_{max} \leftarrow \max_i(\|\boldsymbol{e}\|)$
  $\boldsymbol{M}_1 \leftarrow \boldsymbol{M}(1 : i_{max})$
  $\boldsymbol{M}_2 \leftarrow \boldsymbol{M}(i_{max} : end)$
  if $e_{max} < 0.2 \cdot e_{corner}$ **then**
    $\boldsymbol{M}_1 \leftarrow \boldsymbol{M}$
    $\boldsymbol{M}_2 \leftarrow [0, 0]$
  else
    if $\text{length}(\boldsymbol{M}_1) < n_{edge}$ **then**
      $\boldsymbol{M}_1 \leftarrow \boldsymbol{M}_2$
      $\boldsymbol{M}_2 \leftarrow [0, 0]$
    else if $\text{length}(\boldsymbol{M}_2) < n_{edge}$ **then**
      $\boldsymbol{M}_2 \leftarrow [0, 0]$
    end if
  end if
end procedure

This algorithm takes a set of measurements in the XY plane, denoted by $\boldsymbol{M}$, a distance e_{corner}, and a minimum number of points for an edge n_{edge}. The principle is to determine if the LiDAR is detecting one or two edges, in order to divide the set accordingly and isolate each edge. Ideally, the boundary point at each end of the LiDAR scan corresponds to a corner of the section. A straight line in $\mathbb{R}^2$ can be defined as

$$p_x \cos(\gamma) + p_y \sin(\gamma) = \rho \tag{2.1}$$

where ρ is the perpendicular range from the line to the origin, γ is the angle to the positive X axis, and any point within is of the form $\boldsymbol{p} = \begin{bmatrix} p_x & p_y \end{bmatrix}^T$. Using this representation, the polar parameters of the line that contains the two boundary points can be found. In Cartesian coordinates, the perpendicular offset from a point to a line is given by

$$\boldsymbol{p}^T \boldsymbol{n} + c = e, \quad \text{with } \|\boldsymbol{n}\| = 1 \tag{2.2}$$

where $\boldsymbol{n} = \begin{bmatrix} n_x & n_y \end{bmatrix}^T$ is the unit vector normal to the line, c is the perpendicular offset from the line to the origin, and the error e is the actual offset, being null for points belonging to it. Expanding the vectorial notation of Equation (2.2) into X and Y coordinates, it is simple to obtain the Cartesian parameters from the polar ones. In the algorithm used in this work, the error e from every point on the data set to that line can then be calculated by direct substitution. The point furthest away will be the corner candidate, which upon confirmation can be used to split the data set into two smaller ones. This verification can be performed using the distance between a corner and the diagonal uniting the closest two other corners in a rectangle, defined as

$$e_{corner} = \frac{L_1 L_2}{\sqrt{L_1^2 + L_2^2}} \tag{2.3}$$

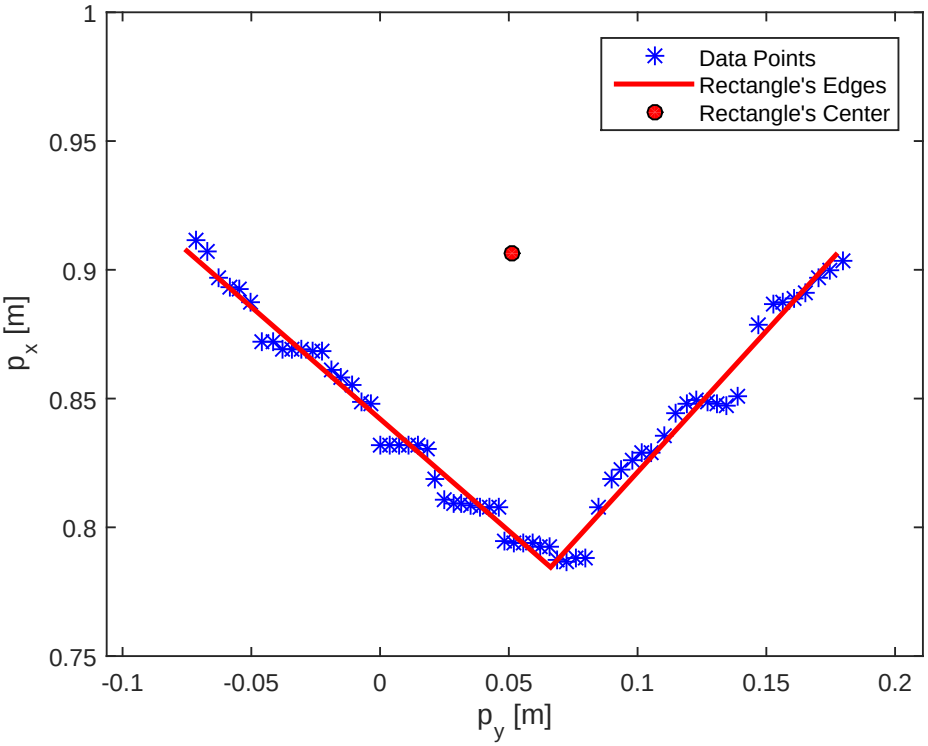
where L_i with $i = 1, 2$ are the dimensions of the pier, known beforehand. In order to distinguish whether the LiDAR is witnessing one or two edges of the pier, 20% of e_{corner} was established as the threshold, yielding the best outcomes upon the verification of a number of cases from several experiments, one of which being described in Section 2.3.2.2.

When the vehicle is facing one edge, the potential corner can be anywhere in the data set and its error e_{max} can still not be negligible as it would be expected, due to quantization errors. If that quantity stays below the threshold, there is most likely only one edge and both subsets have to be merged into the first whereas the second is left empty. In the remaining situations, the assumption is that there are two edges, but this can also happen due to the presence of boundary points resulting from reflections or measurement errors during the transition between one and two edges. Therefore, there is an additional condition regarding a required minimum number of 15 points in each subset, to ensure a sufficiently accurate edge. A subset is discarded when the amount of data points is not enough, otherwise both are kept and there are indeed two edges.

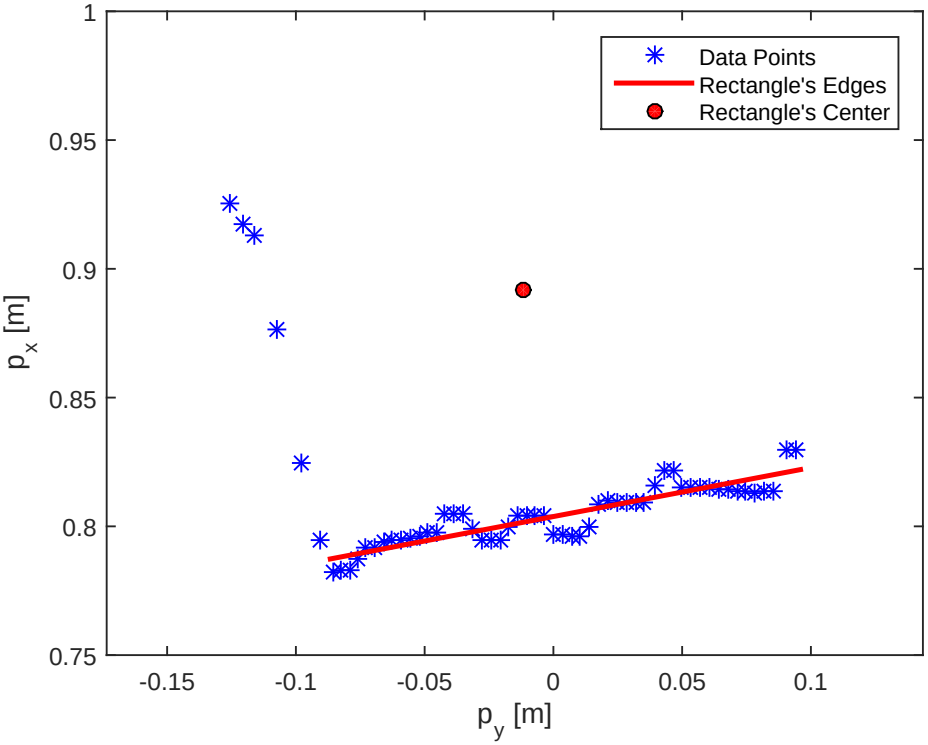
2.3.2.2 LiDAR Experiment #2

A second experiment using the LiDAR consisted of placing a cardboard box, with a 0.177×0.177 m square section, around 0.8 m from the quadcopter. The vehicle was moved manually, with the rotors off, in an approximately $90°$ counterclockwise arc around the box, while attempting to maintain null roll and pitch. The quadcopter started from a situation where both edges of the box made approximately a $45°$ angle to the longitudinal axis of the quadcopter, so the vehicle faced its diagonal at the beginning.

Figure 2.4 presents the output of the pier detection mechanism, in two different time instants of this experiment. The former is near the beginning, with both edges clearly visible, whereas the latter corresponds to the transition stage, where the algorithm helps deciding how many edges there are. For the sake of understanding what Algorithm 1 does, after selecting a potential corner and confirming its



(a) Two edges.



(b) One edge.

Figure 2.4: Cuboid pier detection during the second experiment.

legitimacy in both cases, the data points at the far left of Figure 2.4b were rejected for lacking in quantity. Continuing the movement around the pier, there would come a moment when the potential corner would not be accepted anymore, due to its proximity to the line uniting the two closest corners.

It is possible to confirm the measurement errors in both situations of Figure 2.4, evidenced by the resemblance of the data points to stairs at some points. In general, the line fits obtained are a really close match with what the data suggests to be the edges of the box. Nevertheless, during the transition stage and specially when facing only one edge, it is always expected to have larger errors than in the remaining situations, because in these cases the occurrence of reflections and other false readings is rather common and difficult to identify as outliers.

2.4 Outlier Identification and Removal

In the majority of locations around the pier, the detection strategy will consistently yield either one or two well defined edges. However, reflections of the beams and noise in the measurements generates outliers in the boundaries of the data set. With that in mind, Algorithm 2 was developed and applied before the edge separation, improving the effectiveness of the previous task. This algorithm is meant to filter the outliers from the beginning and end of the data set M, based on the relative distance between data points and two separate criteria f_{point} and $f_{cluster}$. The initial task is to determine the distance between one data point and the next, for the entire data set. The LiDAR will not always be at the same distance to the structure, as it should in order to allow flexibility and robustness in the face of change, therefore the magnitude of the point-to-point distance is bound to variate with the distance from the target. Additionally, running the entire data set every time step to check for outliers can become more computationally expensive than it needs to be. Relying on the observation of some tests, an estimation regarding the number of outliers present in the data set at a given moment was made considering its total size, yielding a linear empirical rule $n_{outliers} = m_{outliers}\, n_{set_{1/2}} + b_{outliers}$, where $m_{outliers} = 0.1$ is the slope, $n_{set_{1/2}}$ is half of the number of points in the data set, and $b_{outliers} = 3$ is the y-intercept.

At first sight, it might seem that because the number of points in an edge highly depends on the angle with which it is being faced by the LiDAR, the best idea would be to detect outliers in each one separately. However, the outliers do not appear with a well defined pattern, in fact it is extremely random and at times very difficult to actually distinguish between a valid and an invalid data point. In other words, the point density in each edge could not be significantly related with the distance between what is considered as a point belonging to a real edge and an outlier. This means that there was no noticeable difference in the distance of an outlier to the rest of the data set, when there was either a high or a low point density. Moreover, proceeding with this approach would defeat the entire purpose of suppressing the outliers before splitting the edges, which is one of the steps where the accuracy of the boundary points matters the most, in this case to decide how many edges there actually are.

Thus, the whole data set should be dealt with simultaneously. But this invalidates any attempt to do so, using the usual distribution tools to detect the centroid of a piece of data such as the mean and standard deviation or even the Median Absolute Deviation (MAD), as suggested in [15], due to

Algorithm 2 Trim Edges

procedure TRIMEDGES($\boldsymbol{M}, f_{point}, f_{cluster}$)
  for $i = 1 : \text{length}(\boldsymbol{M}) - 1$ **do**
    $\boldsymbol{d}(i) = \sqrt{(\boldsymbol{XY}(i+1,1) - \boldsymbol{M}(i,1))^2 + (\boldsymbol{M}(i+1,2) - \boldsymbol{M}(i,2))^2}$
  end for
  $n_{outliers} = m_{outliers}\ \text{length}(\boldsymbol{M})/2 + b_{outliers}$
  $\boldsymbol{d}_{min} = \text{sort}(\boldsymbol{d})$
  $d_{point_{max}} = f_{point}\, \boldsymbol{d}_{min}(1 : n_{outliers})$
  $d_{cluster_{max}} = f_{cluster}\, d_{point_{max}}$
  $i = 1$
  while $i \leq n_{outliers}$ **do**
    if $\boldsymbol{d}(i) > d_{cluster_{max}}$ **then**
      $\boldsymbol{d}(1 : i) = []$
      $\boldsymbol{M}(1 : i, :) = []$
      $i = 0$
    end if
    $i = i + 1$
  end while
  for $i = 1 : n_{outliers}$ **do**
    if $\boldsymbol{d}(i) > d_{point_{max}}$ **then**
      $\boldsymbol{M}(1, :) = []$
    else
      break
    end if
  end for
  $i = 1$
  while $i \leq n_{outliers}$ **do**
    if $\boldsymbol{d}(end - i + 1) > d_{cluster_{max}}$ **then**
      $\boldsymbol{d}(end - i + 1 : end) = []$
      $\boldsymbol{M}(end - i + 1 : end, :) = []$
      $i = 0$
    end if
    $i = i + 1$
  end while
  for $i = 1 : n_{outliers}$ **do**
    if $\boldsymbol{d}(end - i + 1) > d_{point_{max}}$ **then**
      $\boldsymbol{M}(end, :) = []$
    else
      break
    end if
  end for
end procedure

the difference in point density with the motion around the structure. Instead, a way to account for the resolution of the LiDAR, depending on the distance from the target, is to find the minimum distance between points, in the entire data set. But because only one value can easily be an exception, the $n_{outliers}$ smallest distances were averaged out to provide a more accurate measure of this indicator.

It is worth mentioning that these outliers can appear either as isolated points, which is more common, or small clusters of points, hence the existence of two different criteria. This leads to two different stages of trimming, for both the beginning and the end of the data set, the first for small clusters and the second for the remaining isolated points. The factors were set as $f_{point} = 5$ and $f_{cluster} = 6$, based on experimental verification of a number of cases, where the latter is actually cumulative.

The final step is to go through the data set, from the outside to the inside at both ends, and remove the points or groups of points whose distance to the next exceeds the respective thresholds. Each of the two sequences stops as soon as there is one point that respects the threshold in the current direction,

which in addition to the limitation on the number of outliers, would prevent wrongful removals deeper in the middle of the data set, due to quantization errors or other unforeseen factors.

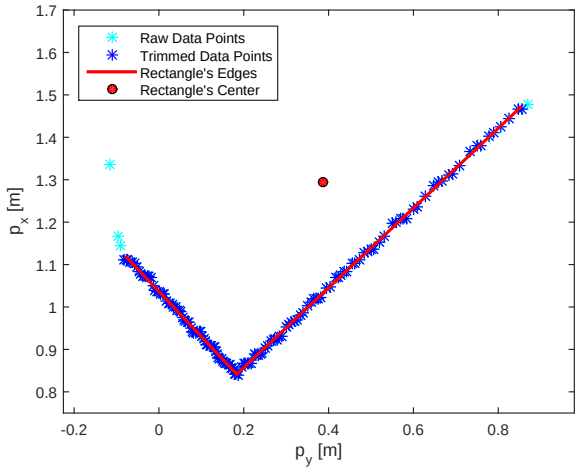
In order to provide an insight into the decision processes involved in this task, several examples of LiDAR scans are given in Figures 2.5 and 2.6, for some of the most uncommon yet distinctive cases that were found during the course of a comprehensive testing period.

Starting with a rectangular sectioned pier, it is interesting to notice the relationship between point density and dimension for each edge in Figure 2.5a, which makes it difficult to treat both of them separately based on relative distances, despite that being an intuitive tool available in this task. While moving around the pier, Figure 2.5b shows the captured data points, when one of the edges is becoming occluded, as well as their unbalanced increase in relative distance. In situations similar to Figure 2.5c, the appearance of such outliers is very frequent and can lead to a wrongful separation of the data set by interfering with the threshold based on e_{corner}. Together with the previous case, the event depicted at Figure 2.5d is one of the most severe cases if left untreated, since there is a cluster of three points at the right and another point even further, none of them corresponding to actual intersections of the laser beams with the pier. These are most likely the result of reflections, whereas the remaining three points in that side are the only ones possibly real, but given their scarcity they would not yield an accurate fix on the edge and should not be considered. Additionally, at the time of this test there was no specialized filter for clusters, however the points within that batch had a sufficiently large relative distance to be detected as regular outliers, when considering the high point density typical of the valid data points with an angle so close to $90°$.

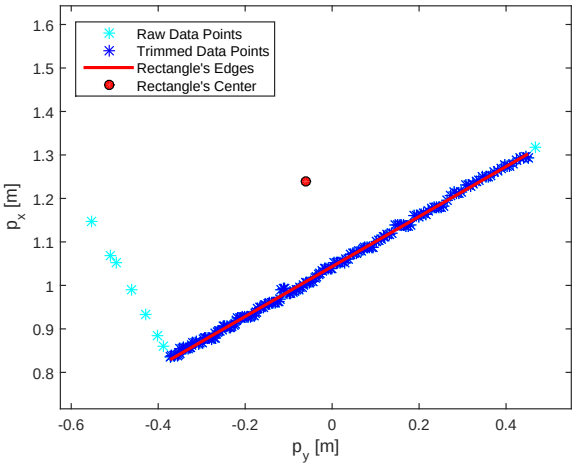
The two cases in Figure 2.6 portray the detection of a pier with a square section and are perfect examples of situations that could hardly be predicted. In both of them, the outlier closest to the sensor was actually perceived as the data point furthest to the left, i.e. as the last point of the scan, and the outliers above were detected right before, after making a tight curve at the tip of the edge. This causes the length of the associated edge to decrease dramatically, increasing the chances of being rejected later. Furthermore, when this occurs in opposite edges in consecutive scans and they get rejected, it would be interpreted as an instantaneous $90°$ jump, which is impossible. Despite the apparent resemblance between both cases, the event in Figure 2.6b was actually the scan that originated the cluster removal strategy. Looking closely, there are two data points almost overlapping in what could be easily interpreted as only the closest outlier, at the left. Leaving that cluster would cause losing around a third of the length of the edge, resulting in its rejection and in a angle shift after the next scan.

2.5 Geometry Fitting

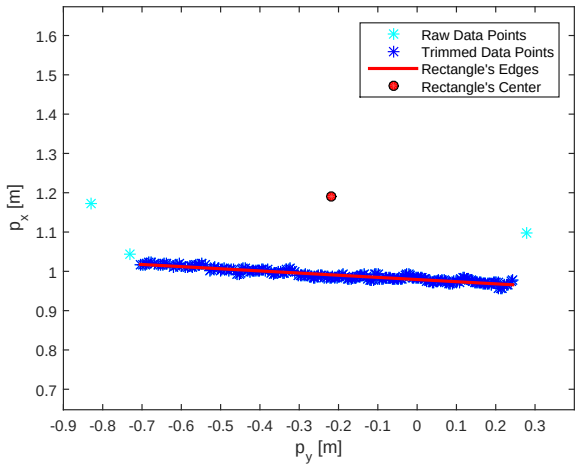
After the data set separation, the identification implies defining the equation of the line for each existing edge. For this application, two main alternatives were chosen to achieve the objective, one based on least squares estimation and the other using the Hough transform, both being described in Sections 2.5.1 and 2.5.2 respectively. In the end, Section 2.5.3 compares their efficiency, to rule the verdict on which is more suited for this case, and applies the best choice to the data.



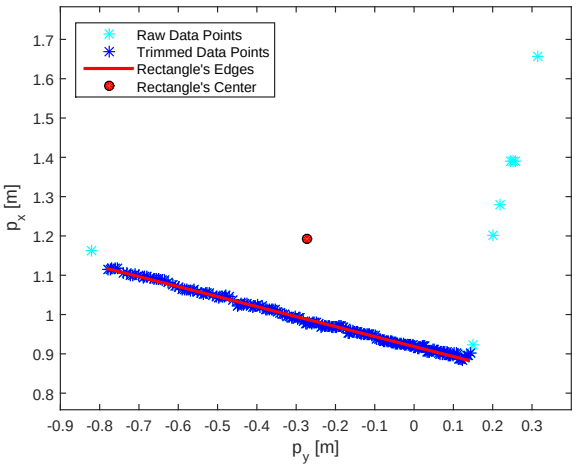
(a) Holding in the initial position at approximately 45°.



(b) Losing the smaller edge at the left.

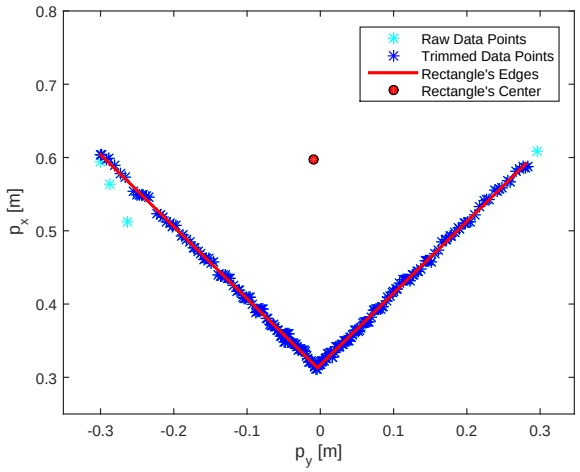


(c) Facing the larger edge almost at 90°.

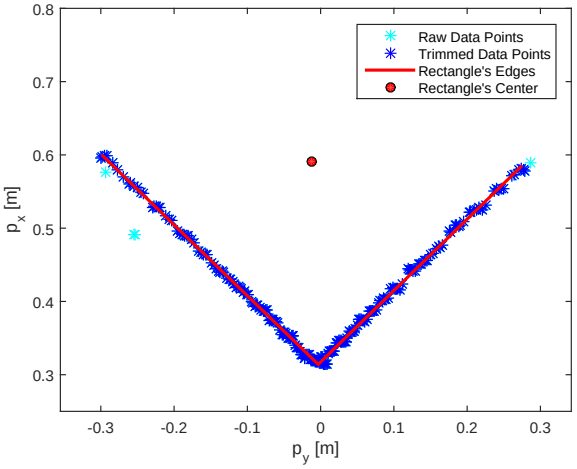


(d) Acquiring another smaller edge at the right.

Figure 2.5: Rectangular pier detection with outlier trimming
in several notoriously unusual cases of a 0.45×0.9 m section.



(a) Single outliers at inconsistent locations.



(b) Mixed single and groups of outliers
at inconsistent locations.

Figure 2.6: Square pier detection with outlier trimming
in several notoriously unusual cases of a 0.425×0.425 m section.

2.5.1 Least Squares Estimation

The problem at hand is to minimize the sum of the squares of the perpendicular offset between the points and fitted line. Therefore, determining the line for which that quantity is minimal comes down to the constrained least squares problem in the form

$$\begin{aligned} \underset{c,n_x,n_y}{\text{Minimize}} \quad & \|\boldsymbol{e}\|^2 = \sum_{i=1}^{N} e_i^2 \\ \text{Subject to} \quad & \begin{bmatrix} 1 & x_1 & y_1 \\ \vdots & \vdots & \vdots \\ 1 & x_N & y_N \end{bmatrix} \begin{bmatrix} c \\ n_x \\ n_y \end{bmatrix} = \begin{bmatrix} e_1 \\ \vdots \\ e_N \end{bmatrix}, \\ & n_x^2 + n_y^2 = 1 \end{aligned} \tag{2.4}$$

which can also be found in [16]. In the first constraint of this optimization, the matrix of the system is $\boldsymbol{A}$, the vector of unknowns is $\boldsymbol{x}$ and the right hand side is $\boldsymbol{e}$. In order to account for the nonlinear constraint, a QR decomposition such that $\boldsymbol{A} = \boldsymbol{QR}$ with $\boldsymbol{Q}^T\boldsymbol{Q} = \boldsymbol{I}$ and $\boldsymbol{R}$ is upper triangular, can reduce the problem to solving the reduced system

$$\boldsymbol{Ax} = \boldsymbol{e} \Leftrightarrow \boldsymbol{Q}^T\boldsymbol{Ax} = \boldsymbol{Q}^T\boldsymbol{e} \Leftrightarrow \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ 0 & r_{22} & r_{23} \\ 0 & 0 & r_{33} \\ 0 & 0 & 0 \\ \vdots & \vdots & \vdots \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} c \\ n_x \\ n_y \end{bmatrix} = \boldsymbol{y} \tag{2.5}$$

This is possible since the the 2-norm is invariant under orthogonal transformations $\boldsymbol{y} = \boldsymbol{Q}^T\boldsymbol{e}$, so $\|\boldsymbol{y}\| = \|\boldsymbol{e}\|$. Because the nonlinear constraint only involves two unknowns, Equation (2.5) yields the problem

$$\begin{aligned} \underset{c,n_x,n_y}{\text{Minimize}} \quad & \begin{bmatrix} r_{22} & r_{23} \\ 0 & r_{33} \end{bmatrix} \begin{bmatrix} n_x \\ n_y \end{bmatrix} \approx \begin{bmatrix} 0 \\ 0 \end{bmatrix} \\ \text{Subject to} \quad & n_x^2 + n_y^2 = 1 \end{aligned} \tag{2.6}$$

This new system can be seen as a minimization of $\|\boldsymbol{Bx}\|$, subject to $\|\boldsymbol{x}\| = 1$. The value of the minimum is the smallest singular value of $\boldsymbol{B}$ and the solution is the corresponding singular vector. Then, n_x and n_y can be determined by the Singular Value Decomposition (SVD) of a 2×2 matrix. In the end, the value of c can be found by setting $\boldsymbol{y} = 0$ in Equation (2.5).

2.5.2 Reduced Space Hough Transform

The Hough transform is a feature extraction technique used in image analysis, computer vision, and digital image processing [17]. The purpose of this technique is to find imperfect instances of objects,

within a certain class of shapes, by a voting procedure. This voting procedure is carried out in a parameter space, from which object candidates are obtained as local maxima, in a so-called accumulator space that is explicitly constructed by the algorithm for computing the Hough transform. The Hough transform has been extended to identify positions of arbitrary shapes, most commonly circles or ellipses, but its simplest case is detecting straight lines.

There are several representations for a straight line in $\mathbb{R}^2$, one of which has already been presented in Equation (2.1). Besides that, there is also the slope and y-intercept, as one of the most commonly used, where the equation takes the form $p_y = m_{line}\, p_x + b_{line}$. Out of the two, the polar representation is the most intuitive and easy to apply, considering a search in a restricted range of parameters. Furthermore, vertical lines pose a problem with the y-intercept representation, as unbounded values of the slope parameter m would arise.

The Hough transform algorithm uses an accumulator to detect the existence of a line, through the polar representation. The dimension of the accumulator equals the number of unknown parameters, in this case γ and ρ. For each point $\boldsymbol{p}$, the idea is to determine if it is contained in the line described by each set of polar parameters, from the defined accumulator space. If so, it will look for the corresponding accumulator's bin, where the parameters fall into, and increment its value. By finding the bins with the highest values, typically by looking for local maxima in the accumulator space, the most likely lines can be extracted and their approximate geometric definitions acquired [17].

The final result of the linear Hough transform is a two-dimensional array, where one dimension of this matrix is the quantized angle γ and the other dimension is the quantized offset ρ. Each element of the matrix has a value equal to the number of points that are positioned on the corresponding line, so the element with the highest value indicates the most recurrent straight line [18]. A reduced space version of this technique is implemented in Algorithm 3 and exemplified graphically in Figure 2.7.

Algorithm 3 Reduced Space Hough Transform

procedure REDUCEDSPACEHOUGHTRANSFORM($\boldsymbol{M}, n_{rows}, n_{cols}, \gamma, \delta_\gamma, \epsilon_\gamma, \rho, \delta_\rho, \epsilon_\rho$)
  $\boldsymbol{H} \leftarrow \mathbf{0}_{n_{rows} \times n_{cols}}$
  $\boldsymbol{\gamma} \leftarrow (\gamma - \delta_\gamma : \epsilon_\gamma : \gamma + \delta_\gamma)^T$
  $\boldsymbol{\rho} \leftarrow (\rho - \delta_\rho : \epsilon_\rho : \rho + \delta_\rho)$
  for $i = 1 : \mathrm{length}(\boldsymbol{M})$ **do**
    $\boldsymbol{\rho}_{point} \leftarrow \boldsymbol{M}(i,1)\cos(\boldsymbol{\gamma}) + \boldsymbol{M}(i,2)\sin(\boldsymbol{\gamma})$
    $\boldsymbol{\Delta\rho} \leftarrow \boldsymbol{\rho}_{point}\, \mathbf{1}_{1 \times n_{cols}} - \mathbf{1}_{n_{rows} \times 1}\, \boldsymbol{\rho}$
    $\boldsymbol{\Delta\rho}_{min} \leftarrow \min(\|\boldsymbol{\Delta\rho}\|)$
    $\boldsymbol{i}_{min} \leftarrow \min_i(\|\boldsymbol{\Delta\rho}\|)$
    $\boldsymbol{i}_{min}(\boldsymbol{\Delta\rho}_{min} > \frac{\epsilon_\rho}{2}, :) \leftarrow []$
    $\boldsymbol{H}(\boldsymbol{i}_{min}) \leftarrow \boldsymbol{H}(\boldsymbol{i}_{min}) + 1$
  end for
  $\rho, \gamma \leftarrow \max_{i,j}(\boldsymbol{H})$
  $\boldsymbol{n} \leftarrow [\cos(\gamma), \sin(\gamma)]^T$
  $c \leftarrow -\rho$
end procedure

Before running this algorithm, some relevant quantities regarding the accumulator space need to be defined. In this case, a very important information is the initial estimation of the line, or in the context of

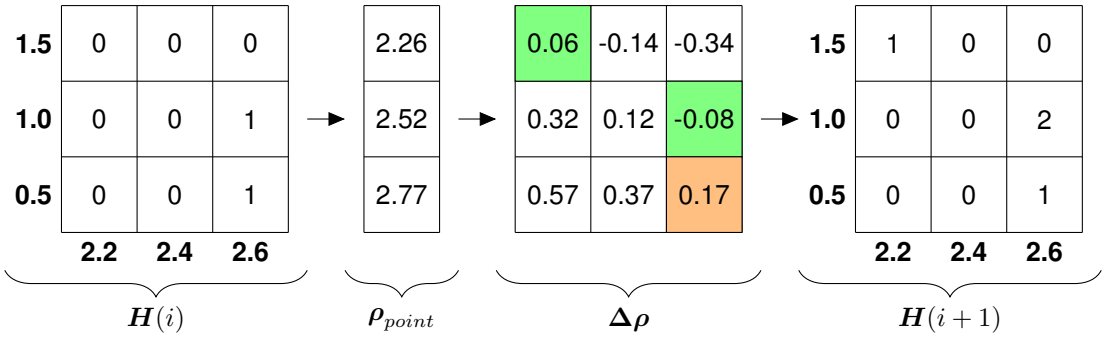


Figure 2.7: Example of an iteration of the reduced space Hough transform, with $\delta_\gamma = 1°$, $\epsilon_\gamma = \frac{\delta_\gamma}{2}$[°], $\delta_\rho = 0.4$ m, and $\epsilon_\rho = \frac{\delta_\rho}{2}$ [m].

the Hough transform, the center of the accumulator space around which the polar parameters will vary. A possibility is to split in half the data subset associated with an edge and determine the centroid of each part. From there, these two centroids can be used to obtain a fast and rough estimate of the polar parameters. Now that the central values are defined, it is necessary to decide what is the range δ_γ and δ_ρ allowed for each parameter to variate and the step ϵ_γ and ϵ_ρ between two consecutive values. After observing some cases, it was seen that setting $\delta_\gamma = 1.5°$, $\epsilon_\gamma = \frac{\delta_\gamma}{15}$[°], $\delta_\rho = 0.025$ m, and $\epsilon_\rho = \frac{\delta_\rho}{10}$ [m] yielded a suited accumulator space with the sufficient flexibility. The dimensions n_{rows} and n_{cols} of the accumulator space can then be easily found.

Algorithm 3 receives the data set $\boldsymbol{M}$ and all these aforementioned settings. The initial step inside the routine is to initialize the accumulator $\boldsymbol{H}$ and expand the definition of γ and ρ to contain all the possible values, where the former is a column vector and the latter a row vector. Then, an iteration is run for every point, in which knowing every possibility for γ, the required ρ_{point} for the line to contain the point is computed. There will be only one ρ_{point} per line, i.e. for each value of γ, so $\boldsymbol{\Delta\rho}$ will now become a matrix with the difference between this required $\boldsymbol{\rho}_{point}$ and the allowed $\boldsymbol{\rho}$ vector, for each line. Therefore, the best suited bin within the accumulator will correspond to the minimum values of that differential. This is done after ensuring that the minimum for each line is in fact lower than half the space between bins, otherwise the first or last bin would be incremented because ρ_{point} did not fit in any, i.e. it was higher or lower than the minimum or maximum allowed values respectively, but ended up being the closest one. Finally, the optimal polar parameters can be found by locating the bin with the highest value in the accumulator, after which they can be converted to its Cartesian representation.

2.5.3 Procedure Selection and Application

Both these procedures were applied and produced coherent results, however there are some differences in their performance. On one hand, the least squares estimation is the fastest, whereas the reduced space Hough transform takes $5 - 10$ times longer to complete, due to its iterative nature. On the other hand, the former is highly affected by existing outliers, as it tries to minimize the offset of all points to the estimated line, while the voting mechanism in the latter is very efficient in detecting the points that are most likely valid and adjusting the line only to them.

In terms of effectiveness, despite definitely being a more accurate solution, the improvements of the reduced space Hough transform against the least squares estimation translate into an almost negligible change in the angle of the line, in the majority of cases. Therefore, the bottleneck in this situation comes down to time. Since this procedure has to be applied to every scan of the LiDAR, the time difference between these methods causes the Hough transform to become an inviable alternative, leading to the choice of the least squares estimation.

After obtaining the line parameters for an edge, out of the entire data subset the only necessary data points are the boundaries and the corner, if it exists. Unfortunately, the boundary points are the most affected by uncertainties and measurement errors, what can never be truly avoided. These relevant data points need to be corrected to $\overline{\boldsymbol{p}}$ according to the obtained line fit through

$$\overline{\boldsymbol{p}} = \boldsymbol{p} - \boldsymbol{n}e = \boldsymbol{p} - \boldsymbol{n}(\boldsymbol{p}^T\boldsymbol{n} + c) \tag{2.7}$$

in order to determine where they should have been in the first place. It should be noted that, if there are two edges and consequently a corner, this point is shared by both edges and therefore needs to be corrected using the information from the two fits to provide the best estimation of the intersection.

After correcting the data points, the estimation of the length of the edges can be made. Having two edges, the following requirement is that both have to be larger than f_{length} times the smaller real dimension, otherwise the smallest edge should be disregarded. This factor varies from 70 – 90% and has to be tuned according to the circumstances. This verification occurs in order to exclude exceptional cases where the data set identification's conditions were passed but the result does not have true physical meaning.

2.6 Complete Representation

The corrected boundary points and corner, if it exists, can be identified as the start and end points of each edge, denoted here by $\boldsymbol{p}_{s_i}$ and $\boldsymbol{p}_{e_i}$ with $i = 1, 2$ respectively. The edges are represented by a matrix $\boldsymbol{Q}$, expressed in $\{B\}$, where its columns are the vectors $\boldsymbol{Q}_i = \begin{bmatrix} Q_{i_x} & Q_{i_y} \end{bmatrix}^T$ with $i = 1, 2$, such that $\boldsymbol{Q}_i = \boldsymbol{p}_{e_i} - \boldsymbol{p}_{s_i}$. Notice that all these variables are time-varying and depend on the motion of the vehicle. In particular, the correspondence between boundary points and corner and the points $\boldsymbol{p}_{s_i}$ and $\boldsymbol{p}_{e_i}$ can be obtained by knowing the initial condition, while encountering the pier, and observing it evolve over time.

Planning a trajectory and navigating around a pier is a task highly dependent on the location of its center, since it is the pier's most logical fixed point for maintaining a reference. This location is ideally invariant, relative to the movement of the vehicle, and it can be determined at all times given the real geometry of the pier and the knowledge regarding the number and details of the edges being encountered. With that information, an estimate of the center location can be computed and later used for control tasks, concerning the navigation around the structure. However, this measurement will unavoidably fluctuate around the real location, which can cause some oscillations during the controlled flight.

The entire process is summarized as a flow diagram in Figure 2.9, highlighting the most important details of the stages that have been described along the current chapter.

The representation of the edges in $\{I\}$ is also of great interest, so that they can be related through the rotation matrix from $\{I\}$ to $\{B\}$, denoted by ${}^{B}_{I}\boldsymbol{R}$, according to ${}^{I}\boldsymbol{Q} = {}^{I}_{B}\boldsymbol{R}\,{}^{B}\boldsymbol{Q}$. As illustrated in Figure 2.8, their projection in the XY plane of this reference frame corresponds to the section of the pier and can be defined as $\boldsymbol{l}_i$, where its norm is L_i with $i = 1, 2$. Regarding the Z coordinate, represented by h_i with $i = 1, 2$, in $\{B\}$ it is null but in $\{I\}$ it is the direct result of the rotation the vehicle is exhibiting momentarily. Therefore, the edges in $\{I\}$ become

$$ {}^{I}\boldsymbol{Q}_i = \boldsymbol{l}_i \pm h_i \boldsymbol{e}_3, \text{ for } i = 1, 2 \tag{2.8} $$

where $\boldsymbol{e}_3 = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T$ is the versor of the Z axis. Knowing the dimensions of the pier and the length of the edges, which is independent of the reference frame, the Z coordinate can be obtained through

$$ h_i = \sqrt{\|{}^{B}\boldsymbol{Q}_i\|^2 - L_i^2}, \text{ for } i = 1, 2 \tag{2.9} $$

There are two additional aspects that need to be clarified. Because the LiDAR measurements are subject to errors, when the roll and pitch angles are close to null, it is possible that the identified edges are smaller than reality. That would cause the Z coordinate calculation to yield an imaginary value, therefore this component is taken as zero when such a situation occurs. Moreover, because the aforementioned calculation is made with a square root, there are always two options for the value, according to the sign. The approach to solve this ambiguity aims to maintain continuity, by choosing the closest value to the previous one, assuming there are no swift movements around the leveled flight.

Testing this approach shows that the output of this calculation is more than often zero, meaning that the edges are being seen as smaller than in reality. This will always be a concern for any roll or pitch angles other than zero, due to the way the LiDAR works, since the last laser beam to intercept an edge

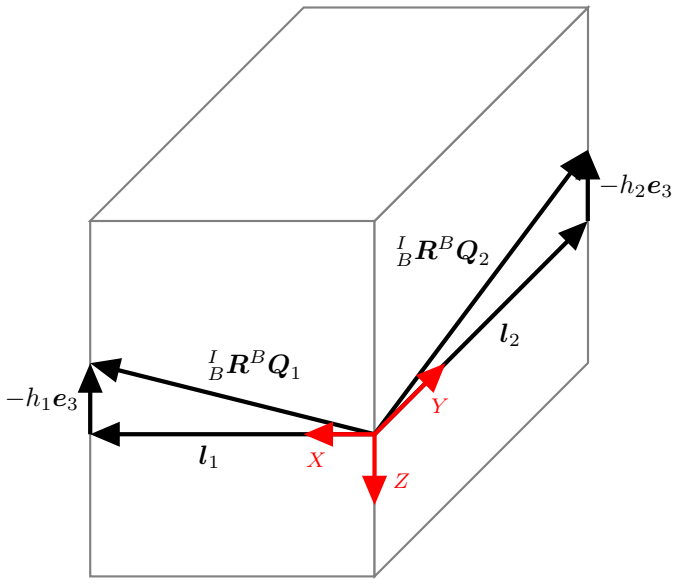


Figure 2.8: Decomposition of the edges in $\{I\}$.

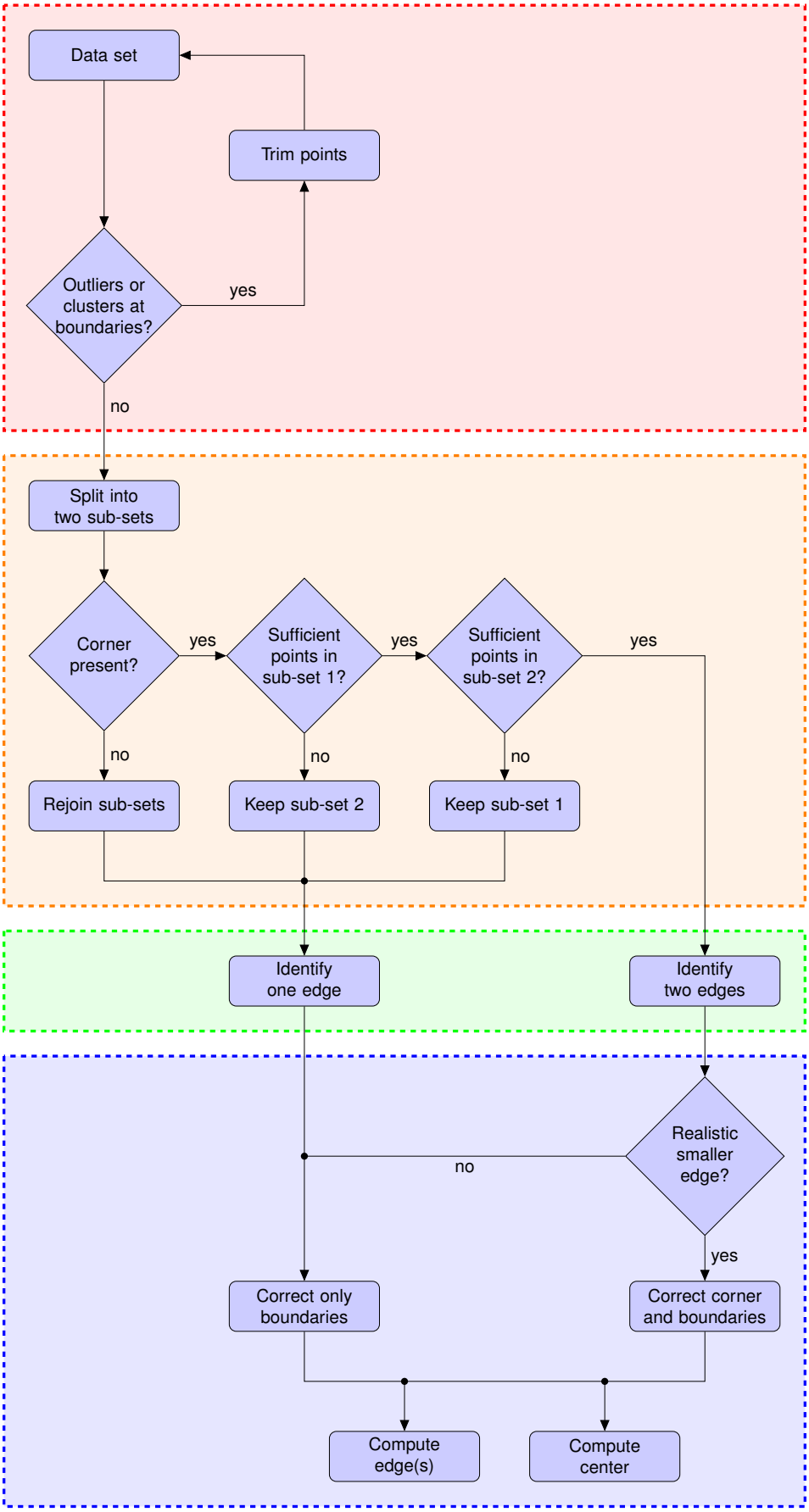


Figure 2.9: Flow diagram of the process of obtaining the pier's edges and center in $\{B\}$, distinguishing the trimming (red), the splitting (orange), the fitting (green), and the processing (blue).

hits either exactly or a bit before the boundary. Therefore, assuming an optimal outlier removal, the detected length is bound to be less than or equal to the real length, whereas most of the times it will be slightly smaller. When the roll and pitch angles are close to zero, this effect is more noticeable because the calculation will involve imaginary numbers and therefore yield a null Z coordinate for default.

However, this calculation only considers the length of the edges. One way to improve this component's detection is to use more information and create an optimization problem. An updated formulation would contain Equation (2.9), for each edge, extended with the cross product of both edges, with information on the angle between them. Each edge can be seen as the sum of two vectors, one known in the XY plane and another in the Z axis trying to be found. Thus, using the distributive property of the cross product, the crossed term involving the parallel vertical portion of both edges would disappear, keeping this as a linear problem. The objective of the optimization, formulated as

$$\begin{aligned} &\underset{h_1^2,h_2^2}{\text{Minimize}} \quad \|\boldsymbol{\varepsilon}\|^2 = \sum_{i=1}^{3} \varepsilon_i^2 \\ &\text{Subject to} \quad \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ L_2^2 & L_1^2 \end{bmatrix} \begin{bmatrix} h_1^2 \\ h_2^2 \end{bmatrix} - \begin{bmatrix} \|{}^B\boldsymbol{Q}_1\|^2 - L_1^2 \\ \|{}^B\boldsymbol{Q}_2\|^2 - L_2^2 \\ \|S({}^B\boldsymbol{Q}_1){}^B\boldsymbol{Q}_2\|^2 - L_1^2 L_2^2 \end{bmatrix} = \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \varepsilon_3 \end{bmatrix}, \\ &\qquad\qquad h_i^2 \geq 0 \,\text{for}\, i = 1, 2 \end{aligned} \tag{2.10}$$

is to minimize an error variable $\boldsymbol{\varepsilon}$, relating properties of the edges in $\{I\}$ and $\{B\}$. In Equation (2.10), the $S(.)$ function produces a skew-symmetric matrix, defined such that, for any vector $\boldsymbol{\nu}$, the $S(\boldsymbol{\nu}_1)\boldsymbol{\nu}_2 = \boldsymbol{\nu}_1 \times \boldsymbol{\nu}_2$ equality verifies.

It should be noted that the variables being optimized are the squares of the Z coordinates, naturally being subject to non-negativity constraints, so in the end the desired result will be the square root of their values. This also means the previous continuity verification still applies, given the sign ambiguity of this operation. Originally with this approach, the length of the edges in $\{B\}$ had a lower boundary established, corresponding to their minimum nominal value, so when the obtained lengths were smaller than their real dimension then they would be reset to that value, to prevent the aforementioned issues with the imaginary results. However, using these constraints in the problem formulation eliminated the need for that verification.

In this optimization, the third equation is clearly the most valuable, for containing information about both edges simultaneously, making it the condition with the highest accuracy. An attempt was made to convert this optimization into a weighted form, where each error component would have a different importance factor assigned. This way, the relationship between them would drive the effort of the optimization in minimizing those highest in ranking, which would have a larger error. In this case, a weights vector $\boldsymbol{w} = \begin{bmatrix} 0.5 & 0.5 & 1.5 \end{bmatrix}$ was applied, giving more relevance to the last equation. This course of action brought some improvements to the results, however implementing this new optimization slowed down the whole process, due to its computational complexity. In the end, the decision was to maintain the original scheme, since the improvements were not high enough when compared with the loss in speed of execution.

An additional idea to make the optimization more robust and improve its result can be investigated by assuming that, between consecutive time steps, the change in the Z coordinates is very small. Considering that assumption, which is fairly reasonable for a short number of time steps, the n previous samples can be incorporated into the current formulation. In other words, for the optimization of the i^{th} step, the constraints used for the n samples before that step could also be added to the current one to provide a more stable and consistent result, creating a backward dependency on the condition but not on the result itself. Of course, the number of samples n would be limited and for each step that value would need to be further restrained to the number of available coherent measurements, i.e. corresponding to not only the same relative position of edges but also to the same number of identified edges. These requirements are essential, the former since in a corner shift the edges do not maintain the same relationship between steps and the latter because in the transitory stage the fading edge is intermittent, which means there would not be continuity between observations, for that edge, and therefore the optimization would have a different degree of accuracy for each edge. In the end, this means that until a maximum of n previous samples, those holding the proper conditions could be used in the optimization, at each time step. Expanding the amount of information used in the optimization shows a slight improvement of the results for increasing n, thus it was decided that using 4 backward coherent samples would be the best compromise between the optimization's stability and the vehicle's motion flexibility.

Chapter 3

Attitude Determination Methods

The procedures involved in the detection, and consequent creation of a clear and intuitive mathematical description of a structure, were elaborated on during Chapter 2. The next step is to analyze this description and propose several methods capable of accurately extracting a partial or the full attitude of the vehicle.

The first approach is to consider a decoupling of the vehicle's movement into the XY plane and the Z axis. Taking that into account, Section 3.1 describes a method for obtaining each of these two partial movements using information from the LiDAR, where the second requires a high-level data fusion from another sensor.

Section 3.2 goes one step further and elaborates on two additional options to compute the attitude of the vehicle, this time considering the 3D world altogether. Both alternatives attempt at fusing data at a low-level from other available sources, where the first is an out-of-the-box solution and the second is developed specifically for this purpose, with the goal of getting improved results.

3.1 Partial Motion

The movements of the vehicle in the X and Y axes are the roll and pitch respectively, which can be described based solely on LiDAR data as Section 3.1.1 shows. For that, the rotation matrix from $\{I\}$ to $\{B\}$ needs to be derived, since it involves the two most important reference frames in this task. Knowing the description of the edges in $\{B\}$, it becomes only a matter of relating those two pieces of information, until the roll and pitch angles can be obtained.

On the other hand, the yaw movement is associated with the Z axis. This component of the vehicle's attitude can be computed as presented in Section 3.1.2, where information from the LiDAR is fused with information from an IMU. This happens because on its own, the LiDAR data has the effect of the roll and pitch movements embedded, and although those components have been chosen not to be retrieved in this approach, their traces still need to be removed from the measurements before processing it. Therefore, the way to obtain the roll and pitch angles from the IMU is explained first, followed by the actual discussion of the method that yields the yaw angle.

3.1.1 Roll and Pitch Estimator

The edges are detected with the LiDAR in the $\{B\}$ frame, but their dimension is known and corresponds to the norm of ${}^{H}\boldsymbol{Q}_i = \boldsymbol{l}_i$ with $i = 1, 2$. The rotation matrix from $\{B\}$ to $\{H\}$, defined as

$$\begin{aligned} {}^{H}_{B}\boldsymbol{R} &= \boldsymbol{R}_{xy} = \boldsymbol{R}_y(\theta)\boldsymbol{R}_x(\phi) \\ &= \begin{bmatrix} c_\theta & 0 & s_\theta \\ 0 & 1 & 0 \\ -s_\theta & 0 & c_\theta \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & c_\phi & -s_\phi \\ 0 & s_\phi & c_\phi \end{bmatrix} = \begin{bmatrix} c_\theta & c_\theta s_\phi & s_\theta c_\phi \\ 0 & c_\phi & -s_\phi \\ -s_\theta & c_\theta s_\phi & c_\theta c_\phi \end{bmatrix} \end{aligned} \tag{3.1}$$

where s_ϕ and c_ϕ represent the sine and cosine of the roll angle ϕ respectively and the same applies to the pitch angle θ, can be used to obtain the projection of the edges

$$\|\boldsymbol{\Pi}_{\boldsymbol{e}_3}\,{}^{H}\boldsymbol{Q}_i\|^2 = \|\boldsymbol{\Pi}_{\boldsymbol{e}_3}\,{}^{H}_{B}\boldsymbol{R}\,{}^{B}\boldsymbol{Q}_i\|^2 = L_i^2, \text{ for } i = 1, 2 \tag{3.2}$$

where $\boldsymbol{\Pi}_{\boldsymbol{e}_3} = \boldsymbol{I} - \boldsymbol{e}_3\boldsymbol{e}_3^T$. Since ${}^{H}_{B}\boldsymbol{R}$ is a function of both these angles, they can be estimated with (3.2). Knowing the elements of this matrix, from Equation (3.1), the relationship between the angles and the edges ${}^{B}\boldsymbol{Q}$ is given by

$$\begin{aligned} \begin{bmatrix} {}^{H}_{B}r_{13} \\ {}^{H}_{B}r_{23} \end{bmatrix} &= \begin{bmatrix} s_\theta c_\phi \\ -s_\phi \end{bmatrix} = \begin{bmatrix} {}^{B}\boldsymbol{Q}_1 & {}^{B}\boldsymbol{Q}_2 \end{bmatrix}^{-1} \begin{bmatrix} \pm\sqrt{\|{}^{B}\boldsymbol{Q}_1\| - L_1^2} \\ \pm\sqrt{\|{}^{B}\boldsymbol{Q}_2\| - L_2^2} \end{bmatrix} \\ \Leftrightarrow \begin{bmatrix} \theta \\ \phi \end{bmatrix} &= \begin{bmatrix} \sin^{-1}\left(\frac{\pm Q_{2_y}\sqrt{Q_{1_x}^2+Q_{1_y}^2-L_1^2} \mp Q_{1_y}\sqrt{Q_{2_x}^2+Q_{2_y}^2-L_2^2}}{(Q_{1_x}Q_{2_y}-Q_{1_y}Q_{2_x})c_\phi}\right) \\ \sin^{-1}\left(\frac{\pm Q_{2_x}\sqrt{Q_{1_x}^2+Q_{1_y}^2-L_1^2} \mp Q_{1_x}\sqrt{Q_{2_x}^2+Q_{2_y}^2-L_2^2}}{Q_{1_x}Q_{2_y}-Q_{1_y}Q_{2_x}}\right) \end{bmatrix} \end{aligned} \tag{3.3}$$

It should be noted that the sign ambiguity occurs because there is a symmetry along the XY plane, therefore Equation (3.2) is valid whenever the vehicle is detecting data points on the pier above its altitude or their symmetric below.

However, this method heavily relies on the length of the edges, which greatly depends on the boundary points, that in turn are subject to the highest measurement errors. Despite the lengths not being accurate enough, the angle between the edges is also affected by the motion of the vehicle but it was not used yet. This quantity does not depend solely on a pair of points, as the lengths, rather it is the result of fitting an entire set of points, thus carrying less uncertainty.

3.1.2 Yaw Estimator

The output from the IMU is the rotation matrix from $\{I\}$ to $\{S\}$. Given that this sensor is mounted with its X and Z axes pointing forward and upward respectively, one additional operation needs to be made in order to obtain the actual ${}^{B}_{I}\boldsymbol{R}$. It consists of multiplying the initial rotation matrix

$$
\begin{aligned}
{}^{S}_{I}\boldsymbol{R} &= \boldsymbol{R}_{zyx} = \boldsymbol{R}_x(-\phi)\boldsymbol{R}_y(-\theta)\boldsymbol{R}_z(-\psi) \\
&= \begin{bmatrix} 1 & 0 & 0 \\ 0 & c_\phi & s_\phi \\ 0 & -s_\phi & c_\phi \end{bmatrix} \begin{bmatrix} c_\theta & 0 & -s_\theta \\ 0 & 1 & 0 \\ s_\theta & 0 & c_\theta \end{bmatrix} \begin{bmatrix} c_\psi & s_\psi & 0 \\ -s_\psi & c_\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} c_\psi c_\theta & c_\theta s_\psi & -s_\theta \\ c_\psi s_\phi s_\theta - c_\phi s_\psi & c_\phi c_\psi + s_\phi s_\psi s_\theta & c_\theta s_\phi \\ s_\phi s_\psi + c_\phi c_\psi s_\theta & c_\phi s_\psi s_\theta - c_\psi s_\phi & c_\phi c_\theta \end{bmatrix}
\end{aligned} \tag{3.4}
$$

with ${}^{B}_{S}\boldsymbol{R}$, corresponding to an $180°$ rotation about the X axis. This operation yields

$$
\begin{aligned}
{}^{B}_{I}\boldsymbol{R} &= {}^{B}_{S}\boldsymbol{R}\,{}^{S}_{I}\boldsymbol{R} = \boldsymbol{R}_x(\pi)\,{}^{S}_{I}\boldsymbol{R} \\
&= \begin{bmatrix} 1 & 0 & 0 \\ 0 & c_\pi & -s_\pi \\ 0 & s_\pi & c_\pi \end{bmatrix} \begin{bmatrix} c_\psi c_\theta & c_\theta s_\psi & -s_\theta \\ c_\psi s_\phi s_\theta - c_\phi s_\psi & c_\phi c_\psi + s_\phi s_\psi s_\theta & c_\theta s_\phi \\ s_\phi s_\psi + c_\phi c_\psi s_\theta & c_\phi s_\psi s_\theta - c_\psi s_\phi & c_\phi c_\theta \end{bmatrix} \\
&= \begin{bmatrix} c_\psi c_\theta & c_\theta s_\psi & -s_\theta \\ c_\phi s_\psi - c_\psi s_\phi s_\theta & -c_\phi c_\psi - s_\phi s_\psi s_\theta & -c_\theta s_\phi \\ -s_\phi s_\psi - c_\phi c_\psi s_\theta & c_\psi s_\phi - c_\phi s_\psi s_\theta & -c_\phi c_\theta \end{bmatrix}
\end{aligned} \tag{3.5}
$$

The roll and pitch angles can then be obtained, directly through the third column of ${}^{B}_{I}\boldsymbol{R}$, according to

$$
\begin{cases} \tan(\phi) = \frac{{}^{B}_{I}r_{23}}{{}^{B}_{I}r_{33}} \\ -\sin(\theta) = {}^{B}_{I}r_{13} \end{cases} \Leftrightarrow \begin{cases} \phi = \tan^{-1}\left(\frac{{}^{B}_{I}r_{23}}{{}^{B}_{I}r_{33}}\right) \\ \theta = \sin^{-1}(-{}^{B}_{I}r_{13}) \end{cases} \tag{3.6}
$$

The computation of the roll angle ϕ, resorting to the inverse tangent function, returns values in the $[-\pi/2, \pi/2]$ interval, disregarding possible quadrant information. Therefore, the implementation of this calculation needs to account for the sign of ${}^{B}_{I}r_{23}$ and ${}^{B}_{I}r_{33}$, through the four-quadrant inverse tangent function, denoted by `atan2`. On the other hand, in Equation (3.6) the pitch angle θ is found with the inverse sine function, whose result lies within the aforementioned interval. However, since the vehicle never actually exceeds that range in pitch, and knowing that $\sin(-\theta) = -\sin(\theta)$, the obtained values are unique and do not need additional corrections.

Although the roll and pitch angles are no longer being determined based on the LiDAR, they can be combined with data from that sensor, to provide a better estimate of the yaw angle ψ. This is useful since, without the LiDAR measurements, the yaw is being acquired as a combination of the measurements from gyroscopes and magnetometers, the latter being susceptible to electromagnetic disturbances and having an unknown drift, in general.

Taking on an approach based on the angle between the edges rather than their length, the idea behind the projection introduced in Equation (3.2) is revisited. Starting with the roll and pitch angles from the IMU, the edges ${}^{B}\boldsymbol{Q}$ can be projected to the XY plane of $\{H\}$, leaving the yaw angle ψ as the only unknown. Given the fact that, no matter what roll and pitch the vehicle has in a given instant, the projection of the data points to $\{H\}$ will always result in two orthogonal lines, then the edge identification step can be reformulated. Following this perspective, the data points can be rotated to $\{H\}$ before the constrained least squares problem instead of after, through

$$^{H}\boldsymbol{M} = \boldsymbol{\Pi}_{\boldsymbol{e}_3}\, {}^{H}_{B}\boldsymbol{R}\, {}^{B}\boldsymbol{M} \tag{3.7}$$

Having the data points matching two orthogonal lines, Equation (2.4) can be rewritten as

$$\begin{aligned}
&\underset{c_1,c_2,n_x,n_y}{\text{Minimize}} \quad \|\boldsymbol{e}\|^2 = \sum_{i=1}^{N_1+N_2} e_i^2 \\
&\text{Subject to} \quad \begin{bmatrix} 1 & 0 & x_{1,1} & y_{1,1} \\ \vdots & \vdots & \vdots & \vdots \\ 1 & 0 & x_{1,N_1} & y_{1,N_1} \\ 0 & 1 & y_{2,1} & -x_{2,1} \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 1 & y_{2,N_2} & -x_{2,N_2} \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ n_x \\ n_y \end{bmatrix} = \begin{bmatrix} e_1 \\ \vdots \\ e_{N_1+N_2} \end{bmatrix}, \\
&\qquad\qquad n_x^2 + n_y^2 = 1
\end{aligned} \tag{3.8}$$

to consider the minimization of two straight orthogonal lines simultaneously. In this case, the QR decomposition of matrix $\boldsymbol{A}$ leads to the reduced system

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{e} \Leftrightarrow \boldsymbol{Q}^T\boldsymbol{A}\boldsymbol{x} = \boldsymbol{Q}^T\boldsymbol{e} \Leftrightarrow \begin{bmatrix} r_{11} & r_{12} & r_{13} & r_{14} \\ 0 & r_{22} & r_{23} & r_{24} \\ 0 & 0 & r_{33} & r_{34} \\ 0 & 0 & 0 & r_{44} \\ 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ n_x \\ n_y \end{bmatrix} = \boldsymbol{y} \tag{3.9}$$

Taking into account the nonlinear constraint, updated for this situation, the problem assumes the form

$$\begin{aligned}
&\underset{c_1,c_2,n_x,n_y}{\text{Minimize}} \quad \begin{bmatrix} r_{33} & r_{34} \\ 0 & r_{44} \end{bmatrix} \begin{bmatrix} n_x \\ n_y \end{bmatrix} \approx \begin{bmatrix} 0 \\ 0 \end{bmatrix} \\
&\text{Subject to} \quad n_x^2 + n_y^2 = 1
\end{aligned} \tag{3.10}$$

With this approach, the least squares estimation becomes independent of the vehicle's roll and pitch angles and avoids the uncertainty of the angle between the edges, which corresponds to $90°$. This means the error in the estimated heading to the structure can be reduced, since the data points of both edges now contribute to an unified objective, with a joint error, rather than two separated goals with unrelatable errors. The idea is no longer to find the angle of each individual edge, but instead the angle of part of a known rectangular section, which carries great benefit in the two edges case. However, the accuracy of the roll and pitch estimates, obtained from the IMU, will also have an impact on the accuracy the yaw estimate. With the edges $^{H}\boldsymbol{Q}$, the rotation matrix from $\{H\}$ to $\{I\}$ given by

$$
\begin{aligned}
{}^{I}_{H}\boldsymbol{R} &= \boldsymbol{R}_z = \boldsymbol{R}_z(\psi) \\
&= \begin{bmatrix} c_\psi & -s_\psi & 0 \\ s_\psi & c_\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}
\end{aligned} \tag{3.11}
$$

is the last step towards having the edges ${}^{I}\boldsymbol{Q}$. Performing the rotation of an edge from $\{H\}$ to $\{I\}$ only depends on the yaw angle ψ, whose estimate can be obtained through

$$
{}^{I}\boldsymbol{Q}_i = {}^{I}_{H}\boldsymbol{R}\,{}^{H}\boldsymbol{Q}_i \Leftrightarrow \hat{\psi} = \tan^{-1}\left(-\frac{Q_{i_y}}{Q_{i_x}}\right), \text{ for } i \in \{1, 2\} \tag{3.12}
$$

It should be noted that the yaw usually has its reference at the magnetic North, so applying this method implies setting an artificial reference. In this case, it was chosen that the yaw angle would be set relative to the edge of the structure initially at the left. Therefore, the initial yaw angle ψ should be obtained using ${}^{H}\boldsymbol{Q}_1$, which corresponds to ${}^{I}\boldsymbol{Q}_1$, that in turn equals $\boldsymbol{l}_1$ in the starting leveled flight condition.

Furthermore, when navigating around the structure, the LiDAR captures a sequence of time steps corresponding to different views, forming a progression. Most of the times, there are effectively two edges, but in the transition stage, the accuracy of the data points and the consequent identification can originate misleading situations. As mentioned in Section 2.3.2.1, there are several reasons why there could be an intermittent edge, depending on the existence of outliers and the fulfillment of the edge splitting criteria. This leads to degenerated situations, where there is only one edge and it has a tilt from the perfect 90° angle, with respect to the vehicle. In practice, the case at hand ends up being one of the eight different possibilities from Figure 3.1, therefore Algorithm 4 was developed to detect these cases and ensure a continuous transition scheme. This diagram was built as a basis for the algorithm and with the objective of representing all the possible alternatives, for the sake of completeness, considering the combinations of facing one or two edges in the previous and current steps, depending on the direction of rotation as well.

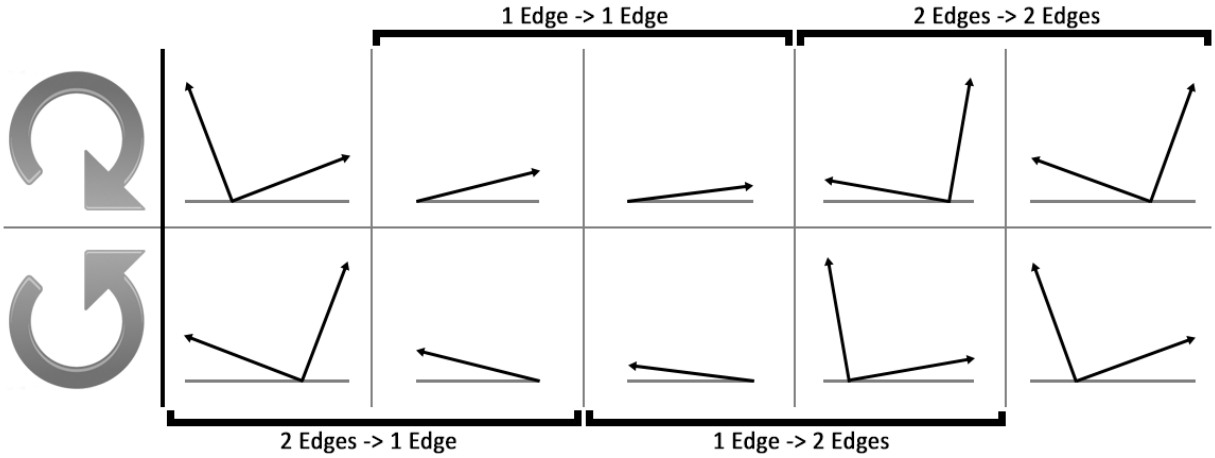


Figure 3.1: Progression of the view around the structure with all the transition cases.

Algorithm 4 Yaw Continuity

procedure YAWCONTINUITY($^H\boldsymbol{Q}, {}^H\boldsymbol{Q}_{new}, \hat{\psi}$)
  if $\hat{\psi} = 0$ **then**
    $\hat{\psi}_{new} = -\operatorname{atan2}({}^HQ_{1_{y_{new}}}, {}^HQ_{1_{x_{new}}})$
  else
    $\hat{\psi}_1 = -\operatorname{atan2}({}^HQ_{1_y}, {}^HQ_{1_x})$
    $\hat{\psi}_{1_{new}} = -\operatorname{atan2}({}^HQ_{1_{y_{new}}}, {}^HQ_{1_{x_{new}}})$
    if ${}^H\boldsymbol{Q}_2 \neq 0$ **and** ${}^H\boldsymbol{Q}_{2_{new}} \neq 0$ **then**
      $d\psi = \hat{\psi}_{1_{new}} - \hat{\psi}_1$
      $\hat{\psi}_{new} = \mathrm{YawShiftX2X}(d\psi, \hat{\psi}, 45°)$
    else if ${}^H\boldsymbol{Q}_2 = 0$ **and** ${}^H\boldsymbol{Q}_{2_{new}} = 0$ **then**
      $d\psi = \hat{\psi}_{1_{new}} - \hat{\psi}_1$
      $\hat{\psi}_{new} = \mathrm{YawShiftX2X}(d\psi, \hat{\psi}, 90°)$
    else if ${}^H\boldsymbol{Q}_2 \neq 0$ **and** ${}^H\boldsymbol{Q}_{2_{new}} = 0$ **then**
      $\hat{\psi}_2 = -\operatorname{atan2}({}^HQ_{2_y}, {}^HQ_{2_x})$
      if $\|\hat{\psi}_{1_{new}} - \hat{\psi}_1\| < \|\hat{\psi}_{1_{new}} - \hat{\psi}_2\|$ **then**
        $d\psi = \hat{\psi}_{1_{new}} - \hat{\psi}_1$
        $\hat{\psi}_{new} = \mathrm{YawShiftX2YCW}(d\psi, \hat{\psi}, \hat{\psi}_2, \hat{\psi}_{1_{new}})$
      else
        $d\psi = \hat{\psi}_{1_{new}} - \hat{\psi}_2$
        $\hat{\psi}_{new} = \mathrm{YawShiftX2YCCW}(d\psi, \hat{\psi}, \hat{\psi}_1, \hat{\psi}_{1_{new}})$
      end if
    else if ${}^H\boldsymbol{Q}_2 = 0$ **and** ${}^H\boldsymbol{Q}_{2_{new}} \neq 0$ **then**
      $\hat{\psi}_{2_{new}} = -\operatorname{atan2}({}^HQ_{2_{y_{new}}}, {}^HQ_{2_{x_{new}}})$
      if $\|\hat{\psi}_{1_{new}} - \hat{\psi}_1\| < \|\hat{\psi}_{2_{new}} - \hat{\psi}_1\|$ **then**
        $d\psi = \hat{\psi}_{1_{new}} - \hat{\psi}_1$
        $\hat{\psi}_{new} = \mathrm{YawShiftX2YCCW}(d\psi, \hat{\psi}, \hat{\psi}_1, \hat{\psi}_{2_{new}})$
      else
        $d\psi = \hat{\psi}_{2_{new}} - \hat{\psi}_1$
        $\hat{\psi}_{new} = \mathrm{YawShiftX2YCW}(d\psi, \hat{\psi}, \hat{\psi}_1, \hat{\psi}_{1_{new}})$
      end if
    end if
  end if
end procedure

The algorithm receives the previous yaw angle estimate $\hat{\psi}$ and the previous and current edges, ${}^H\boldsymbol{Q}$ and ${}^H\boldsymbol{Q}_{new}$ respectively, where the second vector of these matrices can be null in the one edge case. The first time it runs, there is no history, so the yaw angle estimate $\hat{\psi}$ is defined relative to the left edge and it becomes the reference. The main idea is to take the previous yaw angle estimate $\hat{\psi}$ and add the yaw angle variation $d\psi$ perceived in the current step. When there is the same number of edges in the previous and current steps, there is a straightforward relationship between the edges and they can be easily associated. In those cases, the increment is simply the difference in the yaw angle provided by a chosen edge.

However, if the vehicle is moving fast enough, it is possible that the one edge case may be missed entirely, so there are two consecutive two edge cases that actually have a hidden $90°$ shift regarding the yaw angle obtained. Therefore, if there is an increment higher than $45°$, the necessary correction is applied. A similar situation occurs when the vehicle is close to facing one of the structure's edges head on. The ideal transition, considering that there are multiple instances with one edge, would be to have the edge tilted one way in the first moment, followed by a perfect $90°$ angle in the next and then being tilted in the other direction, completing the rotation. But again, because the probability of that middle step actually happening is very slim, there will most likely be a discontinuity around the $180°$. This is

due to the abrupt shift of the quadrant to which the edge belongs, which can be seen by remembering the trigonometric circle and knowing that the edge always points outwards starting from the origin. This means that a correction of π must be applied, which also depends on the direction of the rotation. Both of these cases are dealt with using Algorithm 5, in which there is a transition with the same number of edges before and after, the difference being the value of the condition and the correction that needs to be applied.

When there is a transition between one and two edges or vice-versa, there are two situations that can occur. The identification can wrongfully detect only one edge in an isolated step, which is exactly the case depicted in the first transition of Figure 3.1, or a turn can actually be starting, as it happens in the third transition. In either case, the task now is to find out which pair of edges leads to a smaller increment $d\psi$, meaning they probably refer to the same edge in different steps, and take that value as the yaw angle variation. This is it if the algorithm is recovering from a misidentification, where it actually needs to find the closer edge. When a turn is actually being made, there will be an shift higher than $45°$, so the increment becomes $\overline{\Delta\psi} = \pm(\pi \pm \Delta\psi^*)$, where the sign depends on the direction of rotation and $\Delta\psi^*$ is now obtained with the other edge combination, since there is a quadrant shift of the relevant edge. Algorithms 6 and 7 were developed for exceptions in which the number of edges before is different from after, to implement the required verifications, depending on the direction of rotation.

Algorithm 5 Yaw Shift X2X

procedure YAWSHIFTX2X($d\psi, \hat{\psi}, d\psi_{max}$)
  if $\|d\psi\| < d\psi_{max}$ **then**
    $\hat{\psi}_{new} = \hat{\psi} + d\psi$
  else if $d\psi > d\psi_{max}$ **then**
    $\hat{\psi}_{new} = \hat{\psi} + d\psi - 2 \cdot d\psi_{max}$
  else if $d\psi < -d\psi_{max}$ **then**
    $\hat{\psi}_{new} = \hat{\psi} + d\psi + 2 \cdot d\psi_{max}$
  end if
end procedure

Algorithm 6 Yaw Shift X2Y CW

procedure YAWSHIFTX2YCW($d\psi, \hat{\psi}, \hat{\psi}_i, \hat{\psi}_f$)
  if $\|d\psi\| < 45°$ **then**
    $\hat{\psi}_{new} = \hat{\psi} + d\psi$
  else
    $\hat{\psi}_{new} = \hat{\psi} - (180° - (\hat{\psi}_f - \hat{\psi}_i))$
  end if
end procedure

Algorithm 7 Yaw Shift X2Y CCW

procedure YAWSHIFTX2YCCW($d\psi, \hat{\psi}, \hat{\psi}_i, \hat{\psi}_f$)
  if $\|d\psi\| < 45°$ **then**
    $\hat{\psi}_{new} = \hat{\psi} + d\psi$
  else
    $\hat{\psi}_{new} = \hat{\psi} + (180° + (\hat{\psi}_f - \hat{\psi}_i))$
  end if
end procedure

3.2 Full Motion

Considering the complete motion capabilities of a vehicle in attitude involves procedures which can be more expensive, at a computational level.

First, taking on a problem known in spacecraft attitude determination for many years, Section 3.2.1 describes what this robust solution consists of and how it can be used for this purpose, with the information available from the LiDAR and specific components of an IMU. This formulation involves the computation of the optimal rotation matrix from $\{I\}$ to $\{B\}$ in its closed-form solution, establishing the relationship between the observations in their main reference frames.

The next is designed from scratch in Section 3.2.2, as an attempt to build a customized tool for this application, where the stability principles and control theory behind it are explained as well. It is also proven the convergence of this method, which is actually an observer fusing LiDAR data with raw information from the IMU, based on the boundedness of the bias it incurs throughout time. Additionally, its most important features and limitations are addressed.

The rotation matrix acquired from the IMU is a product of the combination of three types of sensors, namely accelerometers, gyroscopes, and magnetometers. The data coming from accelerometers has a high precision, but by depending on gravity, it cannot be used to describe the motion around the Z axis. Using the information from the gyroscopes complements that description, however it implies the integration of angular velocity over time, which is a process that accumulates errors and generates a drift with growing significance. The magnetometers are supposed to provide an additional correction, in the vertical axis, hereby completing the 3D attitude solution. Different combinations of these sensors can be made, to obtain alternative descriptions of the vehicle's attitude.

3.2.1 Wahba's Problem Application

The challenge commonly known as Wahba's problem refers to the determination of the three-axis attitude of an aircraft, by estimating the proper orthogonal matrix ${}^B_I\hat{\boldsymbol{R}}$ that minimizes the least squares loss function

$$L({}^B_I\hat{\boldsymbol{R}}) = \frac{1}{2}\sum_{i=1}^{n_{obs}} \boldsymbol{w}_i \| {}^B\boldsymbol{O}_i - {}^B_I\hat{\boldsymbol{R}}{}^I\boldsymbol{O}_i \|^2 = \sum_{i=1}^{n_{obs}} \boldsymbol{w}_i - \sum_{i=1}^{n_{obs}} \boldsymbol{w}_i {}^B\boldsymbol{O}_i^T {}^B_I\hat{\boldsymbol{R}}{}^I\boldsymbol{O}_i \tag{3.13}$$

where ${}^I\boldsymbol{O}$ is the set of observations represented in $\{I\}$, ${}^B\boldsymbol{O}$ is the same set but represented in $\{B\}$, $\boldsymbol{w}$ is a vector with positive weights associated with each individual observation, and n_{obs} is the total number of observations [19]. The loss function is based on the idea that, if the measurements are free of errors and the true rotation matrix ${}^B_I\boldsymbol{R}$ is the same for all measurements, then ${}^B\boldsymbol{O}_i = {}^B_I\boldsymbol{R}{}^I\boldsymbol{O}_i, \forall i$.

Having the description of the edges in both $\{I\}$ and $\{B\}$, they can be directly introduced in the loss function. However, despite the fact that most of the times there are two edges that fully define the attitude of the vehicle, when only one edge is being seen, an ambiguity arises due to an additional Degree Of Freedom (DOF). The accelerometer within the IMU can be used to provide an extra piece of information and obtain an unequivocal attitude, assuming that the vehicle's acceleration is negligible

relative to the gravitational acceleration. With that in mind, the input of the minimization will correspond to the observations matrix ${}^{*}\boldsymbol{O} = \begin{bmatrix} {}^{*}\boldsymbol{Q} & {}^{*}\boldsymbol{a} \end{bmatrix} = \begin{bmatrix} {}^{*}\boldsymbol{Q}_1 & {}^{*}\boldsymbol{Q}_2 & {}^{*}\boldsymbol{a} \end{bmatrix}$, where the additional observation is the normalized acceleration vector and $*$ represents each of the two reference frames.

Considering a generic matrix $\boldsymbol{H} = \sum_{i=1}^{n_{obs}} \boldsymbol{w}_i {}^{B}\boldsymbol{O}_i {}^{I}\boldsymbol{O}_i^T$, the last term of Equation (3.13) becomes $-\operatorname{tr}({}^{B}_{I}\hat{\boldsymbol{R}}\boldsymbol{H}^T)$. In this expression, tr and the superscript T refer to the trace and transpose of a matrix respectively. This transformation is possible due to the invariance of the trace of a product of matrices under a cyclic permutation of the factors in the product. The SVD of matrix $\boldsymbol{H}$ is given by

$$\boldsymbol{H} = \boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^T = \boldsymbol{U}\operatorname{diag}(s_1, s_2, s_3)\boldsymbol{V}^T \tag{3.14}$$

where $\boldsymbol{U}$ and $\boldsymbol{V}$ are orthogonal matrices and the diagonal elements of $\boldsymbol{S}$ respect $s_1 \geq s_2 \geq s_3$. The trace of ${}^{B}_{I}\hat{\boldsymbol{R}}\boldsymbol{H}^T$ can now be written in the form

$$\operatorname{tr}({}^{B}_{I}\hat{\boldsymbol{R}}\boldsymbol{H}^T) = \operatorname{tr}({}^{B}_{I}\hat{\boldsymbol{R}}\boldsymbol{V}\operatorname{diag}(s_1, s_2, s_3)\boldsymbol{U}^T) = \operatorname{tr}(\boldsymbol{U}^T {}^{B}_{I}\hat{\boldsymbol{R}}\boldsymbol{V}\operatorname{diag}(s_1, s_2, s_3)) \tag{3.15}$$

In order to minimize the aforementioned loss function, the trace of ${}^{B}_{I}\hat{\boldsymbol{R}}\boldsymbol{H}^T$ must be maximized, according to the minus sign associated with that term. Taking into account the constraint $\det({}^{B}_{I}\hat{\boldsymbol{R}}) = 1$, the closed-form solution of the rotation matrix estimate ${}^{B}_{I}\hat{\boldsymbol{R}}$ is found through

$$\boldsymbol{U}^T {}^{B}_{I}\hat{\boldsymbol{R}}\boldsymbol{V} = \operatorname{diag}(1, 1, \det(\boldsymbol{U})\det(\boldsymbol{V})) \Leftrightarrow {}^{B}_{I}\hat{\boldsymbol{R}} = \boldsymbol{U}\operatorname{diag}(1, 1, \det(\boldsymbol{U})\det(\boldsymbol{V}))\boldsymbol{V}^T \tag{3.16}$$

3.2.2 Rotation Matrix Observer

The magnetometers are very sensible to electromagnetic radiation and suffer from interferences that cause disruptions, so replacing them with the LiDAR can lead to a more accurate attitude determination tool. The LiDAR has its highest accuracy when it is used to estimate the movement about the Z axis, whereas the information it yields for the remaining axes is more susceptible to measurement errors. Thus, the accelerometer is the perfect addition to minimize those uncertainties.

As shown in [20], given the definition of angular velocity $\boldsymbol{\omega}$, the kinematics of the rotation matrix ${}^{B}_{I}\boldsymbol{R}$ are given by

$${}^{B}_{I}\dot{\boldsymbol{R}} = -S(\boldsymbol{\omega}){}^{B}_{I}\boldsymbol{R} \tag{3.17}$$

The same system is replicated for the estimator of the rotation matrix ${}^{B}_{I}\hat{\boldsymbol{R}}$, such that

$${}^{B}_{I}\dot{\hat{\boldsymbol{R}}} = -S(\hat{\boldsymbol{\omega}}){}^{B}_{I}\hat{\boldsymbol{R}} \tag{3.18}$$

where $\hat{\omega}$ is yet to be determined. Assuming that $\boldsymbol{\omega}(t)$ is a known input, Equations (3.17) and (3.18) are in the form of a Linear Time-Varying (LTV) system, described by $\dot{\boldsymbol{x}}(t) = \boldsymbol{A}(t)\boldsymbol{x}(t)$, where $\boldsymbol{A}(t)$ corresponds to $-S(\boldsymbol{\omega}(t))$, the skew-symmetric kinematics matrix computed from the $\boldsymbol{\omega}(t)$ vector, whose dependency on time will be omitted. It should be noted that these kinematics are built based solely on the integration

of the angular velocity information from the IMU. Moreover, a variable $\tilde{\boldsymbol{R}} = {}^{B}_{I}\boldsymbol{R}^{B}_{I}\hat{\boldsymbol{R}}^{T}$ can be defined to represent the error between the true rotation matrix and its estimate.

Theorem 3.1. *Considering the system in Equation* (3.17) *and its estimator in Equation* (3.18)*, if* $\hat{\boldsymbol{\omega}}$ *is defined as*

$$\hat{\boldsymbol{\omega}} = \boldsymbol{\omega} + K_{obs}\sum_{i=1}^{3} S({}^{B}\boldsymbol{O}_i)\tilde{\boldsymbol{R}}^{T}{}^{B}\boldsymbol{O}_i \tag{3.19}$$

where $K_{obs} > 0$ *is a tunable gain, then the equilibrium point* $\tilde{\boldsymbol{R}} = \boldsymbol{I}$*, from the error system, is almost globally asymptotically stable.*

Proof. The Lyapunov stability theory is based on the analysis of a function, generally associated with the energy of the system at hand. In this case, an expression related with the error in the rotation matrix such as

$$V(\tilde{\boldsymbol{R}}) = \operatorname{tr}(\boldsymbol{I} - \tilde{\boldsymbol{R}}) \tag{3.20}$$

where $\boldsymbol{I}$ is the identity matrix, can be considered a reasonable candidate Lyapunov function to evaluate the stability of the system. In order to guarantee the stability requirements for an equilibrium point $\tilde{\boldsymbol{R}} = \boldsymbol{I}$, the Lyapunov function needs to be positive definite, i.e. $V(\boldsymbol{I}) = 0$ and $V(\tilde{\boldsymbol{R}}) > 0 \,\forall\, \tilde{\boldsymbol{R}} \in \mathcal{SO}(3) \setminus \{\boldsymbol{I}\}$, and its derivative must be negative semi-definite, meaning that $\dot{V}(\boldsymbol{I}) = 0$ and $\dot{V}(\tilde{\boldsymbol{R}}) \leq 0 \,\forall\, \tilde{\boldsymbol{R}} \in \mathcal{SO}(3) \setminus \{\boldsymbol{I}\}$, and negative definite to reach asymptotic stability, verifying $\dot{V}(\boldsymbol{I}) = 0$ and $\dot{V}(\tilde{\boldsymbol{R}}) < 0 \,\forall\, \tilde{\boldsymbol{R}} \in \mathcal{SO}(3) \setminus \{\boldsymbol{I}\}$. The derivative of the Lyapunov function is

$$\dot{V}(\tilde{\boldsymbol{R}}) = -\operatorname{tr}\left[S(\boldsymbol{\omega})\tilde{\boldsymbol{R}} - \tilde{\boldsymbol{R}}S(\hat{\boldsymbol{\omega}})\right] = \operatorname{tr}\left[S(\boldsymbol{\omega} - \hat{\boldsymbol{\omega}})\tilde{\boldsymbol{R}}\right] \tag{3.21}$$

after using the distributive and associative properties of the trace of a matrix and skew-symmetric matrices respectively. From that expression, it can be shown that

$$\hat{\boldsymbol{\omega}} = \boldsymbol{\omega} + K_{obs}\sum_{i=1}^{3} S({}^{B}\boldsymbol{O}_i)\tilde{\boldsymbol{R}}^{T}{}^{B}\boldsymbol{O}_i = \boldsymbol{\omega} + K_{obs}\sum_{i=1}^{3} S({}^{B}\boldsymbol{O}_i){}^{B}_{I}\hat{\boldsymbol{R}}^{I}\boldsymbol{O}_i \tag{3.22}$$

The latter expression for $\hat{\boldsymbol{\omega}}$ was rearranged specially for implementation purposes, whereas the following reasonings will build upon the former. Making use of the skew-symmetric matrix inverse function S^{-1}, not to be confused with the inverse of a skew-symmetric matrix, the first description of $\hat{\boldsymbol{\omega}}$ in Equation (3.22) can be rewritten as

$$\hat{\boldsymbol{\omega}} = \boldsymbol{\omega} + K_{obs}\sum_{i=1}^{3} S^{-1}\left(\tilde{\boldsymbol{R}}^{T}{}^{B}\boldsymbol{O}_i{}^{B}\boldsymbol{O}_i^{T} - {}^{B}\boldsymbol{O}_i{}^{B}\boldsymbol{O}_i^{T}\tilde{\boldsymbol{R}}\right) \tag{3.23}$$

knowing that, for any $\boldsymbol{\nu} \in \mathbb{R}^3$, the $S(\boldsymbol{\nu_1})\boldsymbol{\nu_2} = S^{-1}(\boldsymbol{\nu_2}\boldsymbol{\nu_1}^{T} - \boldsymbol{\nu_1}\boldsymbol{\nu_2}^{T})$ property of skew-symmetric matrices holds. Using Equation (3.23), the derivative of the Lyapunov function becomes

$$\begin{aligned}\dot{V}(\tilde{\boldsymbol{R}}) &= \operatorname{tr}\left[S\left(-K_{obs}\sum_{i=1}^{3} S^{-1}\left(\tilde{\boldsymbol{R}}^{T}\,{}^{B}\boldsymbol{O}_i\,{}^{B}\boldsymbol{O}_i^{T} - {}^{B}\boldsymbol{O}_i\,{}^{B}\boldsymbol{O}_i^{T}\tilde{\boldsymbol{R}}\right)\tilde{\boldsymbol{R}}\right)\right] \\ &= -K_{obs}\sum_{i=1}^{3} {}^{B}\boldsymbol{O}_i^{T}\left(\boldsymbol{I} - \tilde{\boldsymbol{R}}^{2}\right){}^{B}\boldsymbol{O}_i\end{aligned} \tag{3.24}$$

which fulfills the stability requirements in the domain $\tilde{\boldsymbol{R}} \in \mathcal{SO}(3) : V(\tilde{\boldsymbol{R}}) < 4 - \epsilon \,\forall\, 0 < \epsilon < 4$. The cyclic permutation property of the trace of a matrix was one of the tools used during the simplification. In fact, it can be shown that the error system is almost globally asymptotically stable, i.e. it is stable and attractive except for a zero measure set of initial conditions [21]. □

The previously developed work, for treating the LiDAR data, yielded an edges matrix in $\{B\}$, which then depended on what the sensor was encountering at a given time. When there are two edges, both its vectors are filled, but when only one can be seen, its information is stored in the first vector, without indication about which edge it refers to. In order to tackle this association issue and make it possible to compute $\hat{\boldsymbol{\omega}}$, and therefore ${}^{B}_{I}\hat{\boldsymbol{R}}$, the edges in $\{I\}$ being faced at the start have to be set and then updated over time. When there is the same number of edges in the previous and current measurements, they stay the same, but with transitions between one and two edges and vice-versa, the closest edge needs to be found across measurements, so that the association can be made considering continuity and the direction of rotation.

The integration of the measurements from the gyroscopes suffers from drift over time, with two different origins. On one hand, there is the apparent drift, due to the Earth's rotation, and on the other, the real drift, due to bearing friction and gimbal and mass unbalance. Additionally, carrying them over the Earth's surface causes a similar effect that depends only on latitude, named transport wander. While this second factor might not be very meaningful in the scenarios where this application will take place, given the approximately constant latitude, there is still a drift whose compensation in the IMU, by incorporating data from other sensors, may not very effective. In the end, the estimate will converge to the rotation matrix obtained from a biased angular velocity, which means the bias will still be present in ${}^{B}_{I}\hat{\boldsymbol{R}}$ and there will always be an error between the true and the estimated rotation matrix. This error will add up to the effect that the magnetic field's distortion causes on the data fusion with the magnetometers, within the IMU.

The Rodrigues' Rotation Formula is an efficient algorithm for rotating a unitary vector $\boldsymbol{\eta}$ in space, given an axis and angle of rotation ξ. In this case, it can be used to describe $\tilde{\boldsymbol{R}}$ through

$$\tilde{\boldsymbol{R}} = \boldsymbol{I} + \sin(\xi)S(\boldsymbol{\eta}) + (1 - \cos(\xi))S(\boldsymbol{\eta})^2 \tag{3.25}$$

Theorem 3.2. *Considering the estimator in Equation* (3.18)*, if there is a biased angular velocity* $\overline{\boldsymbol{\omega}} = \boldsymbol{\omega} + \boldsymbol{\delta\omega}$ *such that* $\hat{\boldsymbol{\omega}}$ *becomes*

$$\hat{\boldsymbol{\omega}} = \boldsymbol{\omega} + \boldsymbol{\delta\omega} + K_{obs}\sum_{i=1}^{3} S({}^{B}\boldsymbol{O}_i)\tilde{\boldsymbol{R}}^{T}\,{}^{B}\boldsymbol{O}_i \tag{3.26}$$

then the estimation error is ultimately bounded.

Proof. In order to determine what is the actual effect of the bias, in the behavior of the convergence process, the biased angular velocity is considered in the Lyapunov function, leading to

$$\hat{\boldsymbol{\omega}} = \boldsymbol{\omega} + \boldsymbol{\delta\omega} + K_{obs} \sum_{i=1}^{3} S^{-1} \left(\tilde{\boldsymbol{R}}^T {}^B\boldsymbol{O}_i {}^B\boldsymbol{O}_i^T - {}^B\boldsymbol{O}_i {}^B\boldsymbol{O}_i^T \tilde{\boldsymbol{R}} \right) \tag{3.27}$$

With this extra term in Equation (3.27), the simplification presented in Equation (3.24) needs to be revisited. Replacing the vectorial sums with their matrix notation, the derivative of the Lyapunov function, taking into account the bias, is

$$\begin{aligned} \dot{V}(\tilde{\boldsymbol{R}}) &= \operatorname{tr}\left[S\left(-\boldsymbol{\delta\omega} - K_{obs} S^{-1} \left(\tilde{\boldsymbol{R}}^T {}^B\boldsymbol{O} {}^B\boldsymbol{O}^T - {}^B\boldsymbol{O} {}^B\boldsymbol{O}^T \tilde{\boldsymbol{R}} \right) \tilde{\boldsymbol{R}} \right) \right] \\ &= \operatorname{tr}\left[S(\boldsymbol{\delta\omega}) \left(\boldsymbol{I} - \tilde{\boldsymbol{R}} \right) - K_{obs} {}^B\boldsymbol{O} {}^B\boldsymbol{O}^T \left(\boldsymbol{I} - \tilde{\boldsymbol{R}}^2 \right) \right] \end{aligned} \tag{3.28}$$

where the fact that $\operatorname{tr}(S(\boldsymbol{\nu})\boldsymbol{I}) = 0$ is used to add a term to the expression. Replacing the Rodrigues' Rotation Formula into the the derivative of the Lyapunov function produces

$$\begin{aligned} \dot{V}(\tilde{\boldsymbol{R}}) &= \operatorname{tr}\left[\left(\boldsymbol{I} - \tilde{\boldsymbol{R}} \right) \left(S(\boldsymbol{\delta\omega}) - K_{obs} {}^B\boldsymbol{O} {}^B\boldsymbol{O}^T \left(\boldsymbol{I} + \tilde{\boldsymbol{R}} \right) \right) \right] \\ &= \operatorname{tr}\left[- \left(\sin(\xi) S(\boldsymbol{\eta}) + (1 - \cos(\xi)) S(\boldsymbol{\eta})^2 \right) \right. \\ &\qquad \left. \left(S(\boldsymbol{\delta\omega}) - 2K_{obs} {}^B\boldsymbol{O} {}^B\boldsymbol{O}^T - K_{obs} \sin(\xi) {}^B\boldsymbol{O} {}^B\boldsymbol{O}^T S(\boldsymbol{\eta}) - (1 - \cos(\xi)) K_{obs} {}^B\boldsymbol{O} {}^B\boldsymbol{O}^T S(\boldsymbol{\eta})^2 \right) \right] \end{aligned} \tag{3.29}$$

Developing Equation (3.29) through the application of the distributive property of multiplication, the property of skew-symmetric matrices $S(\boldsymbol{\nu})^3 = -(\boldsymbol{\nu}^T\boldsymbol{\nu}) S(\boldsymbol{\nu}) = -S(\boldsymbol{\nu})$ for $\|\boldsymbol{\nu}\| = 1$, the trigonometric identity $\cos^2(\alpha) - \sin^2(\alpha) = 1 - 2\sin^2(\alpha)$, and once again the fact that $\operatorname{tr}[S(\boldsymbol{\nu})] = 0$ yields

$$\begin{aligned} \dot{V}(\tilde{\boldsymbol{R}}) = \operatorname{tr}\Big[& -\sin(\xi) S(\boldsymbol{\eta}) S(\boldsymbol{\delta\omega}) - K_{obs} \sin(\xi)(1 + \cos(\xi)) S(\boldsymbol{\eta}) {}^B\boldsymbol{O} {}^B\boldsymbol{O}^T \\ & -2K_{obs} \sin^2(\xi) S(\boldsymbol{\eta})^2 {}^B\boldsymbol{O} {}^B\boldsymbol{O}^T + (1 - \cos(\xi)) S(\boldsymbol{\eta})^2 S(\boldsymbol{\delta\omega}) \Big] \end{aligned} \tag{3.30}$$

A few more properties must be used, involving skew-symmetric matrices and the trace of a matrix. Knowing that $S(\boldsymbol{\nu})^2 = \boldsymbol{\nu}\boldsymbol{\nu}^T - (\boldsymbol{\nu}^T\boldsymbol{\nu})\boldsymbol{I}$ is useful to manipulate the third term, by introducing an additional matrix, according to

$$\begin{aligned} \operatorname{tr}\left[S(\boldsymbol{\nu_1})^2 \boldsymbol{\nu_2} \boldsymbol{\nu_2}^T \right] &= \operatorname{tr}(\boldsymbol{\nu_1} \boldsymbol{\nu_1}^T \boldsymbol{\nu_2} \boldsymbol{\nu_2}^T - \boldsymbol{\nu_1}^T \boldsymbol{\nu_1} \boldsymbol{\nu_2} \boldsymbol{\nu_2}^T) \\ &= \boldsymbol{\nu_1}^T \boldsymbol{\nu_2} \boldsymbol{\nu_2}^T \boldsymbol{\nu_1} - \operatorname{tr}(\boldsymbol{\nu_2} \boldsymbol{\nu_2}^T) \\ &= \boldsymbol{\nu_1}^T \left[\boldsymbol{\nu_2} \boldsymbol{\nu_2}^T - \operatorname{tr}(\boldsymbol{\nu_2} \boldsymbol{\nu_2}^T) \boldsymbol{I} \right] \boldsymbol{\nu_1} \\ &= \boldsymbol{\nu_1}^T \overline{\boldsymbol{\nu_2}} \boldsymbol{\nu_1} \text{ for } \|\boldsymbol{\nu_1}\| = 1 \end{aligned} \tag{3.31}$$

The aforementioned property also leads to $\operatorname{tr}[S(\boldsymbol{\nu_1}) S(\boldsymbol{\nu_2})] = (\boldsymbol{\nu_1}^T \boldsymbol{\nu_2}) \operatorname{tr}[1 - \boldsymbol{I}] = -2\boldsymbol{\nu_1}^T \boldsymbol{\nu_2}$, which can help simplifying the first term of the Lyapunov function's derivative. Considering the trace of a product, the second and forth terms in Equation (3.30) disappear, since $\operatorname{tr}[kS(\boldsymbol{\nu})] = 0$. With these actions, the derivative takes the form

$$\dot{V}(\tilde{\boldsymbol{R}}) = 2\sin(\xi) \boldsymbol{\eta}^T \boldsymbol{\delta\omega} + 2K_{obs} \sin^2(\xi) \boldsymbol{\eta}^T {}^B\overline{\boldsymbol{O}} \boldsymbol{\eta} \tag{3.32}$$

where ${}^B\overline{\boldsymbol{O}} = {}^B\boldsymbol{O}{}^B\boldsymbol{O}^T - \mathrm{tr}({}^B\boldsymbol{O}{}^B\boldsymbol{O}^T)$ is a negative definite matrix. For any given symmetric matrix $\boldsymbol{\zeta}$, the Rayleigh-Ritz inequality says that $\lambda_{min}(\boldsymbol{\zeta})\boldsymbol{\nu_2}^T\boldsymbol{\nu_2} \leq \boldsymbol{\nu_2}^T\boldsymbol{\zeta}\boldsymbol{\nu_2} \leq \lambda_{max}(\boldsymbol{\zeta})\boldsymbol{\nu_2}^T\boldsymbol{\nu_2}$, so taking into account the properties of ${}^B\overline{\boldsymbol{O}}$, the upper bound of $-{}^B\overline{\boldsymbol{O}}$ would be $\lambda_{min}({}^B\overline{\boldsymbol{O}})$. Because the main idea was to analyze the effect of $\boldsymbol{\delta\omega}$ in the derivative of the Lyapunov function and its consequences regarding stability, the next step is determining its bounds through

$$\begin{aligned}\dot{V}(\tilde{\boldsymbol{R}}) &\leq 2|\sin(\xi)|\|\boldsymbol{\delta\omega}\| - 2K_{obs}|\sin(\xi)|^2\lambda_{min}({}^B\overline{\boldsymbol{O}}) \\ &= 2|\sin(\xi)|\left[\|\boldsymbol{\delta\omega}\| - K_{obs}|\sin(\xi)|\lambda_{min}({}^B\overline{\boldsymbol{O}})\right]\end{aligned} \tag{3.33}$$

to find out how the precision limitation affects the convergence and what its biggest error can approximately be. It can be shown that the Lyapunov function's derivative is negative definite while

$$\|\boldsymbol{\delta\omega}\| - K_{obs}|\sin(\xi)|\lambda_{min}({}^B\overline{\boldsymbol{O}}) < 0 \Leftrightarrow |\sin(\xi)| > \frac{1}{K_{obs}\lambda_{min}({}^B\overline{\boldsymbol{O}})}\|\boldsymbol{\delta\omega}\| \tag{3.34}$$

Thus, in the presence of sufficiently small angular velocity bias, the estimation error has an ultimate bound [22]. □

It is worth noting that, even when the LiDAR is encountering only one edge, the accelerometer provides an additional measurement. This prevents the ${}^B\boldsymbol{O}$ matrix from having just one meaningful column, which would lead to $\lambda_{min}({}^B\overline{\boldsymbol{O}}) = 0$. Without bias and having at least two measurements, the derivative of the Lyapunov function will be negative definite, whereas when bias is added this can only happen when $|\sin(\xi)|$ is high enough to overpower the term with $\|\boldsymbol{\delta\omega}\|$, according to (3.34).

Chapter 4

Simulation

With the routines ready to implement, the next step is to build a simulation environment, suited for testing them and remove the dependence on the actual hardware, so that new scenarios can be evaluated and several parameters adjusted.

The simulation architecture is presented in Figure 4.1 and it is divided into five parts, where each one will be described separately in Sections 4.1 to 4.5.

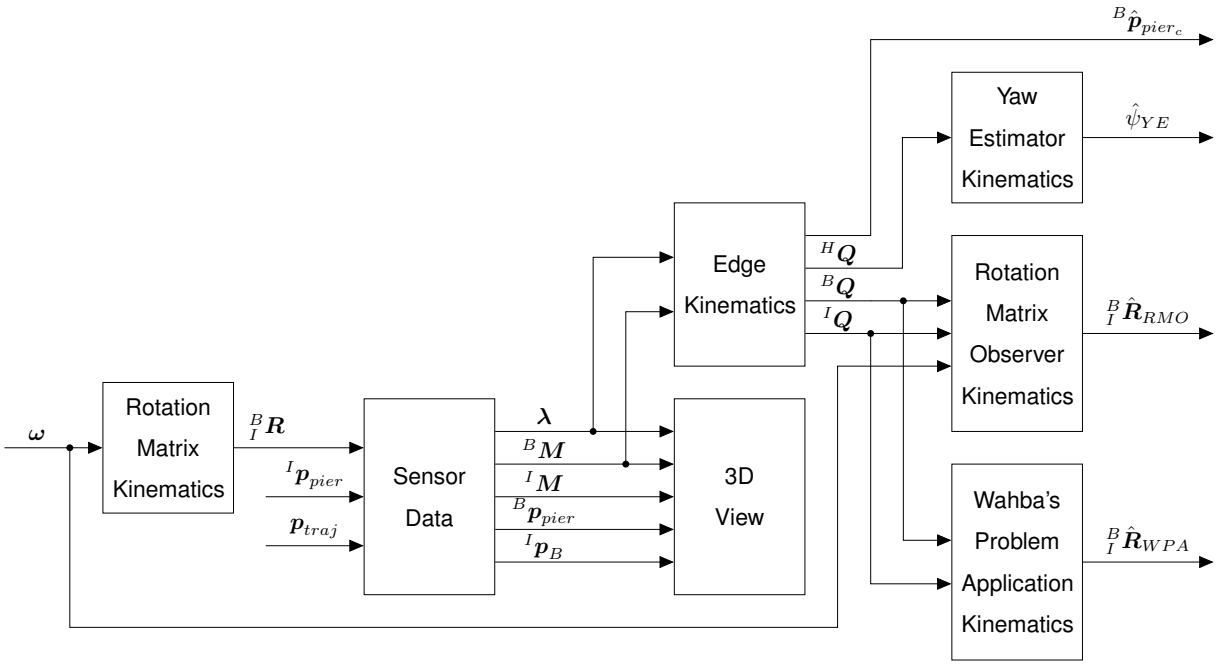


Figure 4.1: Block diagram of the simulation.

Before defining the several components of this diagram, some data structures have to be set. The pier is defined by $^{I}\boldsymbol{p}_{pier}$, which contains eight vertices representing its rectangular sections at the top and bottom, composed by four points each. Those vertices are initialized in the beginning of the simulation, knowing the characteristics of the rectangular section and the distance from the center of the pier to the origin. The trajectory of the vehicle $\boldsymbol{p}_{traj}$ is also known beforehand, consisting of only an history of distances and respective angle to the center of the pier.

4.1 Rotation Matrix Kinematics

This block concerns the kinematics of the rotation matrix from $\{I\}$ to $\{B\}$ and it is visible in the leftmost area of Figure 4.1.

The system receives as input the angular velocity $\boldsymbol{\omega} = \begin{bmatrix} \omega_x & \omega_y & \omega_z \end{bmatrix}^T$ desired for the vehicle and computes ${}^B_I\boldsymbol{R}$ at each time, which is directly related with the attitude the vehicle exhibits.

This simulation, whose base is the naturally continuous-time system presented in Equation (3.17), should run and produce all the data in discrete-time, with the same sampling time as the actual LiDAR. Its discretization can be accomplished by considering the time varying $\boldsymbol{\omega}$ vector as constant for an infinitesimal period of time dt. All that is left is to take the exponential of the product of the system's $\boldsymbol{A}$ matrix, obtained from the unbiased angular velocity, with the sampling time, according to

$$ {}^B_I\boldsymbol{R}(k+1) = e^{-S(\boldsymbol{\omega})h_{LiDAR}} {}^B_I\boldsymbol{R}(k) \tag{4.1} $$

It should be noted that the bias of the angular velocity was not estimated and then included in the simulation, a task that is left for future work. Moreover, imposing a sampling time to the model, in order to discretize it, does not prevent smooth transitions in the motion kinematics and leads to a proper reproduction of the data set observed in the real world.

4.2 Sensor Data

The Sensor Data block matches the process that, given the rotation matrix from $\{I\}$ to $\{B\}$, yields the set of data points ${}^B\boldsymbol{M}$ resulting from modeling the LiDAR, ready to be used in the previously presented routines, together with all the data structures necessary for visualizing the events.

The fundamental idea is to compute the vehicle's position ${}^I\boldsymbol{p}_B$ and attitude $\boldsymbol{\lambda}$ at each time, the former using the knowledge of the desired trajectory $\boldsymbol{p}_{traj}$ beforehand and the latter given ${}^B_I\boldsymbol{R}$ coming from the previous block. Having the location of the pier ${}^I\boldsymbol{p}_{pier}$ as well, it is possible to determine the location of the pier relative to the vehicle ${}^B\boldsymbol{p}_{pier}$, which is as if the pier is being viewed from the vehicle's Point Of View (POV).

The next step is to mimic the process by which the LiDAR sensor actually acquires the data points. According to the LiDAR's data sheet, it emits 1081 beams between -135 and $135°$, with the origin at the positive X axis in $\{B\}$. Therefore, knowing the angle of each beam, only the range remains undetermined. Having the pier defined in $\{B\}$, the furthest point can be easily obtained and its distance to the vehicle taken as the range used for every beam. There are toolboxes with functions especially developed for computing the intersection between a ray and a 3D polyhedron available. Despite their completeness, these solutions are exceedingly expensive at a computational level, in terms of time, to apply for the total number of beams during a large amount of steps. With that in mind, a new method was developed, with the goal of optimizing the speed of the data point acquisition, being implemented in Algorithm 8 and exemplified graphically in Figure 4.2.

Algorithm 8 Intersect Ray Polyhedron

procedure INTERSECTRAYPOLYHEDRON($\boldsymbol{\alpha}, \boldsymbol{n}, \boldsymbol{p}_{face_c}, \boldsymbol{r}_{beam_{max}}, \boldsymbol{R}_{zyx}, L$)
  $\boldsymbol{u}_{beam} = [\cos(\boldsymbol{\alpha}), \sin(\boldsymbol{\alpha}), 0]^T$
  $i_{point} = 0$
  for $i = 1 : \text{length}(\boldsymbol{\alpha})$ **do**
    for $j = 1 : 2$ **do**
      $r_{beam_j} = \frac{\boldsymbol{n}_j^T \boldsymbol{p}_{face_{c_j}}}{\boldsymbol{n}_j^T \boldsymbol{u}_{beam}(i)}$
    end for
    if $r_{beam_1} \notin \left[0, r_{beam_{max_1}}\right]$ **and** $r_{beam_2} \notin \left[0, r_{beam_{max_2}}\right]$ **then**
      continue
    else if $r_{beam_1} \notin \left[0, r_{beam_{max_1}}\right]$ **then**
      ${}^B\boldsymbol{f} = -\boldsymbol{p}_{face_{c_2}} + r_{beam_2}\boldsymbol{u}_{beam}(i)$
      ${}^I\boldsymbol{f} = \boldsymbol{R}_{zyx}^T {}^B\boldsymbol{f}$
      if $\|\boldsymbol{\Pi}_{\boldsymbol{e}_3} {}^I\boldsymbol{f}\| < 0.5 \cdot L_2$ **then**
        $i_{point} = i_{point} + 1$
        $\boldsymbol{M}(i_{point}, :) = r_{beam_2}\boldsymbol{\Pi}_{\boldsymbol{e}_3}\boldsymbol{u}_{beam}^T(i)$
      else
        if $i_{point} > 1$ **then**
          break
        end if
      end if
    else if $r_{beam_2} \notin \left[0, r_{beam_{max_2}}\right]$ **then**
      ${}^B\boldsymbol{f} = -\boldsymbol{p}_{face_{c_1}} + r_{beam_1}\boldsymbol{u}_{beam}(i)$
      ${}^I\boldsymbol{f} = \boldsymbol{R}_{zyx}^T {}^B\boldsymbol{f}$
      if $\|\boldsymbol{\Pi}_{\boldsymbol{e}_3} {}^I\boldsymbol{f}\| < 0.5 \cdot L_1$ **then**
        $i_{point} = i_{point} + 1$
        $\boldsymbol{M}(i_{point}, :) = r_{beam_1}\boldsymbol{\Pi}_{\boldsymbol{e}_3}\boldsymbol{u}_{beam}^T(i)$
      else
        if $i_{point} > 1$ **then**
          break
        end if
      end if
    else
      for $j = 1 : 2$ **do**
        ${}^B\boldsymbol{f}_j = -\boldsymbol{p}_{face_{c_j}} + r_{beam_j}\boldsymbol{u}_{beam}(i)$
        ${}^I\boldsymbol{f}_j = \boldsymbol{R}_{zyx}^T {}^B\boldsymbol{f}_j$
      end for
      if $\|\boldsymbol{\Pi}_{\boldsymbol{e}_3} {}^I\boldsymbol{f}_1\| \leq 0.5 \cdot L_1$ **and** $\|\boldsymbol{\Pi}_{\boldsymbol{e}_3} {}^I\boldsymbol{f}_2\| \leq 0.5 \cdot L_2$ **then**
        $i_{point} = i_{point} + 1$
        $\boldsymbol{M}(i_{point}, :) = \min(r_{beam})\boldsymbol{\Pi}_{\boldsymbol{e}_3}\boldsymbol{u}_{beam}^T(i)$
      else if $\|{}^I\boldsymbol{f}(1:2, 1)\| \leq 0.5 \cdot L_1$ **then**
        $i_{point} = i_{point} + 1$
        $\boldsymbol{M}(i_{point}, :) = r_{beam_1}\boldsymbol{\Pi}_{\boldsymbol{e}_3}\boldsymbol{u}_{beam}^T(i)$
      else if $\|{}^I\boldsymbol{f}(1:2, 2)\| \leq 0.5 \cdot L_2$ **then**
        $i_{point} = i_{point} + 1$
        $\boldsymbol{M}(i_{point}, :) = r_{beam_2}\boldsymbol{\Pi}_{\boldsymbol{e}_3}\boldsymbol{u}_{beam}^T(i)$
      else
        if $i_{point} > 1$ **then**
          break
        end if
      end if
    end if
  end for
end procedure

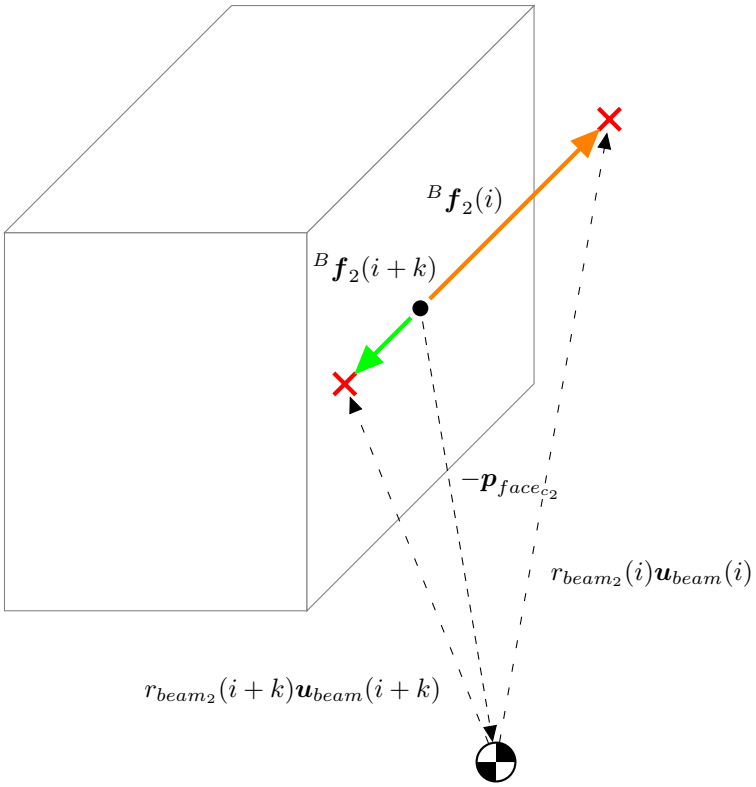


Figure 4.2: Example of the intersection of two beams (long dashed lines) in different steps with the plane of the visible face on the right and their acceptance (green line) or rejection (orange line) based on the comparison between the norm of the corresponding ${}^{I}\boldsymbol{f}_2$ and the length of the face's edge.

In a given instant, there are always two faces of the pier which are closer to the vehicle, except when it makes a perfect 90° angle to the pier, in which there is only one. Similarly as before, now the closest point of the pier can be found from the complete 3D definition available, which in turn allows for the associated faces to be known. This algorithm takes the set of the laser beam's heading angles $\boldsymbol{\alpha}$, the normal vector $\boldsymbol{n}$ and centroid $\boldsymbol{p}_{face_c}$ of the two visible faces, a length threshold $\boldsymbol{r}_{beam_{max}}$ for the beams, the rotation matrix ${}^{B}_{I}\boldsymbol{R}$, and the length of each face's edge L. Knowing that the dot product of a face's normal vector $\boldsymbol{n}_j$ with a vector ${}^{B}\boldsymbol{f}_j$ alongside that face is zero, it is possible to build the latter so that it contains the information regarding the range of the beam r_{beam_j} it took to intersect the plane where the j^{th} face is located. Finding the range of the i^{th} beam, when it intersects the face's plane, can be done through

$$\boldsymbol{n}_j^T {}^{B}\boldsymbol{f}_j = \boldsymbol{n}_j^T \left[-\boldsymbol{p}_{face_{c_j}} + r_{beam_j}\boldsymbol{u}_{beam}(i)\right] = 0 \Leftrightarrow r_{beam_j} = \frac{\boldsymbol{n}_j^T \boldsymbol{p}_{face_{c_j}}}{\boldsymbol{n}_j^T \boldsymbol{u}_{beam}(i)}, \text{ for } j = 1, 2 \tag{4.2}$$

where $\boldsymbol{u}_{beam}(i) = \begin{bmatrix}\cos(\boldsymbol{\alpha}(i)) & \sin(\boldsymbol{\alpha}(i)) & 0\end{bmatrix}^T$ with $i = 1, ..., 1081$ is its direction vector. This way, ${}^{B}\boldsymbol{f}_j$ is the sum of the vector between the centroid $\boldsymbol{p}_{face_{c_j}}$ and the vehicle location, i.e. the origin of the reference frame, with the vector between that location and the point where the beam intersects the plane. It should be noted that the actual face is only a small portion of the infinite plane it is contained in, thus all the intersections that fall outside the actual face must be disregarded. With a valid intersection, rotating ${}^{B}\boldsymbol{f}_j$ to $\{I\}$ yields a vector whose norm along the XY plane is necessarily smaller or equal than half

of the length of that face's edge L_j, which can be used for screening purposes. Additionally, because the data points are contiguous, when the algorithm finds an intersection outside the boundaries after starting to obtain valid matches, it means the pier has ended. Thus, the scan stops to prevent spending time analyzing directions that will no longer yield relevant information.

Another consequence of using planes that extend to infinity is that the calculation in Equation (4.2) can yield both positive and negative values. Clearly only the former will have real physical meaning, since they prolong the beam in its intended direction, whereas a negative r_{beam_j} corresponds to an intersection diametrically opposed, which can affect the sequential sampling of the points and cause a premature ending of the algorithm. On the other hand, having complete knowledge about the location of the pier, an upper boundary for the range of the beam can also be found. This way, $r_{beam_{max_j}}$ contains the range along the XY plane of the furthest point within that face. Because the first task in each step is obtaining this range, imposing these limits to $\boldsymbol{r}_{beam}$ can be used to reduce the computational workload, by only considering one of the faces or even entirely skipping beams, depending on which intersections meet the parameters. When both intersections are within acceptable values, most of the times only one is actually valid. However, close to the perfect 90° angle it can happen that both of them fulfill the necessary conditions thus far. This occurs because the difference in the closest corner is so thin that one of the faces gets shielded behind the other and, for a few beams, this ambiguity arises. The solution is ignoring the furthest intersection, because being behind another face it cannot actually be seen. Implementing these verifications, to guarantee that only the relevant intersections are treated, resulted into a reduction of the execution time to about 50% of the original duration of this algorithm.

These data points correspond to a perfect sampling of the outside world, therefore some noise needs to be added to the measurements, in order to obtain data points closer to reality. Choosing a white noise, with a standard deviation of 3 mm, provided data points very similar to those encountered during the design of the data processing methods. In order to account for the reflection errors at the boundaries of the pier, the last three data points at each end were given a white noise with respectively $\frac{4}{3}$, $\frac{5}{3}$, and $\frac{6}{3}$ of the standard deviation of the remaining measurements.

The last operation in this part is to obtain the data points in another reference frame. The sensor provides the measurements in $\{B\}$, however to complement the representation of the environment, it is necessary to use the aforementioned $\boldsymbol{\lambda}$, to compute the set of data points ${}^{I}\boldsymbol{M}$ as well.

4.3 Edge Kinematics

The block entitled Edge Kinematics in Figure 4.1 comprises the basis of the previously developed work, in the form of a unified dynamic system defined as

$$\begin{bmatrix} {}^{B}\hat{\boldsymbol{p}}_{pier_c}(k) \\ {}^{H}\boldsymbol{Q}(k) \\ {}^{B}\boldsymbol{Q}(k) \\ {}^{I}\boldsymbol{Q}(k) \end{bmatrix} = f_{LiDAR}\left(\begin{bmatrix} {}^{B}\boldsymbol{Q}(k-1) \\ {}^{I}\boldsymbol{Q}(k-1) \end{bmatrix}, {}^{B}\boldsymbol{M}(k), \boldsymbol{\lambda}(k) \right) \tag{4.3}$$

This system has ten states and sixteen outputs, since each edge $\boldsymbol{Q}_i$ with $i = 1, 2$ has two components in $\{H\}$ and $\{B\}$, due to the lack of Z coordinate in these reference frames, same as the location of the pier's center, and three in $\{I\}$. It receives as input the set of data points ${}^{B}\boldsymbol{M}$ resulting from the latest beam swipe and the current vehicle's attitude $\boldsymbol{\lambda}$. The function f_{LiDAR} represents the processes involved in the routines explained during Chapter 2.

4.4 Attitude Computation

This part is where the three main methods for attitude determination are implemented, each under the form of a discrete-time system, working on the LiDAR's sampling time and described in Sections 4.4.1 to 4.4.3.

4.4.1 Yaw Estimator Kinematics

The first method aims to obtain the yaw angle, as explained in Section 3.1.2. The function responsible for this task is

$$\hat{\psi}_{YE}(k) = \mathrm{YawEstimator}({}^{H}\boldsymbol{Q}(k-1), {}^{H}\boldsymbol{Q}(k), \hat{\psi}_{YE}(k-1)) \tag{4.4}$$

receiving the previous and current edges $\boldsymbol{Q}$ in $\{H\}$ and the previous yaw angle estimation $\hat{\psi}_{YE}$. Because during the simulation there is no access to the data from the IMU, the best measure for the true vehicle's attitude comes from the rotation matrix obtained through the integration of angular velocity, as presented in Section 3.2.2. However, since during the simulation the angular velocity is assumed to be unbiased, that attitude is considered the ground truth.

4.4.2 Rotation Matrix Observer Kinematics

The kinematics for this system follow from the explanation present in Section 3.2.2, that combined with the discretization discussed in Section 4.1 yield

$${}^{B}_{I}\hat{\boldsymbol{R}}_{RMO}(k) = e^{-S(\hat{\boldsymbol{\omega}}(k))h_{LiDAR}}\,{}^{B}_{I}\hat{\boldsymbol{R}}_{RMO}(k-1) \tag{4.5}$$

In this system, $\hat{\omega}$ depends on ω, which outside the simulation would be acquired through the IMU. Since this source of data is not available in the simulation, the angular velocity will correspond to the provided input. The angular velocity coming from the IMU is rather noisy, when compared to the supplied constant step input, thus the simulation can yield results presenting smoother transitions than an experiment with similar conditions.

4.4.3 Wahba's Problem Application Kinematics

This last method is also implemented as a function

$$ {}^{B}_{I}\hat{\boldsymbol{R}}_{WPA}(k) = \text{WahbasProblemApplication}({}^{B}\boldsymbol{Q}(k), {}^{I}\boldsymbol{Q}(k)) \tag{4.6} $$

Its inputs are the current edges $\boldsymbol{Q}$, in both $\{B\}$ and $\{I\}$, yielding the rotation matrix that optimally describes the transformation from the latter reference frame to the former.

4.5 3D View

The 3D View block is designed to provide valuable intuition, since it allows the visualization of the whole interaction between the vehicle and the pier, in both $\{I\}$ and $\{B\}$.

There are two viewing modes, the first in which the environment is displayed by considering $\{I\}$ as the reference frame, so the pier is fixed at a given known location and the vehicle can be seen moving, according to its predefined trajectory around the pier. Additionally, $\boldsymbol{\lambda}$ is used to properly represent the attitude of the vehicle, in this reference frame, and the data points collected by the vehicle are projected in the pier. The alternative method of representing the surroundings is to take $\{B\}$ as the reference frame, where in this case the vehicle is fixed at the origin and the pier moves in relation to the vehicle, as if it was being seen through its "eyes". With this view, the data points remain in the same level as the vehicle, since they are the result of a beam swipe.

Both these modes can be seen in Figures 4.3 and 4.4, in which the vehicle is performing a circular trajectory around the pier, with a sinusoidal input to the angular velocities in the X and Y axes and a constant angular velocity in the Z axis.

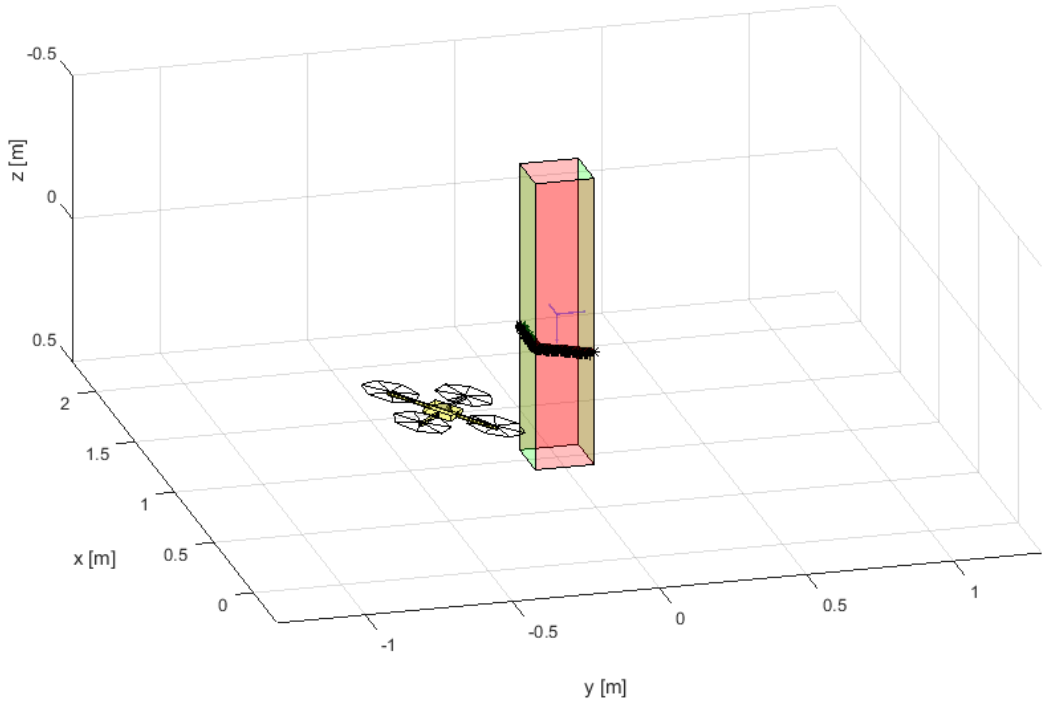


Figure 4.3: Visualization in $\{I\}$ of a circular trajectory around the pier by the vehicle.

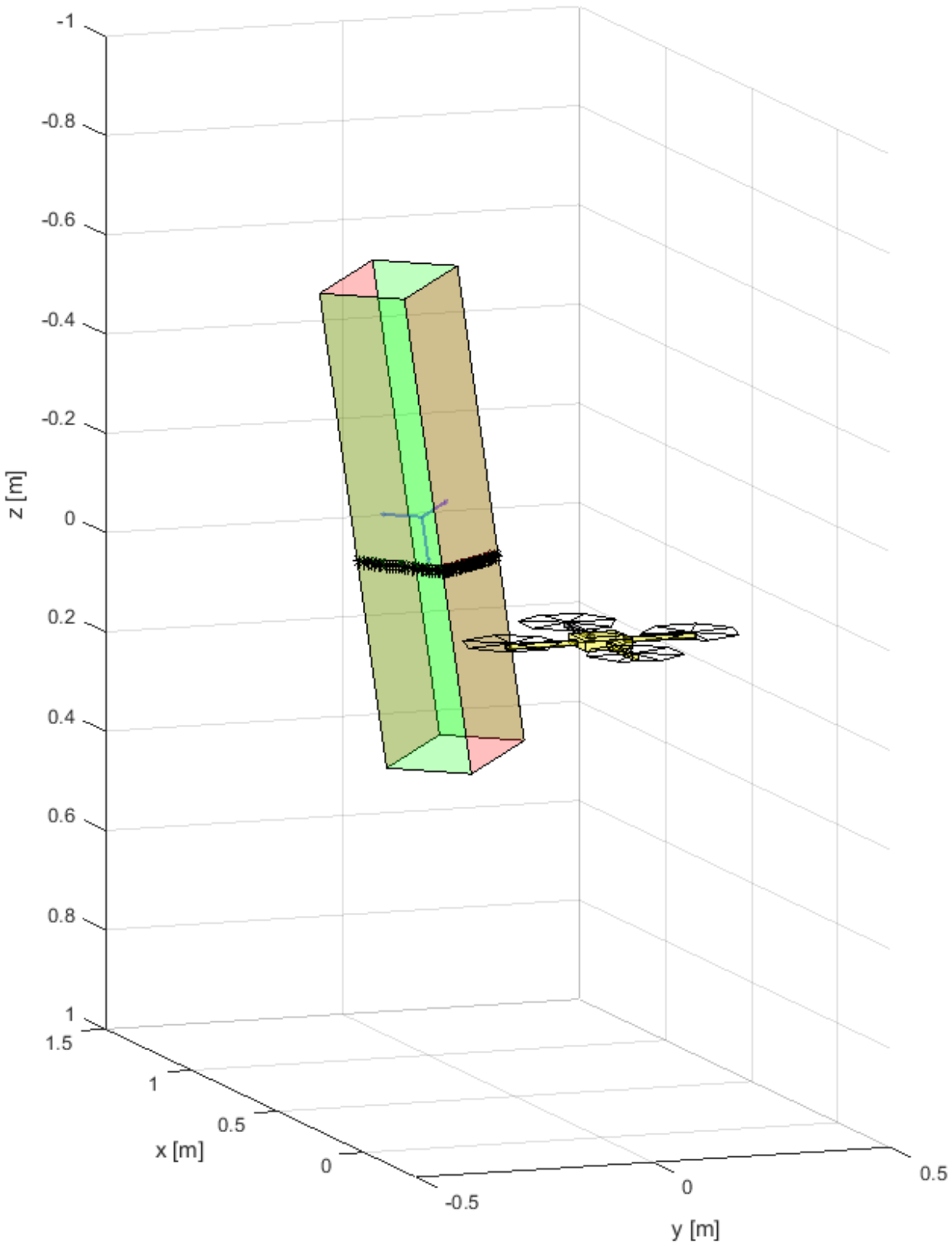


Figure 4.4: Visualization in $\{B\}$ of a circular trajectory around the pier by the vehicle.

Chapter 5

Performance Evaluation

In Chapter 3, several methods were developed for attitude determination. Three of them demonstrated potential and will be analyzed in Section 5.2, against what was being previously obtained from the IMU. Before that, two more scenarios were planned and executed, both in a real and in a simulated environment, with the goal of validating the aforementioned methods and are being described in Section 5.1.

5.1 Testing Scenarios

With the intention of testing the methods in a wider range of conditions, several changes were made starting from the initial experiments. These are being detailed in Section 5.1.1, whereas Section 5.1.2 goes through the necessary steps to mimic those conditions in the simulation discussed in Chapter 4.

5.1.1 LiDAR Experiment #3 and #4

These experiments go a step further than the last performed, discussed in Section 2.3.2.2, by considering an object with slightly larger dimensions, more specifically a cuboid cabinet with a rectangular section of 0.95×0.4 m. This is a relevant improvement, since a wider surface resembles more the future structures to be inspected and allows for more data points to be collected, which makes the edge length determination process more accurate. Similarly to Experiment #2, the idea is to start by facing both edges with a 45° angle relative to the vehicle's longitudinal axis and perform, with the rotors off, an about 90° counterclockwise arc around the object. However, this time the quadcopter was moved on a trolley, which might contribute to additional stability and reduction of oscillations.

While Experiment #3 is made aiming at null roll and pitch angles in the image of Experiment #2, with Experiment #4 the objective is to test the data gathering and attitude determination processes with reasonable angles in the two axes related to these movements. For that purpose, the quadcopter was placed on top of the trolley in a leveled position during the former experiment and tilted with wedges in the latter, so that it would register a significant value of those angles.

5.1.2 LiDAR Simulation #1 and #2

The idea behind these simulations is to mimic the conditions in place during the experiments, in order to be able to establish a comparison and see how reliable the simulation platform actually is. The input to the simulation consists of the angular velocity $\boldsymbol{\omega}$ and the initial condition for the rotation matrix kinematics.

The conditions of the experiments translate into angular rates along the axes in $\{B\}$, which are not the same as angular velocities, since they refer to the axes in $\{I\}$. The relationship between those two quantities is

$$\boldsymbol{\omega} = \boldsymbol{Q}^{-1}(\phi, \theta)\dot{\boldsymbol{\lambda}} \Leftrightarrow \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} = \begin{bmatrix} 1 & 0 & -s_\theta \\ 0 & c_\phi & s_\phi c_\theta \\ 0 & -s_\phi & c_\phi c_\theta \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \tag{5.1}$$

therefore to approximate the behavior of the quadcopter during the experiments, the roll and pitch rates are null and the yaw rate is a constant step of $-4°$/s, corresponding to an approximately $90°$ rotation in 22.5 s, while the starting rotation matrix is build with a $45°$ yaw angle. Regarding Simulation #1, there is $\phi = \theta = 0°$ to compute $\boldsymbol{\omega}$ and the initial condition of the rotation matrix, whereas to account for the non-zero roll and pitch angles in Experiment #4, these values were modified to $\phi = 8°$ and $\theta = 12°$ in Simulation #2.

5.2 Analysis of Methods

With the scenarios fully defined in both environments, the implementation of the methods can finally be tested. These approaches were developed in Sections 3.1.2, 3.2.1 and 3.2.2 and will be evaluated in Sections 5.2.1 to 5.2.3 respectively.

The IMU used during these tests is a MicroStrain 3DM-GX3-35. It provides several raw measurements, associated with each sensor that composes it, as well as a set of quantities resulting from an internal filter combining these measurements, such as a rotation matrix.

5.2.1 Yaw Estimator

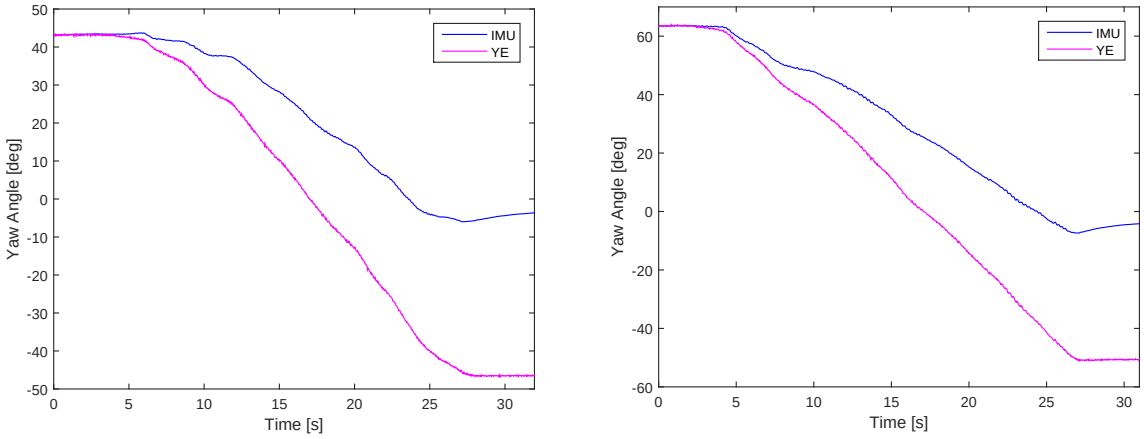
The yaw angle ψ is now being compared between two possible sources, the rotation matrix from the IMU and the continuity processing of the LiDAR data between consecutive steps. Figure 5.1 contains the results for the two aforementioned experiments.

The yaw motion was performed with the objective of maintaining a constant velocity and consequently a linear evolution of the corresponding angle, always in the same direction. Acquiring the yaw angle ψ through the rotation matrix from the IMU, resulting from the combination of a gyro and magnetometers, proves not to be an entirely reliable method. This happens since the total traveled angles of approximately $50°$ and $70°$, in the third and fourth experiments respectively, are much smaller than the actually planned values of around $90°$ for the former and what ended up being slightly above that mark,

in reality, for the latter. Additionally, near the end of the experiment, the IMU shows an inversion in the direction of the motion. This might be due to magnetic interference, in the testing environment, or poor calibration of the algorithms running on the IMU, that fuse the data from the internal sensors to obtain the attitude of the vehicle. On the other hand, the yaw angle ψ computed with LiDAR data presents a realistic start and finish points, leading to a measure of the total traveled angle resembling the initial goal, together with a closely linear slope decrease between these two instants, without motion inversion.

It should be noted that these two methods work according to different principles and have a natural offset at the beginning, which was artificially removed for the purpose of a simpler visualization and comparison. The relevant information retrieved from these results is the total traveled angle and how the angle progresses along the way. Considering those parameters, the LiDAR data provides a description much closer to reality, for what happened during the experiment. These remarks extend to both experiments, with emphasis on the fact that this method was robust enough and yielded consistent results even with significant roll and pitch angles.

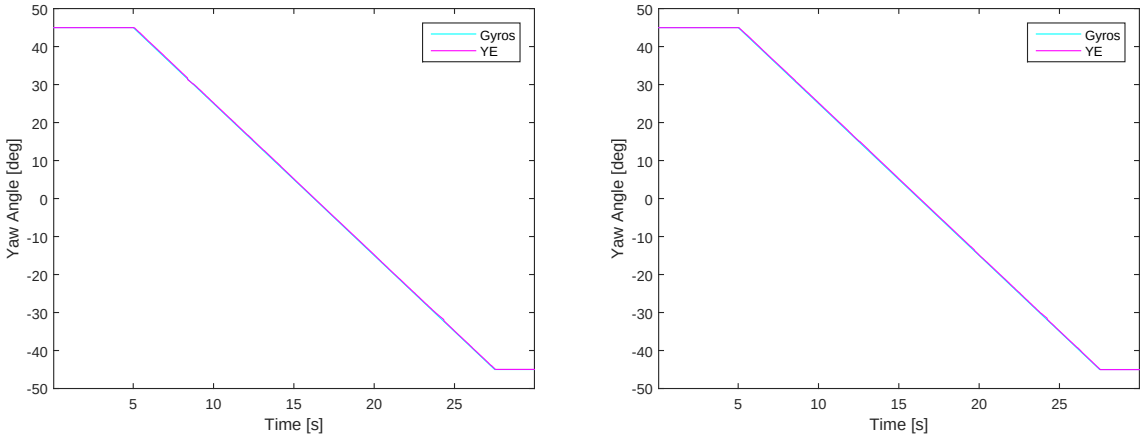
Regarding the simulation, the results are present in Figure 5.2. Aside from the minor inevitable variations of the slope, due to the artificially introduced noise that ended up being well attenuated, it is



(a) Third experiment.

(b) Fourth experiment.

Figure 5.1: Experimental yaw angle of the quadcopter from the IMU and the yaw estimator.



(a) First simulation.

(b) Second simulation.

Figure 5.2: Simulated yaw angle of the quadcopter from the gyroscopes and the yaw estimator.

practically linear and a very good match with the experimental results, both with and without significant roll and pitch angles.

Additionally, despite the rotation matrix used to compute the true attitude being derived from an unbiased angular velocity in the simulation, what matters is how well the yaw angle obtained through this method follows the chosen reference. This can be evaluated resorting to the quantitative analysis of the yaw angle error in Table 5.1.

Error	**Simulation**	
	#1	**#2**
Mean	$-0.15°$	$-0.16°$
Standard Deviation	$0.10°$	$0.12°$
Maximum	$0.48°$	$0.42°$

Table 5.1: Simulated yaw angle's error of the quadcopter from the yaw estimator.

From that perspective, an error under $0.5°$, with an absolute mean and a standard deviation within $0.2°$, for both simulations, proves that this environment achieved its main purpose of providing a trustworthy representation of the results that would be obtained in an experiment, where the only difference is that the rotation matrix would be coming from the IMU.

5.2.2 Wahba's Problem Application

The attitude determination solution provided by the application of the Wahba's problem takes the measurements from the LiDAR and the accelerometer to calculate an optimal rotation matrix. It was discussed in Section 3.2.1 and yields the results presented in Figure 5.3, for the experiments.

One of the first details to be noticed is the noise associated with the angles, specially regarding the roll and pitch and despite the scale of the yaw. This method, by finding the optimal rotation matrix that relates the measurements in both reference frames directly, at each time step independently, lacks some really important filtering capabilities to provide a smooth description. This happens more evidently to the movements in the XY plane, since there are more abrupt variations in the detail of the edges that contribute to those calculations, their length. For the third experiment, the roll angle oscillates predominantly in the positive values, whereas the pitch angle fluctuates around zero. When it comes to the experiment with reasonable roll and pitch angles, both get higher variations. However, the former still has a general tendency to be overestimated, while the latter becomes overall slightly underestimated.

Another point of interest is related with the integration of the data from the accelerometer. Without this additional source, the ambiguity caused by having only one edge in some situations, and therefore trying to resolve so many DOFs with so little information, would cause the results to be completely unusable, with $90°$ jumps at these instants. Adding the acceleration vector to the observations led to the quantitative analysis present in Table 5.2, where the error of each of the angles concerning movements in the XY is kept below $10.1°$, with a mean between $\pm 1.6°$ and a standard deviation lower than $2.6°$.

Error	Experiment				Simulation		
	#3		#4		#1		
	Roll	**Pitch**	**Roll**	**Pitch**	**Roll**	**Pitch**	**Yaw**
Mean	$-1.39°$	$-0.26°$	$-1.57°$	$1.52°$	$0.00°$	$0.00°$	$-0.15°$
Standard Deviation	$2.02°$	$1.47°$	$2.30°$	$2.58°$	$0.00°$	$0.00°$	$0.11°$
Maximum	$10.02°$	$7.75°$	$7.37°$	$9.13°$	$0.00°$	$0.00°$	$0.48°$

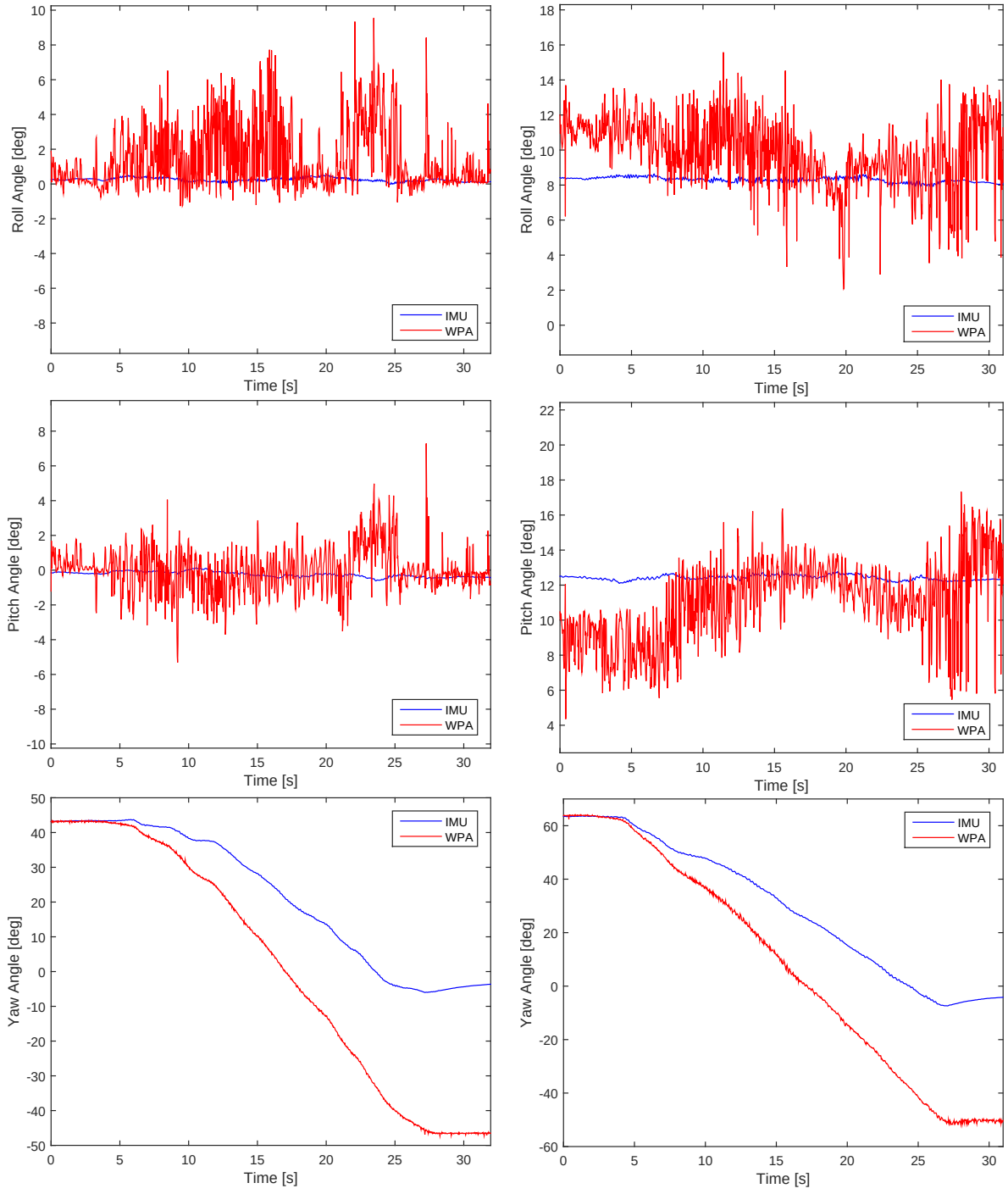
Table 5.2: Experimental and simulated attitude's error of the quadcopter from the Wahba's problem application.

Referring to the yaw angle in particular, it presents a very high resemblance to the previously presented method, which is a way to verify both results. Notice the slightly increased oscillations in this angle during the fourth experiment, in contrast with a steadier response during the previous experiment. The variations in the lengths of the edges mostly affect the roll and pitch angles, however they still have an effect in the yaw angle, although with a reduced magnitude. Because there is no ground truth to compare the yaw angle with, the error in this variable was left out of the quantitative analysis.

Given the properties of this method, such as the independence between steps and consequently the intrinsic noise in the results, it should not be used for deriving an attitude description with the current precision, despite all the three angles showing a close proximity to the expected values, in average. Moreover, with a more accurate LiDAR sensor, both the noise and the error could be significantly reduced up to a point where the use of this method would become feasible, with the advantage of a rather simple implementation and low computational requirements.

Considering the simulated environment, for which the results are shown in Figure 5.4, there are a few limitations while using this method. Because the simulation does not produce the equivalent to the accelerometer measurements, a possible ambiguity cannot be resolved and the aforementioned jumps will show up in the angles. For the first simulation, the intended roll and pitch angles are null so the Z coordinate of the edges in $\{I\}$ is naturally zero. In practice, setting these conditions is virtually the same as removing a DOF from the problem, so the angle duality issue will never arise in this scenario due to a lack of LiDAR measurements. Regarding the second simulation, the additional DOF is inherently present and the issue may present itself when combined with the noise limitations. Despite the best efforts to mimic its real behavior in the data, the noise is always unpredictable, given its statistical properties. It specially affects the length of the edges, which is from where their Z coordinate in $\{I\}$ is determined.

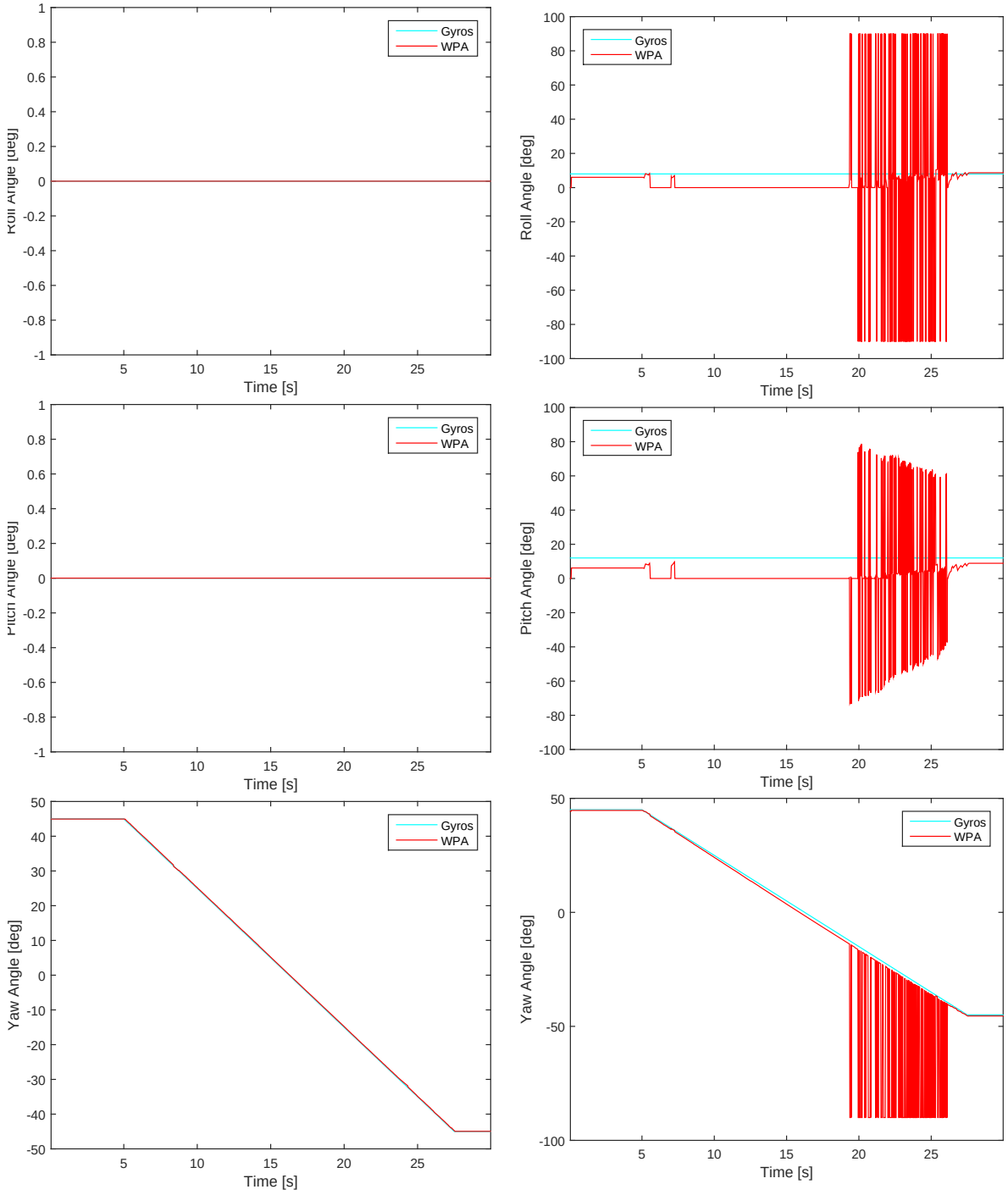
In the first simulation, the results are very accurate because of the roll and pitch conditions, with these angles being null the entire time. The noise causes the lengths to be either really close or even smaller than the real dimensions, which in the optimization discussed in the final part of Section 2.6 leads to a null vertical component. However, if the noise would cause a length larger than the pier's dimensions, it would originate an unrealistic Z coordinate, restoring the lost DOF and causing jumps. These effects were not observed during this simulation, where the yaw angle also presented a great match with the conditions set.



(a) Third experiment.

(b) Fourth experiment.

Figure 5.3: Experimental attitude of the quadcopter from the IMU and the Wahba's problem application.



(a) First simulation.

(b) Second simulation.

Figure 5.4: Simulated attitude of the quadcopter
from the gyroscopes and the Wahba's problem application.

On the other hand, the impact of the limitations that the noise imposes are specially visible during the second simulation. Although the angles set for roll and pitch are considered significant from an attitude perspective, their consequences on the length of the edges is still very small and comparable to the previous simulation. Thus, the noise will cause the Z coordinate of the edges in $\{I\}$ to oscillate between their true and nominal values and the corresponding angles between their desired value and zero. Closer to the end of the second simulation, the ambiguity of having only one edge and the noise, that constantly changes the Z coordinate to zero back and forth, are the two factors that combined originate the jumps observed in all three angles. It should be noted that, until the point when the jumps started, the yaw angle followed the reference closely, but with an error regarding the previous simulation, due to the small but existing effect of the lengths of the edges.

Considering the behavior observed during the last simulation, it makes sense to conduct a quantitative analysis only on the first. The results from Table 5.2 show, as expected, a null error for the roll and pitch. When it comes to the yaw angle, an error smaller than 0.5°, with an absolute mean and a standard deviation not reaching 0.2°, also reveals a great modeling.

Based on all these findings, the simulated environment currently has a sufficient effectiveness to represent this method in a real experiment with null roll and pitch angles. The same does not apply when there are reasonable values of these angles, given the effects caused by the lack of accelerometer data and the performance of the noise reproduction mechanism.

5.2.3 Rotation Matrix Observer

This approach, described in Section 3.2.2, estimates the rotation matrix based on data from the LiDAR, the gyroscopes, and the accelerometers, in order to compute the full attitude of the vehicle, where the last two sensors are part of the IMU. The results using this method, for the experiments, are presented in Figure 5.5.

The first observation concerning these results is the presence of the aforementioned drift in attitude between the rotation matrix from the IMU and from the biased angular velocity. The trend remains the same, but there is a notorious error over time in all three angles, also increasing with motion. This can be verified specially by the large deviation in the yaw angle, probably resulting from a flawed internal compensation regarding the magnetometers, again due to electromagnetic interference or malfunctioning of the data fusing mechanisms within the IMU.

The estimation has the starting rotation matrix from the IMU as initial condition and it presents a satisfactory convergence, in both experiments. These results can be seen as a filtered version of the Wahba's problem application, without all the noise derived from having a step independent approach, where all the angles show the same progression mentioned earlier.

Regarding the roll and pitch angles, the convergence to the data from the gyroscopes is better in the third experiment and their oscillations, in either of them, are once again due to the precision limitations associated with the procedure used to determine the length of the edges. However, this behavior corroborates the fulfillment of the aforementioned Lyapunov conditions for stability, where the error caused by the bias seems to remain bounded. In the quantitative analysis of Table 5.3, the error in

the roll and pitch angles is within 4.9° during either of the experiments, where its mean stays between $\pm 1.6^\circ$ and its standard deviation is kept under 1.7°.

Error	Experiment				Simulation					
	#3		#4		#1			#2		
	Roll	Pitch	Roll	Pitch	Roll	Pitch	Yaw	Roll	Pitch	Yaw
Mean	-1.41°	-0.31°	-1.58°	1.45°	0.00°	0.00°	-0.07°	3.27°	4.75°	0.74°
Standard Deviation	0.93°	0.67°	1.12°	1.64°	0.00°	0.00°	0.06°	3.13°	1.67°	0.55°
Maximum	4.07°	2.64°	3.27°	4.88°	0.00°	0.00°	0.16°	8.54°	7.83°	1.53°

Table 5.3: Experimental and simulated attitude's error of the quadcopter from the rotation matrix observer.

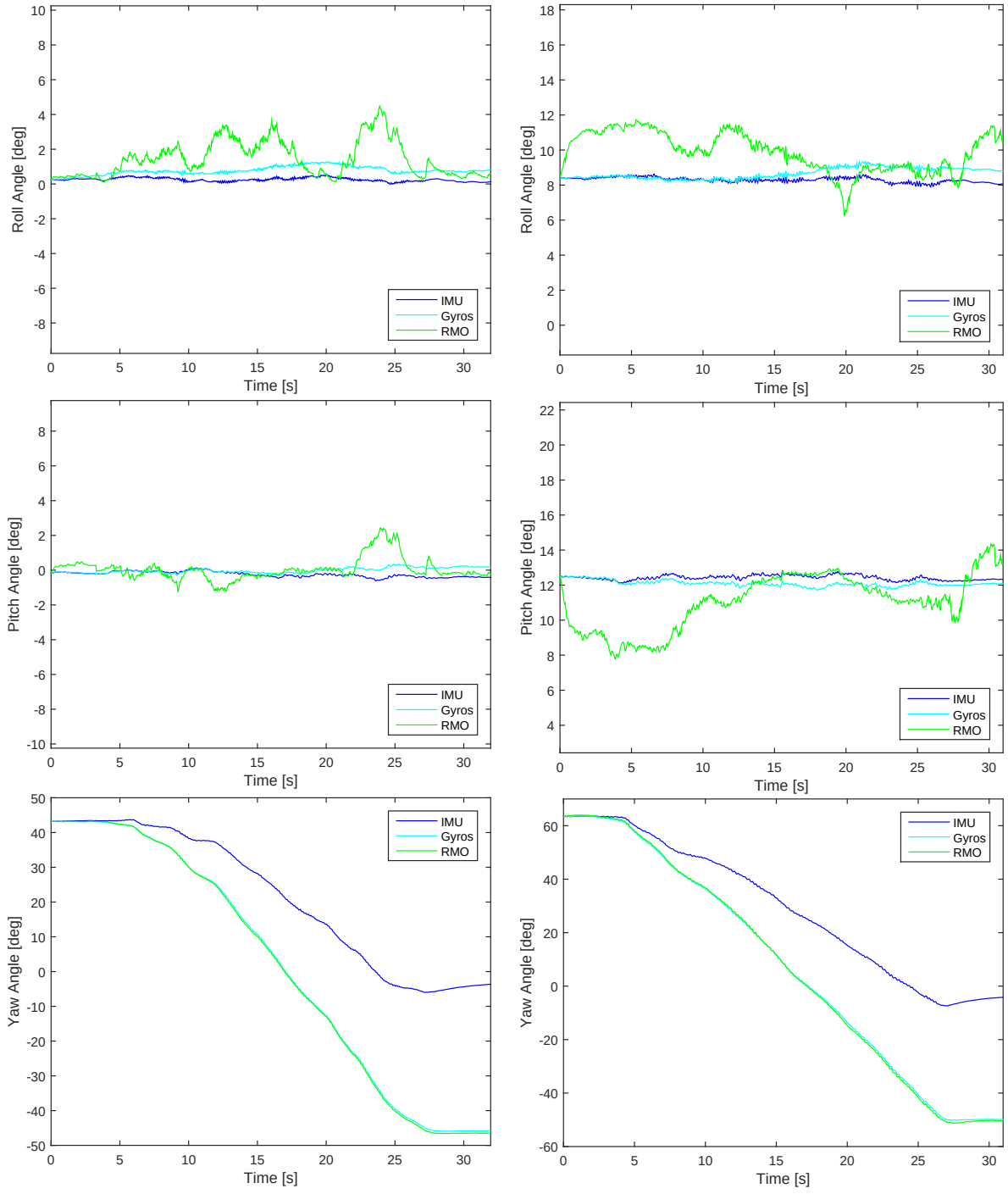
For the yaw angle, in the entire duration of both experiments the convergence is much higher, revealing a great match to the product of biased angular velocity. The response is also almost identical with the previous two methods, which validates the accuracy of this observer regarding the yaw motion, despite its inherent drift originated from the gyroscopes. As observed with the previous method, there is a negligible increase in oscillations from the third to the fourth experiment.

In the end, the proximity in the roll and pitch angles makes them reliable until a certain degree, similar to the application of the Wahba's problem. However, their smooth evolution is what differentiates both methods, which may be enough to control the vehicle with a sufficient accuracy. Once more, improving the LiDAR measurements would reduce the uncertainty in the length of the edges and contribute to a more trustworthy representation of the vehicle's attitude, taking advantage of this method's great filtering characteristics and relatively low computational effort.

When it comes to the simulation, the results shown in Figure 5.6 are relatively consistent with the experiments. In this case, since the attitude is derived from unbiased angular velocity, the existent deviations are only due to the lack of the acceleration and the artificial noise introduced in the data, same as with the simulation of the previous method.

Regarding the roll and pitch angles, the explanation for this behavior is the same as discussed in the final part of Section 5.2.2. In the first simulation, the conditions are favorable to a great accuracy, with all three angles presenting a close tracking of the reference, for the whole duration, despite the number of observations and the noise. However, this method has a consistent output even with only one edge at times, due to its history and filtering capabilities. This means that, in the second simulation, a lack of measurements will not have such a damaging effect on the angles, but will prevent a better convergence throughout time. The deviation it incurs will naturally also be amplified by the noise generation system, ending up to be rather significant. In terms of errors, the results in Table 5.3 are according to the expectation, regarding the first simulation, where the null error, in both angles, follows the same reasoning as with the application of the Wahba's problem. However, this time the second simulation can be analyzed and the error is lower than 8.6°, with a mean between 3.2 and 4.8° and a standard deviation below 3.2°.

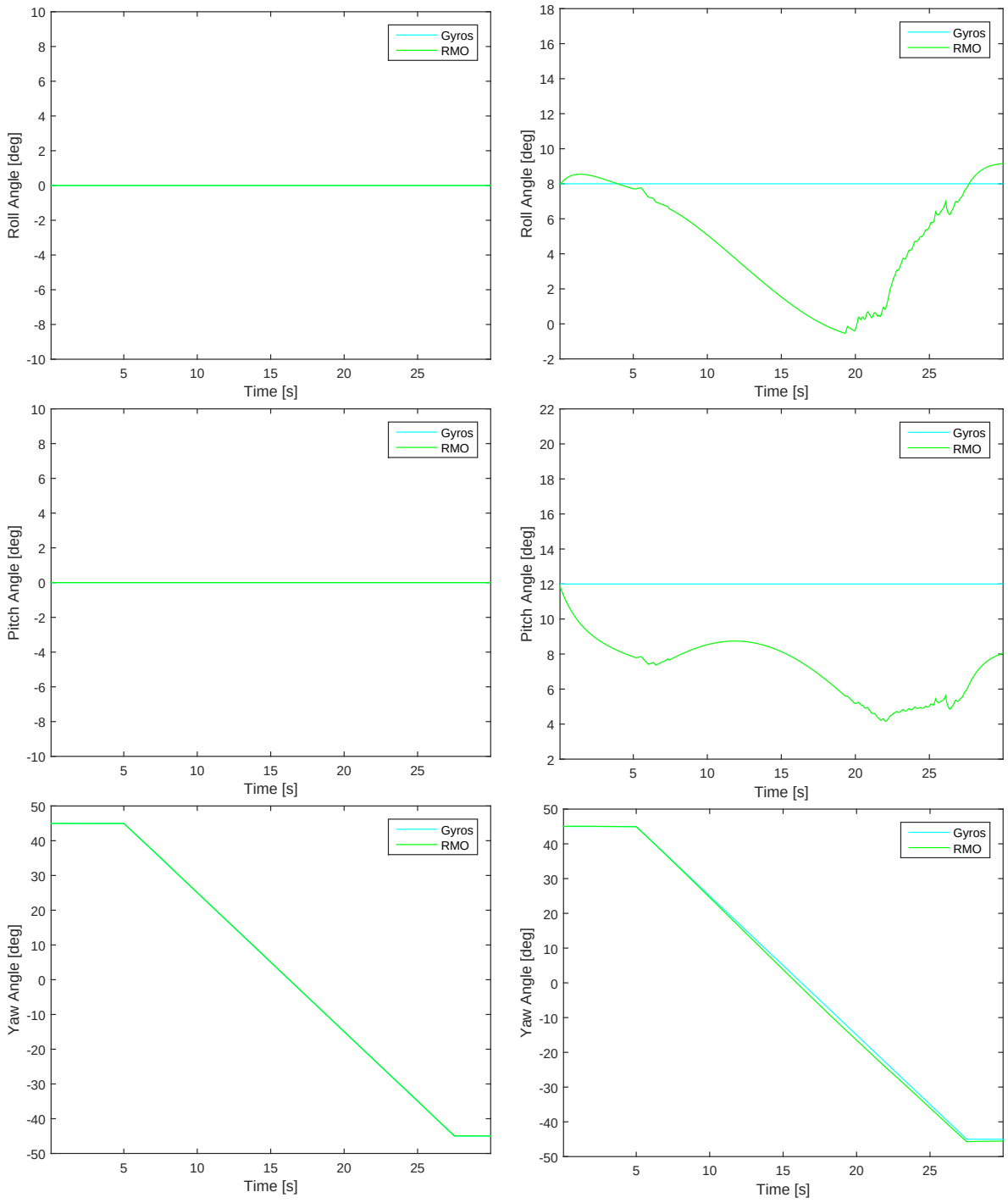
Once more, the yaw angle presents an almost perfect match between the rotation matrix from the estimation and the unbiased angular velocity, in both simulations. Nevertheless, there is a small error of



(a) Third experiment.

(b) Fourth experiment.

Figure 5.5: Experimental attitude of the quadcopter
from the IMU, the gyroscopes, and the rotation matrix observer.



(a) First simulation.

(b) Second simulation.

Figure 5.6: Simulated attitude of the quadcopter
from the gyroscopes and the rotation matrix observer.

the second simulation with respect to the first, as observed with the Wahba's problem application. For the first simulation, the error in this angle does not reach $0.2°$, where its mean and standard deviation are nearly null. When it comes to the second, the error has an upper bound of $1.6°$, with a mean and standard deviation smaller than $0.8°$.

Considering these results, it can be concluded that the simulation properly represents the events from the experiments, although with limitations. According to these scenarios in particular, it proved to be less reliable in describing what actually happens to the attitude with significant roll and pitch angles.

Chapter 6

Synthesis of Controllers

So far, the basic tools for interpreting LiDAR data and retrieving both position and attitude information from it have been discussed in Chapters 2 and 3 respectively, being fully validated later in Chapter 5. The last stage is to build upon that work and design the real-time controllers, in order to achieve the objectives established at the beginning. Because the quadcopter already has a base code created by the research team at ISR, to perform various tasks, Section 6.1 goes through the necessary planning to combine it with the new framework and make sure everything runs properly. Two separate controllers will be designed at this time, with different purposes. The first is explained in Section 6.2, consisting of a simple heading following controller, for a structure matching the proper requirements, and whose goal is to mainly observe the behavior of the detection routines, in real-time, and their compatibility with control tasks. Afterwards, a more evolved controller detailed in Section 6.3 aims at more complex features, such as tracking a desired trajectory, defined relatively to the object of inspection.

6.1 Framework Integration

As mentioned in Section 2.3, there is a pre-existing LiDAR processing within the code that runs in the quadcopter, which is responsible for clustering the data. In order to assess how the current framework affects the runtime of the working setup, the execution times need to be compared and the total duration has to be compatible with the processing power of the on-board CPU. Additionally, the control cycle of the quadcopter is being run at the same frequency as the IMU, so at each step its 50 Hz are the upper limit, in terms of total time.

Until this point, all the code had been developed in MATLAB, whereas the CPU on board runs C code. Therefore, the routines had to be rewritten in this new language so that it could run natively, in the quadcopter. After converting the code, the execution times were revisited, only to find that the new framework led to an increase of less than 20% in the total duration, when compared with the pre-existing code, which meant a change from around 2.1 ms to 2.6 ms. The final value is still reasonably smaller than the maximum of 20 ms allowed, despite these numbers only accounting for the LiDAR data processing.

6.2 Heading Locking Controller

The control of the yaw motion of an aerial vehicle can be usually described by second order dynamics, when considering an actuation in torque. Because the quadcopter used for the testing did not go through an identification procedure, there is not a mathematical model for the dynamics of the system. However, there is an internal controller that given an angular velocity input, generates the necessary torques for the vehicle. Before proceeding to the real-time implementation of a controller capable of maintaining a desired heading to a pier, it is a good policy to build a simulated version in order to predict how the system would behave in reality. For that purpose, the yaw kinematics were modeled as a first order system, whose closed loop transfer function is given by

$$\frac{\omega_z(s)}{\omega_{z_d}(s)} = \frac{1}{T_{\dot{\psi}}s + 1} \tag{6.1}$$

where $T_{\dot{\psi}}$ is the time constant of the model and was chosen as 0.2, in order to produce a behavior which is typical in these vehicles. In principle, the difference in behavior between the real system and the simulation will mostly come down to the relationship between their time constants. To actually control this model, a P controller defined as

$$u = \omega_{z_d} = K_P \tilde{\alpha}_{pier_c} \tag{6.2}$$

was considered to be suited, where $\tilde{\alpha}_{pier_c} = \alpha_{pier_{c_d}} - \hat{\alpha}_{pier_c}$ is the error between the desired and the true heading to the pier's center and K_P is the controller's proportional gain. A very important part of any control system is limiting the actuator inputs, via saturation of the signals. In particular, because the heading of the vehicle with respect to the pier is being controlled and there is a dead zone in the LiDAR angle opening, it is crucial to ensure that it never reaches that area of the LiDAR's field of view, otherwise it would lose sight of the pier and interrupt the entire process without guarantee of recovery. Given that this mechanism is controlling the angular velocity, the saturation limits were set to $-127°$/s and $127°$/s, for reasons that will be discussed shortly.

The complete block diagram of this system, in its simulated version, can be seen in Figure 6.1, in which the initial simulation environment, discussed in Chapter 4, is condensed in the Simulation block, without the viewing data structures and unused methods. This trimming leaves as output the only quantity of interest in this case, which corresponds to the position of the pier's center. Because the actual

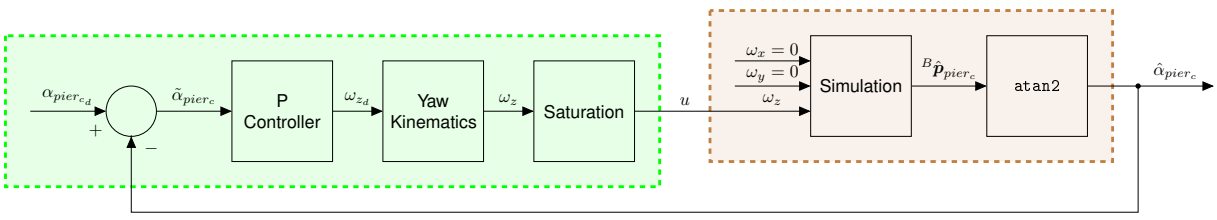
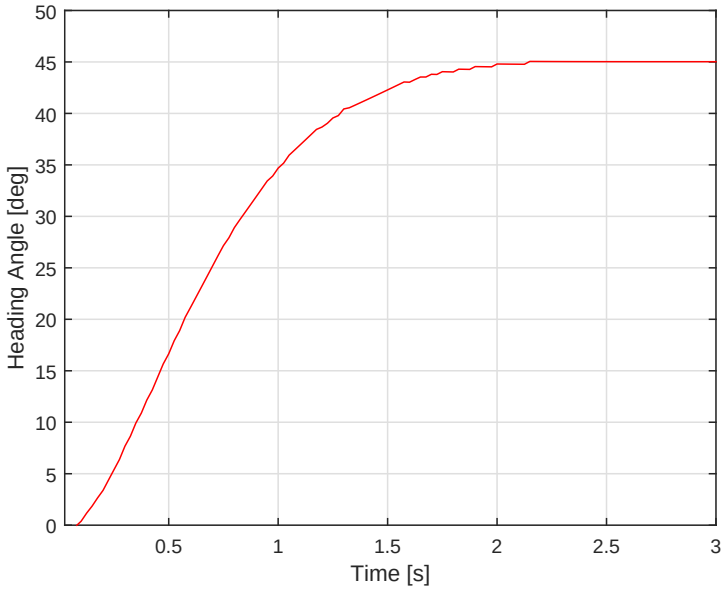


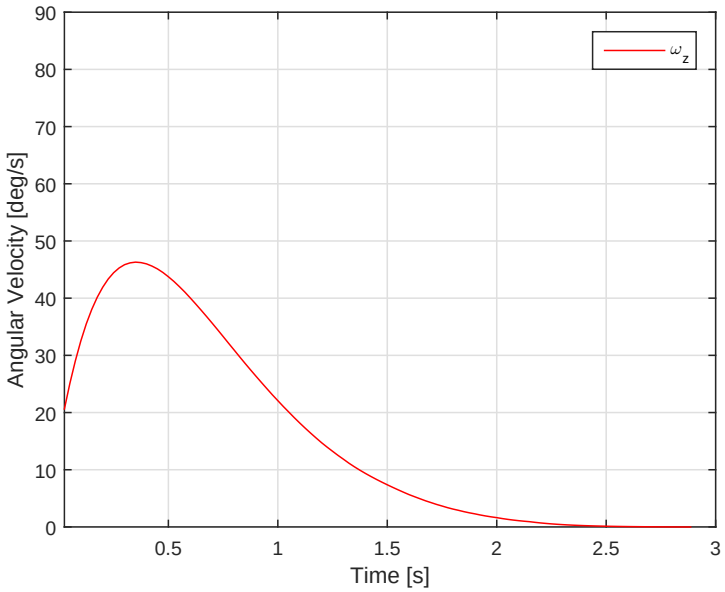
Figure 6.1: Block diagram of the heading locking controlled system distinguishing the controller (green), and the system plus sensor and processing (brown).

control variable is the heading angle of the pier's center in the LiDAR's angle opening, it can be simply computed though an inverse tangent, using the coordinates of this point. Furthermore, the horizontal components of the angular velocity are now null, in order to keep the vehicle in the same position, and the signal from the control loop is being fed to the simulation, as the remaining vertical component.

Concerning the tuning of the controller, the proportional term led to no static error or oscillations. For that reason, no integrative or derivative terms were necessary in the control mechanism. The tuning of the gain is partly a trial and error task, but in the end setting $K_P = 1.3$ produced a prompt response to a setup input of $45°$, with a settling time lower than 2 s and no overshoot or steady state error, as Figure 6.2a shows. The control effort that a controller requires from the system is also a significant performance indicator, which in this case is seen in Figure 6.2b as having reasonable values.



(a) Heading response.



(b) Control input.

Figure 6.2: Simulated output of the heading locking controlled system to a step input of $45°$.

With a successful simulation of the controlled system, the next step is to implement it in the real quadcopter. Before doing so, some precautions were necessary, in order to make sure the site was secure. A platform on which to bound the quadcopter was built, so that it could operate within specific limits, imposed regarding its intended DOFs. For this controller in particular, only movement around the Z axis was necessary. With this in mind, the platform consisted of a very low drag rotating tray, to which the landing gear was strapped on. A photograph of this early setup is presented in Figure 6.3.

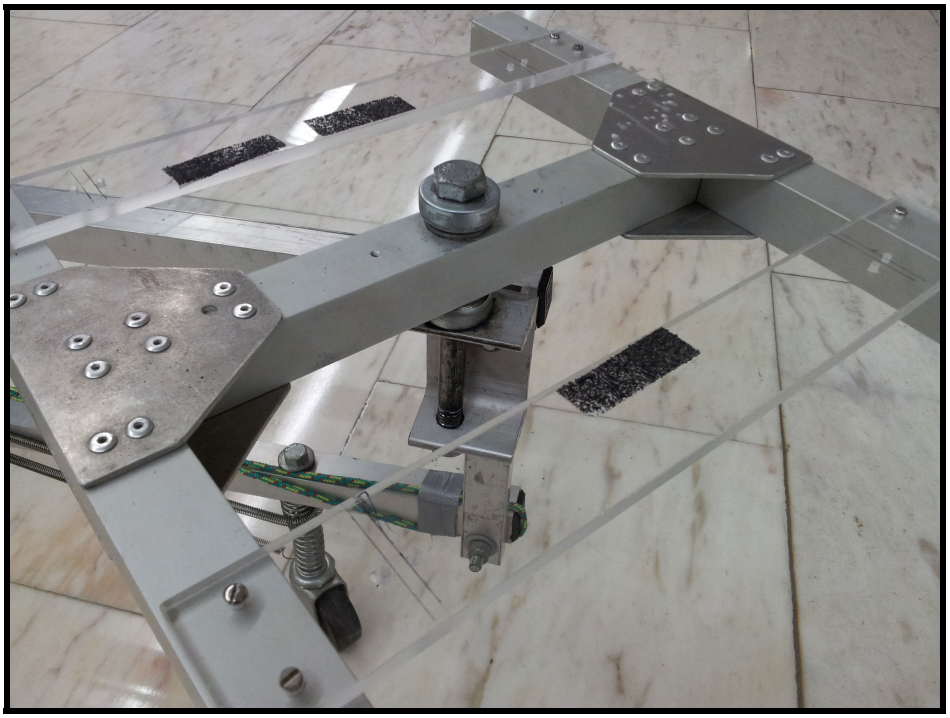


Figure 6.3: Experimental setup of the heading locking controlled system.

Additionally, a Guest User Interface (GUI) was designed to allow a better real-time tuning and visualization of the events. This interface was developed by the research team in the Robot Operating System (ROS) and can be seen in Figure 6.4. It contains a graphic with the evolution of the heading relative to the pier, a 3D view of the LiDAR data points at each beam swipe, and a control area where the heading reference, the P parameter, and the trust command could be adjusted, effective immediately.

There are some details that require further explanation at this point. A quadcopter, clear from its designation, possesses four rotors that are responsible for the movement in the 3D space. Generating motion in roll or pitch is rather intuitive, since it implies shifting power within a pair of opposite rotors, from one side to the other. This makes the vehicle acquire a deviation from the horizontal and therefore the lift force to gain a non-vertical component, with a direction from the higher to the lower power rotor. However, in order to perform a maneuver in yaw, the idea is to have opposite rotors spinning in different directions. Then, having motion in this axis is a matter of slowing down one pair of rotors relative to the other. Moreover, usually these vehicles have a main trust source, called collective, and secondary sources, known as cyclic. Because of these facts, there is a direct dependency between the speed to which the movement is performed and the speed at which both the collective and the cyclic are spinning. This means that the faster the rotors spin, the sooner the control operation will be completed.

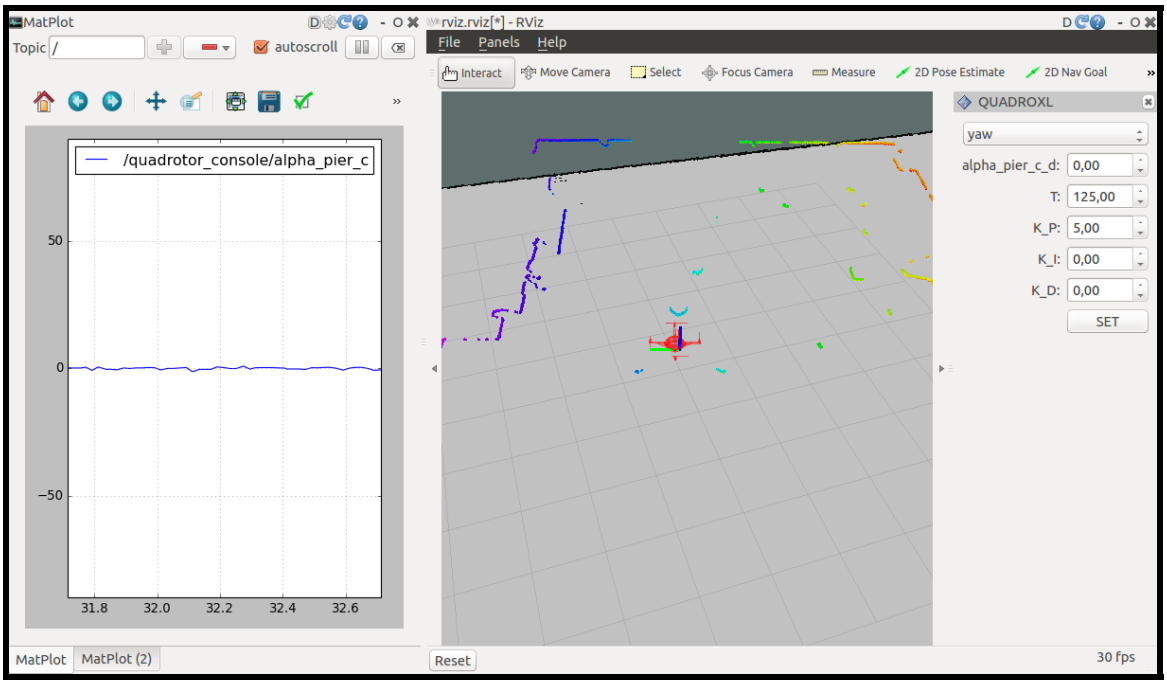
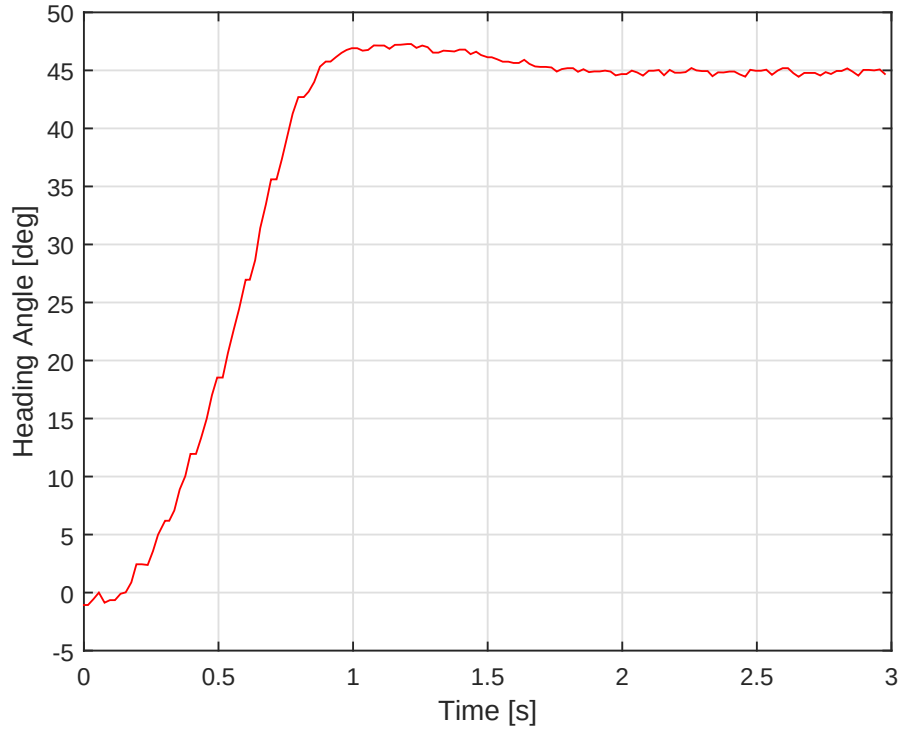


Figure 6.4: Experimental interface of the heading locking controlled system.

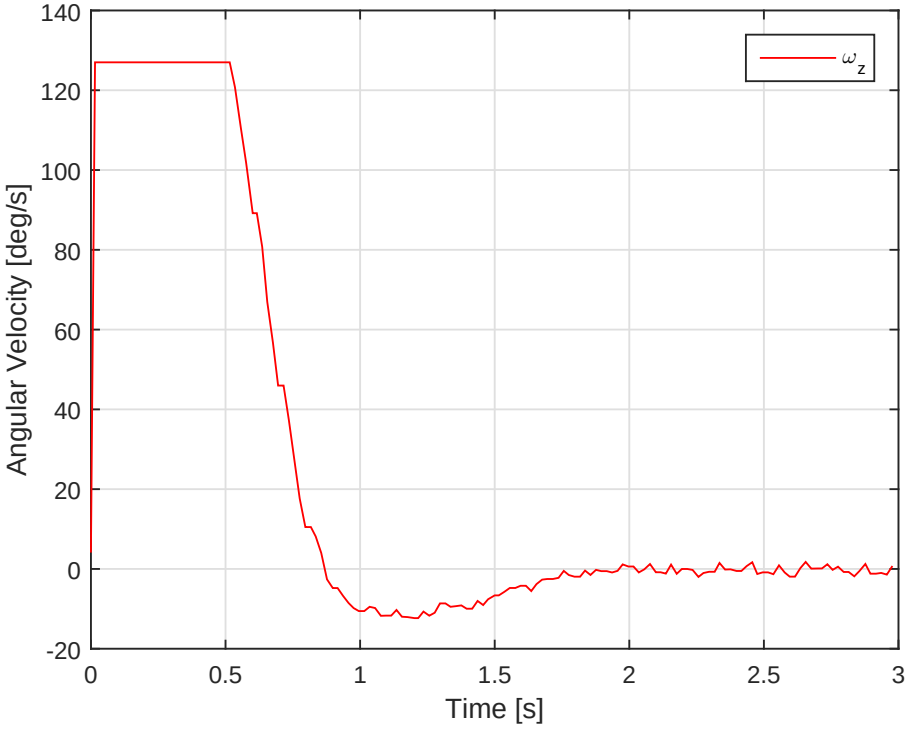
The trust input of the quadcopter has a scale between 0 and 255, corresponding to a range obtained with a total of 8 bits ($2^8 = 256$), that works as a ratio, without a clear physical unit. It was decided to conduct the tests using around 50% of the available thrust, which translated into an input of 125. Regarding the input in the cyclic actuators of the quadcopter, their total variation is the same as the collective actuator but now the range goes from -127 to 127. Again, this interval follows the principle of a scale, between the minimum and the maximum power allowed, that behaves as an angular velocity input, in degrees per second. Intuitively, this was the expectation while controlling the attitude with these actuators and boundaries. Additionally, an interesting discovery made during the tuning was that there is a dead zone on the cyclic inputs, which depends on its proportion in relation to the collective. For this choice of the collective, the dead zone was between around -20 and 20.

For the controller itself, the first attempt was to include only a proportional term, as during the simulated environment. The quadcopter's response showed a relatively constant static error, independent of the reference's amplitude. The initial idea was to add an integrative term to try compensating the error, but it only made the controller oscillatory. The fact that the magnitude of the error did not show a direct dependency on the reference was already an indication that it was not a usual steady state error, which actually led to the aforementioned discovery. Because of the dead zone, as the heading approached the goal, smaller and smaller inputs were required, reaching a point when this input was so small it fell inside that zone and the control action stopped. The behavior of the integrator made sense, because as it never reached the threshold, it kept accumulating until the input was out of the dead zone. But then that input was higher than it actually needed to be, so it pushed the heading too far, eventually leading to the same situation in the opposite side, which created a perpetual oscillation around the goal. In the end, the solution was to keep the P controller and set the gain high enough to fight the dead zone effectively, without bringing an unwanted behavior to the response.

The optimal response was obtained by setting $K_P = 5$, which can be observed in Figure 6.5a, together with the magnitude of the control action in Figure 6.5b. These show a very similar behavior to the simulated environment, maintaining a settling time below 2 s, a reduced overshoot, and an almost negligible static error. Therefore, the simulation provided a trustworthy representation of the dynamics involved in this process.



(a) Heading response.



(b) Control input.

Figure 6.5: Experimental output of the heading locking controlled system to a step input of $45°$.

6.3 Trajectory Tracking Controller

This new and more complete controller intends to track both the position and the heading of the vehicle relative to a structure, thus a more detailed model of the interactions involved in this task is required, for the positioning component. The trajectory of such a vehicle needs to be defined not only in position and velocity, to indicate where to move and how fast to get there, but also in attitude, given the way the vehicle generates motion is through angular offsets of the generated thrust. With this in mind, the simplified model

$$\begin{cases} \dot{\boldsymbol{p}} = \boldsymbol{v} \\ \dot{\boldsymbol{v}} = g\boldsymbol{e}_3 - \frac{T}{m}\boldsymbol{r}_3 \\ \dot{\boldsymbol{r}}_3 = -S(\boldsymbol{r}_3)\boldsymbol{R}^T \begin{bmatrix} \omega_x & \omega_y & 0 \end{bmatrix}^T \end{cases} \tag{6.3}$$

attempts to provide a description of the internal system of the vehicle, where $\boldsymbol{p}$ is the position, $\boldsymbol{v}$ is the linear velocity, $\boldsymbol{r}_3$ is the third column of the rotation matrix from $\{B\}$ to $\{I\}$, denoted here by $\boldsymbol{R}^T$ for simplicity of notation, T is the thrust input, m is the mass, and g is the gravitational acceleration. In this model, which represents the vehicle in $\{I\}$, the position is assumed to be the integral of the velocity and the velocity is obtained by reducing the application of Newton's Second Law to the gravitational force and the thrust in the vehicle's vertical axis, neglecting drag effects and the interaction with the angular velocity. Furthermore, to describe the attitude, only the third column of the rotation matrix was used, since it is the component that affects the linear dynamics. Using this quantity leaves room for the yaw motion to be independently controlled, in parallel with this strategy.

It is common, in control tasks, to define a desired state and an error variable, with respect to the real value of the state. For this case, a desired position $\boldsymbol{p}_d$ and velocity $\boldsymbol{v}_d$ were set, followed by the computation of the error dynamics

$$\begin{cases} \tilde{\boldsymbol{p}} = \boldsymbol{p} - \boldsymbol{p}_d \\ \tilde{\boldsymbol{v}} = \boldsymbol{v} - \boldsymbol{v}_d \\ \tilde{\boldsymbol{r}}_3 = \boldsymbol{r}_3 - \boldsymbol{r}_{3_d} \end{cases} \Leftrightarrow \begin{cases} \dot{\tilde{\boldsymbol{p}}} = \dot{\boldsymbol{p}} - \dot{\boldsymbol{p}}_d = \boldsymbol{v} - \boldsymbol{v}_d \\ \dot{\tilde{\boldsymbol{v}}} = \dot{\boldsymbol{v}} - \dot{\boldsymbol{v}}_d = g\boldsymbol{e}_3 - \frac{T}{m}\boldsymbol{r}_3 - \dot{\boldsymbol{v}}_d \\ \dot{\tilde{\boldsymbol{r}}}_3 = \dot{\boldsymbol{r}}_3 - \dot{\boldsymbol{r}}_{3_d} = -S(\boldsymbol{r}_3)\boldsymbol{R}^T \begin{bmatrix} \omega_x & \omega_y & 0 \end{bmatrix}^T - \dot{\boldsymbol{r}}_{3_d} \end{cases} \tag{6.4}$$

where $\boldsymbol{r}_{3_d}$ and T_d are the desired third column of $\boldsymbol{R}^T$ and thrust input respectively. A desired acceleration $\boldsymbol{a}_d = \dot{\boldsymbol{v}}_d$ at this moment and later a desired jerk $\boldsymbol{j}_d = \ddot{\boldsymbol{v}}_d$ also appear, both which were chosen as null for simplicity purposes.

The approach to stabilize this nonlinear system consists of briefly dividing it into two simpler subsystems, introducing a new state $\boldsymbol{x} = \begin{bmatrix} \tilde{\boldsymbol{p}} & \tilde{\boldsymbol{v}} \end{bmatrix}^T$ for the first. The idea is to take this subsystem and find a control law for the trust input that removes its dependence. This way, the stability of the complete system only relies on $\tilde{\boldsymbol{r}}_3$, the remaining state. For this task, it is assumed full access to the states, i.e. independence from the attitude estimation discussed in Chapter 3, and no saturation considerations are being made at this point. Defining T as

$$T = T_d \boldsymbol{r}_3^T \boldsymbol{r}_{3_d} \tag{6.5}$$

leads to the velocity dynamics

$$\begin{aligned}
\dot{\tilde{\boldsymbol{v}}} &= g\boldsymbol{e}_3 - \frac{T_d}{m}\left(\boldsymbol{r}_3\boldsymbol{r}_3^T - \boldsymbol{I}\right) - \frac{T_d}{m}\boldsymbol{r}_{3_d} - \dot{\boldsymbol{v}}_d \\
&= g\boldsymbol{e}_3 + \frac{T_d}{m}S(\boldsymbol{r}_3)^2\boldsymbol{r}_{3_d} - \frac{T_d}{m}\boldsymbol{r}_{3_d} - \dot{\boldsymbol{v}}_d \\
&= g\boldsymbol{e}_3 - \frac{T_d}{m}S(\boldsymbol{r}_3)^2\left[\boldsymbol{r}_3 - \boldsymbol{r}_{3_d}\right] - \frac{T_d}{m}\boldsymbol{r}_{3_d} - \dot{\boldsymbol{v}}_d \\
&= g\boldsymbol{e}_3 - \frac{T_d}{m}S(\boldsymbol{r}_3)^2\tilde{\boldsymbol{r}}_3 - \frac{T_d}{m}\boldsymbol{r}_{3_d} - \dot{\boldsymbol{v}}_d
\end{aligned} \tag{6.6}$$

The following step is to set $\boldsymbol{r}_{3_d}$ according to

$$\boldsymbol{r}_{3_d} = \frac{m}{T_d}\boldsymbol{F}, \text{ with } \boldsymbol{F} = g\boldsymbol{e}_3 - \dot{\boldsymbol{v}}_d + K_{\tilde{\boldsymbol{p}}}\tilde{\boldsymbol{p}} + K_{\tilde{\boldsymbol{v}}}\tilde{\boldsymbol{v}} \tag{6.7}$$

with the goal of eliminating the unnecessary terms from the $\tilde{\boldsymbol{v}}$ dynamics. In this equation, a vector $\boldsymbol{F}$ was used for simplicity of notation, where $K_{\tilde{\boldsymbol{p}}} > 0$ and $K_{\tilde{\boldsymbol{v}}} > 0$ are the controller's position and linear velocity gains respectively. Because it is a column of a rotation matrix, $\boldsymbol{r}_{3_d}$ needs to be a unitary vector. This requirement can be easily fulfilled by defining T_d as

$$T_d = m\|\boldsymbol{F}\| \tag{6.8}$$

With these changes, the first subsystem assumes the form $\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}(T_d, \tilde{\boldsymbol{r}}_3)$, where the kinematics matrix $\boldsymbol{A}$ and the input matrix $\boldsymbol{B}$ follow from the updated system. Similarly as before, the stability can be analyzed recurring to the Lyapunov stability theory. Building a quadratic Lyapunov function for this subsystem yields

$$V_1(\tilde{\boldsymbol{p}}, \tilde{\boldsymbol{v}}) = \boldsymbol{x}^T\boldsymbol{P}\boldsymbol{x} \tag{6.9}$$

where the symmetric weighting matrix $\boldsymbol{P}$ will be discussed shortly. The derivative of Equation (6.9) is

$$\begin{aligned}
\dot{V}_1(\tilde{\boldsymbol{p}}, \tilde{\boldsymbol{v}}) &= \boldsymbol{x}^T\boldsymbol{P}\dot{\boldsymbol{x}} + \dot{\boldsymbol{x}}^T\boldsymbol{P}\boldsymbol{x} \\
&= \boldsymbol{x}^T\left(\boldsymbol{P}\boldsymbol{A} + \boldsymbol{A}^T\boldsymbol{P}\right)\boldsymbol{x} + 2\boldsymbol{x}^T\boldsymbol{P}\boldsymbol{B}(T_d, \tilde{\boldsymbol{r}}_3) \\
&= -\boldsymbol{x}^T\boldsymbol{\Theta}\boldsymbol{x} - \frac{2T_d}{m}\left(\tilde{\boldsymbol{p}}^T\boldsymbol{P}_{12} + \tilde{\boldsymbol{v}}^T\boldsymbol{P}_{22}\right)S(\boldsymbol{r}_3)^2\tilde{\boldsymbol{r}}_3
\end{aligned} \tag{6.10}$$

where the Lyapunov Equation was applied, in the transition from the second to the third step, to introduce a new positive definite weighting matrix $\boldsymbol{\Theta}$. This matrix can be manipulated to tune the performance of the system, since it establishes a relative importance between the errors in position and velocity and in $\boldsymbol{r}_3$. In this function, the first remaining term is quadratic and negative. Thus, apart from the leftover term in $\tilde{\boldsymbol{r}}_3$, that will be compensated while analyzing the complete system, the function fulfills the Lyapunov requisites for asymptotic stability. At this point, transforming the kinematics of $\boldsymbol{r}_3$ to

$$\dot{\boldsymbol{r}}_3 = -S(\boldsymbol{r}_3)\boldsymbol{\Pi}_{\boldsymbol{r}_3}\overline{\boldsymbol{\omega}} = -S(\boldsymbol{r}_3)\left[-S(\boldsymbol{r}_3)^2\overline{\boldsymbol{\omega}}\right] = S(\boldsymbol{r}_3)^3\overline{\boldsymbol{\omega}} = -S(\boldsymbol{r}_3)\overline{\boldsymbol{\omega}} \tag{6.11}$$

can help with further mathematical developments, where $\overline{\boldsymbol{\omega}}$ is an alternative input. The derivative of $\boldsymbol{r}_{3_d}$ is also present at the $\tilde{\boldsymbol{r}}_3$ kinematics, which is given by

$$\dot{\boldsymbol{r}}_{3_d} = \frac{1}{\|F\|}\boldsymbol{\Pi}_{\boldsymbol{r}_{3_d}}\dot{\boldsymbol{F}} = -\frac{1}{\|F\|}S(\boldsymbol{r}_{3_d})^2\dot{\boldsymbol{F}}, \text{ with } \dot{\boldsymbol{F}} = -\ddot{\boldsymbol{v}}_d + K_{\tilde{\boldsymbol{p}}}\dot{\tilde{\boldsymbol{p}}} + K_{\tilde{\boldsymbol{v}}}\dot{\tilde{\boldsymbol{v}}} \tag{6.12}$$

Following a similar structure, a Lyapunov function for the remaining state can now be defined as

$$V_2(\tilde{\boldsymbol{r}}_3) = \frac{1}{2}\tilde{\boldsymbol{r}}_3^T\tilde{\boldsymbol{r}}_3 \tag{6.13}$$

where the only difference is that the weight is constant and equal to $\frac{1}{2}$. The derivative of the latest Lyapunov function is

$$\begin{aligned}
\dot{V}_2(\tilde{\boldsymbol{r}}_3) &= \tilde{\boldsymbol{r}}_3^T\dot{\tilde{\boldsymbol{r}}}_3 \\
&= \tilde{\boldsymbol{r}}_3^T\left(-S(\boldsymbol{r}_3)\overline{\boldsymbol{\omega}} + \frac{1}{\|F\|}S(\boldsymbol{r}_{3_d})^2\dot{\boldsymbol{F}}\right) \\
&= \tilde{\boldsymbol{r}}_3^T\left(-S(\boldsymbol{r}_3)\overline{\boldsymbol{\omega}} + \frac{1}{\|F\|}S((\boldsymbol{r}_3 - \boldsymbol{r}_{3_d}) + \boldsymbol{r}_{3_d})S(\boldsymbol{r}_{3_d})\dot{\boldsymbol{F}}\right) \\
&= \tilde{\boldsymbol{r}}_3^T S(\boldsymbol{r}_3)\left[-\overline{\boldsymbol{\omega}} + \frac{1}{\|F\|}S(\boldsymbol{r}_{3_d})\dot{\boldsymbol{F}}\right]
\end{aligned} \tag{6.14}$$

which depends on $\overline{\boldsymbol{\omega}}$, the bridge to the undefined inputs of the system. This variable can be set to compensate the necessary terms in each function, in order to ensure asymptotic stability.

Theorem 6.1. *Considering the system in Equation* (6.4)*, if the input* T *is set according to Equation* (6.5)*, the references* $\boldsymbol{r}_{3_d}$ *and* T_d *are given by Equations* (6.7) *and* (6.8) *respectively, and* $\overline{\boldsymbol{\omega}}$ *is defined as*

$$\overline{\boldsymbol{\omega}} = \frac{1}{\|\boldsymbol{F}\|}S(\boldsymbol{r}_{3_d})\dot{\boldsymbol{F}} - S(\boldsymbol{r}_3)\left[\frac{2T_d}{m}\left(\boldsymbol{P}_{12}\tilde{\boldsymbol{p}} + \boldsymbol{P}_{22}\tilde{\boldsymbol{v}}\right) + K_{\tilde{\boldsymbol{r}}_3}\tilde{\boldsymbol{r}}_3\right] \tag{6.15}$$

where $\boldsymbol{F}$ *and* $\dot{\boldsymbol{F}}$ *follow from Equations* (6.7) *and* (6.12) *respectively,* $\boldsymbol{P}_{12}$ *and* $\boldsymbol{P}_{22}$ *are constant design matrices, and* $K_{\tilde{\boldsymbol{r}}_3} > 0$ *is the controller's third state gain, then the closed loop system is asymptotically stable.*

Proof. Writing a joint Lyapunov function, resulting from the sum of Equations (6.9) and (6.13), yields

$$V_f(\tilde{\boldsymbol{p}}, \tilde{\boldsymbol{v}}, \tilde{\boldsymbol{r}}_3) = V_1(\tilde{\boldsymbol{p}}, \tilde{\boldsymbol{v}}) + V_2(\tilde{\boldsymbol{r}}_3) = \boldsymbol{x}^T\boldsymbol{P}\boldsymbol{x} + \frac{1}{2}\tilde{\boldsymbol{r}}_3^T\tilde{\boldsymbol{r}}_3 \tag{6.16}$$

Using Equation (6.15), the derivative of this function corresponds to

$$\dot{V}_f(\tilde{\boldsymbol{p}}, \tilde{\boldsymbol{v}}, \tilde{\boldsymbol{r}}_3) = -\boldsymbol{x}^T\boldsymbol{\Theta}\boldsymbol{x} - K_{\tilde{\boldsymbol{r}}_3}\|S(\boldsymbol{r}_3)\tilde{\boldsymbol{r}}_3\|^2 \tag{6.17}$$

which is composed of only negative definite terms, remembering that $\boldsymbol{\Theta}$ is a positive definite weighting matrix. Therefore, asymptotic stability is guaranteed, as originally intended. □

The constant design matrices $\boldsymbol{P}_{12}$ and $\boldsymbol{P}_{22}$ from Theorem 6.1 correspond to the blocks of $\boldsymbol{P}$ that are relevant to the control task. Using once again the Lyapunov Equation, they are given by

$$
\begin{aligned}
&\boldsymbol{P}\boldsymbol{A} + \boldsymbol{A}^T\boldsymbol{P} = -\boldsymbol{\Theta} \\
\Leftrightarrow &\begin{bmatrix} \boldsymbol{P}_{11} & \boldsymbol{P}_{12} \\ \boldsymbol{P}_{12} & \boldsymbol{P}_{22} \end{bmatrix} \begin{bmatrix} \boldsymbol{0} & \boldsymbol{I} \\ -K_{\tilde{\boldsymbol{p}}}\boldsymbol{I} & -K_{\tilde{\boldsymbol{v}}}\boldsymbol{I} \end{bmatrix} + \begin{bmatrix} \boldsymbol{0} & -K_{\tilde{\boldsymbol{p}}}\boldsymbol{I} \\ \boldsymbol{I} & -K_{\tilde{\boldsymbol{v}}}\boldsymbol{I} \end{bmatrix} \begin{bmatrix} \boldsymbol{P}_{11} & \boldsymbol{P}_{12} \\ \boldsymbol{P}_{12} & \boldsymbol{P}_{22} \end{bmatrix} = -f_\Theta \begin{bmatrix} \boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{I} \end{bmatrix} \\
\Leftrightarrow &\begin{cases} \boldsymbol{P}_{12} = \frac{f_\Theta}{2K_{\tilde{\boldsymbol{p}}}}\boldsymbol{I} \\ \boldsymbol{P}_{22} = \frac{f_\Theta}{2K_{\tilde{\boldsymbol{v}}}}\left(1 + \frac{1}{K_{\tilde{\boldsymbol{p}}}}\right)\boldsymbol{I} \end{cases}
\end{aligned}
\tag{6.18}
$$

where its symmetry property was used and f_Θ is a scaling factor. The input to the system dynamics can now be defined as

$$
\boldsymbol{u} = \begin{bmatrix} \omega_x & \omega_y & \omega_z & T \end{bmatrix}^T \tag{6.19}
$$

where $\begin{bmatrix} \omega_x & \omega_y & 0 \end{bmatrix}^T = -\boldsymbol{R}S(\boldsymbol{r}_3)^2\overline{\boldsymbol{\omega}}$, according to Equation (6.11), and ω_z can be computed with the controller developed in Section 6.2.

The saturation limits of the control signals will be the same used so far, although now the references should be given conservatively, to avoid any unwanted behavior. This means that the angular velocities will be restrained to the assumed boundaries of -127 and $127°$/s, as mentioned in Section 6.2.

Regarding the implementation of the new strategy in a simulated environment, it was necessary to fully redesign the internal mechanisms, to accommodate the added dynamics and establish the required control feedbacks. The block diagram for the new strategy is presented in Figure 6.6, where each separate section is highlighted to provide a better understanding of how each component connects with the next.

From the simulation built in Chapter 4, all the viewing tools were removed for the sake of simplicity, together with the quantities deemed unnecessary for this application. Starting with the System block group, it includes the well known Rotation Matrix Kinematics block and intends to replace the third equation of the system presented in Equation (6.3). The relevant components of its output are being fed to the added Quadcopter Dynamics block, which is not more than the first two equations of the same system. In the Sensor zone, the trajectory of the vehicle is no longer a known and constant piece of information, but rather the product of the dynamics of the system. The Processing block group keeps the same core, when compared with the initial simulation, whose outputs are delivered to the newly built Control zone. The latter is comprised of two controllers, the first from the previous section, where the difference is that the saturation block was relocated, from within the Heading Controller block, to act on all three angular velocity components later. The Position Controller block below receives the desired position and velocity of the vehicle and computes the control law for the angular velocities in the horizontal plane, designed according to Equation (5.1), and the trust, presented in Equation (6.5).

With this controller, the tuning process rests on choosing the best values for the gains $K_{\tilde{\boldsymbol{p}}}$, $K_{\tilde{\boldsymbol{v}}}$ and $K_{\tilde{\boldsymbol{r}}_3}$, which are basically weights of the error variables, and the factor f_Θ. The attitude subsystem, concerning $\boldsymbol{r}_3$, constitutes the inner loop of the system, therefore it has to be faster than the position subsystem. With that in mind, the parameters need to be chosen with the goal of keeping the errors in position and velocity small, with respect to the attitude. Considering $K_{\tilde{\boldsymbol{p}}} = 1.3$, $K_{\tilde{\boldsymbol{v}}} = 2$, $K_{\tilde{\boldsymbol{r}}_3} = 10$,

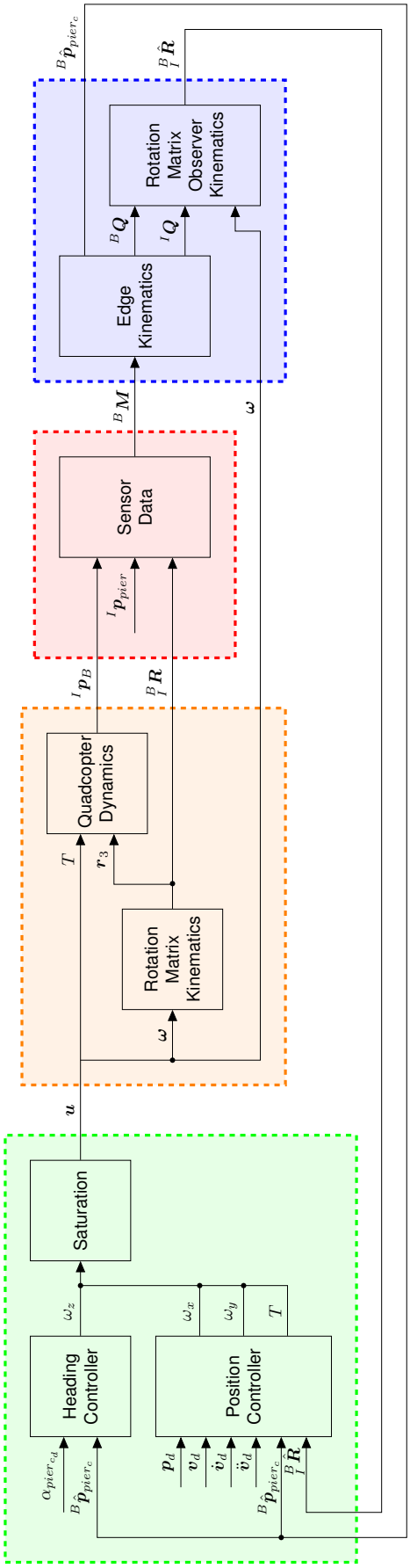


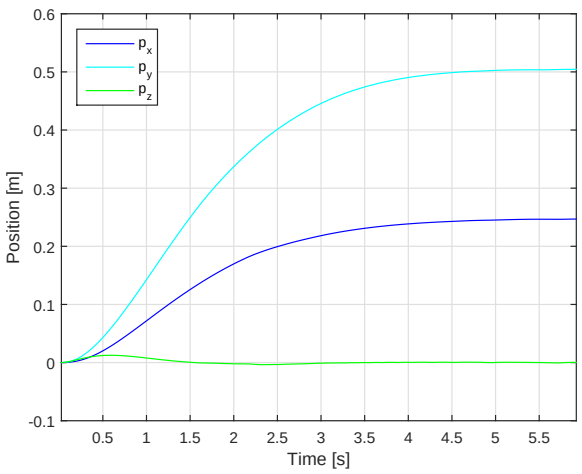
Figure 6.6: Block diagram of the trajectory tracking controlled system distinguishing the controller (green), the system (orange), the sensor (red), and the processing (blue).

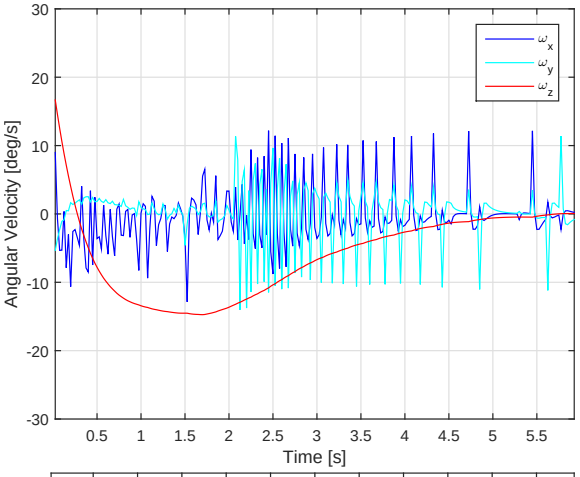
and $f_\Theta = 0.01$ accomplished that objective and yielded a swift response, with a settling time under 4 s and no overshoot or steady state error, as well as an acceptable control effort, shown in Figures 6.7a and 6.7b respectively. Additionally, Figure 6.7c presents an overview of the trajectory traveled by the vehicle, with the evolution of the position and heading combined, to offer a full realization of the control task. It should be noted that, in order to reject disturbances in the heading angle, the gain from the controller in Section 6.2 was retuned to $K_P = 2.3$.

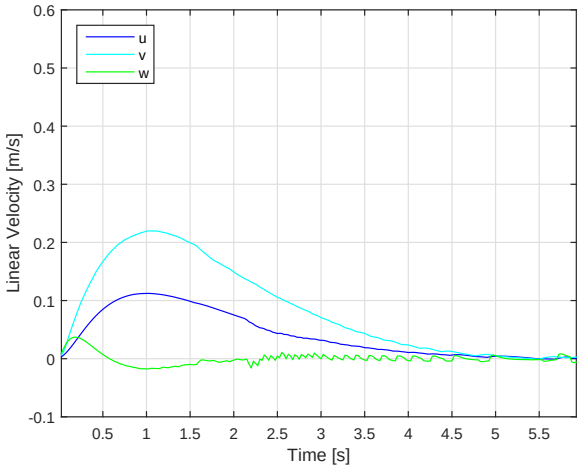
When it comes to the real setup, some upgrades had to be made to unlock the two additional DOFs, necessary for the proposed motion of the quadcopter. For this purpose, the aforementioned tray, to which the vehicle was strapped, was connected through a mechanical arm to a base, allowing it to perform a radial movement around that point, as well as moving vertically. Moreover, the tray now had a system of springs, providing the vehicle with a limited flexibility in roll and pitch and therefore allowing it to move in the horizontal plane. The last additional detail is the presence of a low drag wheel on the bottom of the tray, to easily maneuver the vehicle on low power settings. The latest version of the setup is presented in Figure 6.8.

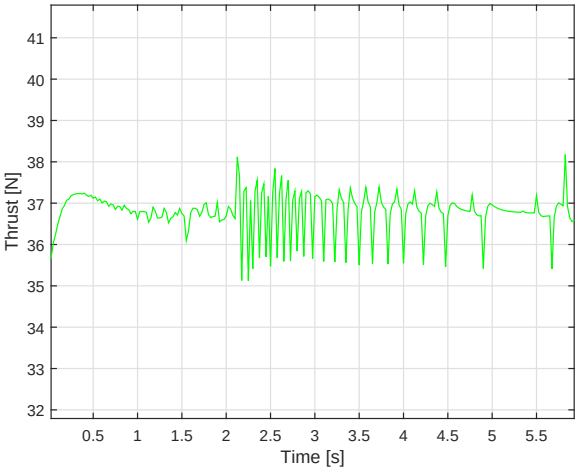
The GUI was also updated, as shown in Figure 6.9, and incorporates an additional graphic, showing the evolution of the vehicle's position, together with the necessary references and gains for this controller.

For the implementation of this controller in the quadcopter, the trials are currently in preparation, for a complete testing, and experimental data can be found in future works.

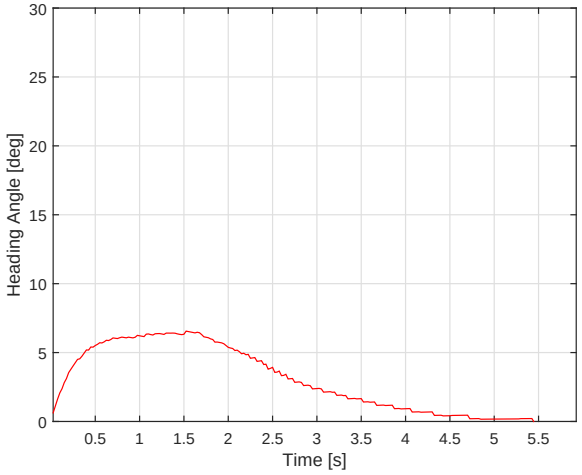




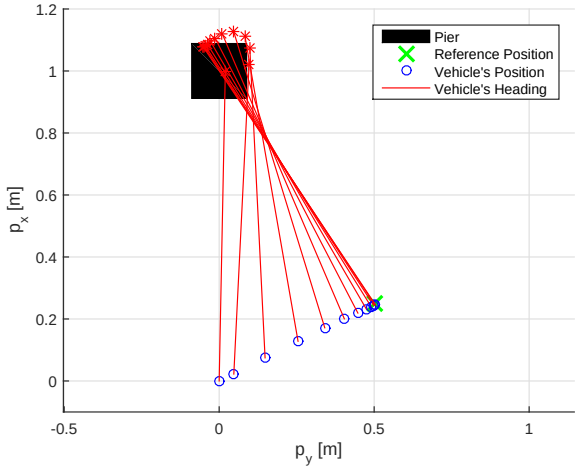




(b) Control input.

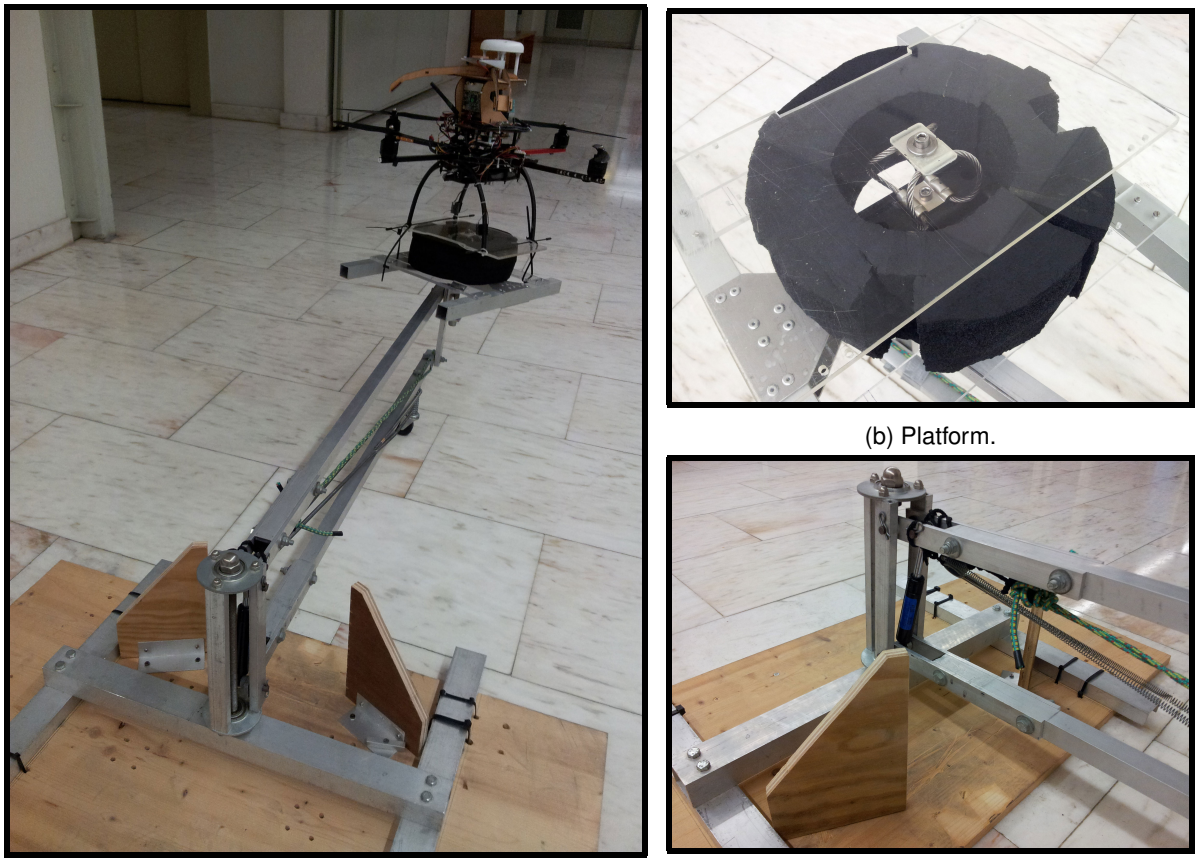


(a) Position, linear velocity, and heading response.



(c) Trajectory.

Figure 6.7: Simulated output of the trajectory tracking controlled system to a step input $\boldsymbol{p}_d = [0.25, 0.5, 0]^T$ m, $\boldsymbol{v}_d = [0, 0, 0]^T$ m/s, and $\alpha_{pier_{c_d}} = 0°$.



(a) Complete assemble.

(b) Platform.

(c) Base.

Figure 6.8: Experimental setup of the trajectory tracking controlled system.

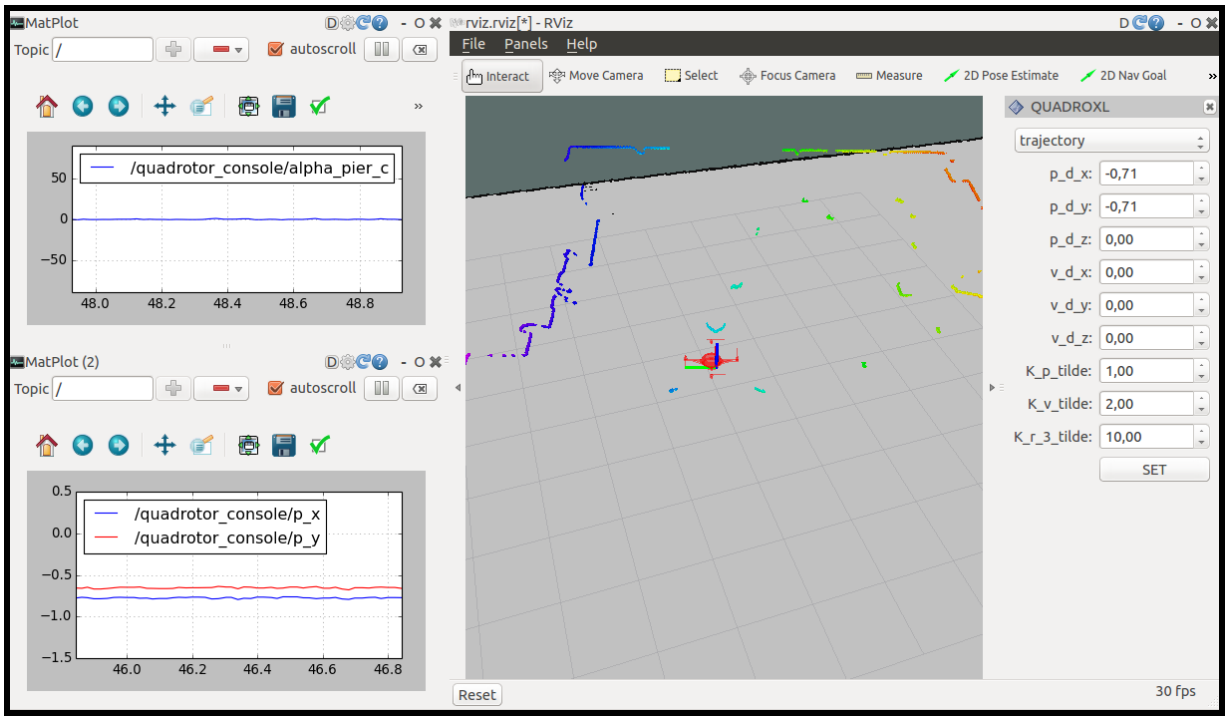


Figure 6.9: Experimental interface of the trajectory tracking controlled system.

Chapter 7

Conclusions

This thesis proposed a solution to the problem of laser-based control of rotary-wing UAVs, considering the entire process comprising the acquisition and treatment of the sensor's measurements, the development of methods to compute the relevant quantities to describe the motion of the vehicle, and the design and implementation of stable and effective controllers, based on that information.

The first chapter attempted at unveiling the status of UAVs in the world, briefly describing what these vehicles are and what can people hope to achieve with them, considering their on-board equipment and general characteristics. It was also mentioned how their stance in society is shifting towards more business oriented tasks, that aim at reducing the amount of risk and money spent on essential procedures, such as routine inspections. After exposing these motivations, a proposal was made regarding a relative positioning solution, based on the available sensors, with several necessary stages until reaching the established objectives.

Perceiving the environment was the topic of Chapter 2, where the work was laid out. First the reference frames were established, crucial to later relate the measurements across several descriptions, followed by an analysis of the sensor's precision, paramount to being able to obtain consistent results with the methods to be designed. There were some limitations regarding the accuracy of the LiDAR, depending both on the distance between the objects and the sensor and on the location, in its field of view, where they were being detected. More specifically, the errors were higher both when the object was at less than 2 m to the sensor and when it was detected in its 45° frontal cone. The geometry initially considered was a cylindrical section, with which the LiDAR could not provide data points accurate enough. This led to the second and final geometry, the rectangular section, where its straight edges, only varying in length, removed part of the unpredictability found in curvature of the previous shape, the circle. The outlier detection was one of the tasks with simultaneously the higher amount of uncertainty involved and future importance. The presence of outliers deeply affects the length of the edges being fitted, which is exactly the source from where the vehicle's attitude is being retrieved, with any of the methods developed. These undesired data points may arise from the sensor's noise, reflection of the laser beams or even lighting conditions, and are very unpredictable. The strategy to identify and remove them went through many stages, with every new experiment reveling new exceptions and requiring a threshold

adjustment, until the ideal trade-off to obtain robust algorithms would be found. The edge splitting and the outlier trimming were two deeply connected processes, so they needed to be carefully tuned together. Regarding the geometry fitting, both a least squares and a reduced space Hough transform algorithm were implemented, where the former led to a better ratio between quality of the results and computational effort than the latter. In the end of this chapter, the full description of the edges of the structure was built, in the appropriate reference frames. The representation in $\{I\}$ heavily depended on their constantly fluctuating length, due to outliers, therefore its calculation had to be carefully considered through an optimization, using the available properties of this geometry and backward coherent samples, in order to obtain smooth results.

The next stage was extracting the attitude from the structure's description, in Chapter 3, using different methodologies. At first, the idea was to provide the attitude regarding the horizontal plane, however solely using the LiDAR measurements yielded results not reliable enough to be applied in a continuous way, given the high variations observed. Taking advantage of their other properties, the attitude in the vertical plane was investigated and after redesigning the least squares fitting, a fast and thorough algorithm was created to describe the yaw motion, also merging data from the IMU at a high level. Nevertheless, fusing data from this sensor at a lower level enabled the pursuit of a full attitude description, by initially applying an existing solution for the Wahba's problem, on the group of rotation matrices, and later designing a nonlinear filter, aiming at an increased accuracy. The application of the Wahba's problem relied in a very straightforward implementation, where its main concerns were the fact that it considered each time step separately, possibly leading to a noisy response, and its need for at least two measurements at each moment, otherwise there would be a DOF unaccounted for and therefore an angle ambiguity in the attitude. On the other hand, the nonlinear observer made an estimation that evolved throughout time and was expected to yield smooth results. This method was also evaluated for its stability, with respect to the possibility of drift in the angular velocity being acquired from gyroscopes, where a bounded bias was seen to lead to a bounded error in the final product.

Chapter 4 was in charge of building a complete simulation environment, to test the previously developed methods, without the need to call upon real hardware right away. This simulation focused on modeling the vehicle dynamics with the rotation matrix kinematics. After these kinematics, the creation of the LiDAR's data points had to be mimicked, to simulate the internal functioning of the sensor, for which an algorithm was designed. This algorithm was much more efficient than other external alternatives for the same base purpose, the intersection of a 3D shape by a line or beam. The rest of the simulation was responsible for applying the procedures, discussed in Chapters 2 and 3, in the form of independent systems, which would be useful later for a real implementation, where modularity is key. Additionally, a visualization tool was added in parallel, to allow an extremely intuitive output of the whole process in a continuous way, featuring the vehicle with its translation and rotation represented, together with the acquired data points projected along the structure.

Two experiments were planned and presented in Chapter 5, with the intention of comparing the methods in equal circumstances, through several scenarios. Because the attitude in the horizontal plane maintains a strong dependence with the accuracy of the edge's lengths, the first was conducted in

null roll and pitch conditions, while the second considered reasonable values for these angles. This way, the impact of the sensor's noise and the effectiveness of the data treatment could be evaluated. Three of the methods presented in Chapter 3 were tested, both in a real and in a simulated environment. The yaw movement tracking presented a good performance in both environments, revealing an adequate adjustment to real data and a proper simulation at the same time. However, despite being accurate and having a low computational load, this method can only describe the motion in the vertical axis, which led to the creation of the full attitude estimation alternatives. Considering the application of the Wahba's problem, the results of the experiments corresponded to the expectation, by achieving a close but noisy description of the attitude, when compared to the reference from the IMU. When it comes to the simulation, because only the kinematics associated with the rotation matrix were being modeled and the accelerometer was not available, the concerns initially raised with this method were confirmed. The lack of sufficient measurements in the simulated environment, together with the artificial noise affecting the length of the edges, led to an unrealistic reproduction of the behavior observed during the experiments. The rotation matrix observer obtained the best results, where the attitude description obtained in the experiments can be seen as a filtered version of the previous method, maintaining a similar proximity to the reference. In the simulation, the absence of the accelerometer also degraded the response. However, it only limits the convergence in the horizontal plane's components of the attitude, since this method does not have the same restrictions regarding the minimum number of measurements as the Wahba's problem application. Therefore, the simulated environment presents a reasonable representation of the events carried out during the experiments. It should be noted that the oscillations in the last two methods, in the roll and pitch angles, are directly related to the uncertainties while determining the length of the edges. Moreover, all methods presented a yaw motion description very similar among themselves and to what was planned in reality, unlike what was obtained with the IMU. For implementation purposes, the last method shows the most promising behavior.

The final stage of this thesis is discussed in Chapter 6 and concerns the integration of the procedures in the vehicle and the design of controllers, set to achieve the initial goal of tracking a trajectory, defined relatively to a structure. First, the code developed in MATLAB needed to be rewritten in C, to allow the vehicle's CPU to run it directly. This conversion process consisted of adding new code to what already worked as a whole before, so it needed to have a compatible execution time and blend seamlessly with the pre-existent framework. After ensuring a successful code migration, a target locking controller was developed, with the objective of maintaining a certain heading to a pier. This control system was built based on a first order model of the yaw kinematics. A simulated version of the controlled system was designed, obtaining a prompt response to a step input, with no overshoot or steady state error, using a P controller. For the real implementation, both an experimental setup with a single DOF in yaw and a real-time controller interface for tuning were built. Because the vehicle had a dead zone, the gain of the P controller needed to be high enough to compensate it and reach a satisfactory response, revealing a great match between the simulation and the real system. The full position controller was then considered, where the simplification of the force balance that describes this system led to the error dynamics. The approach to stabilize this nonlinear system consisted of briefly separating it into two simpler subsys-

tems. This strategy was combined with knowledge derived from Lyapunov's control theory, in order to compute the inputs that guarantee the asymptotic stability of the system. The simulation of the new controlled system was implemented, requiring a deep restructuring of the working simulated environment. The missing step was to tune the parameters of the controller to obtain an optimal response. Reducing the magnitude of the errors in position and velocity with respect to the third state achieved that goal, yielding a swift response, with no overshoot or steady state error. After that, the experimental setup was upgraded to unlock the additional two DOFs, necessary for the proposed motion of the vehicle, and the interface was modified to accommodate the real-time tuning of the new parameters.

7.1 Future Work

Every stage yielded results that can be used on their own for a variety of other applications, given the modularity effort employed throughout this work. The most significant contributions from this work are the LiDAR data processing tools, including the geometry analysis and the data points treatment, the methods to compute the attitude from a limited source of measurements, such as the rotation matrix observer, and the trajectory tracking controller, guaranteeing the asymptotic stability of a nonlinear system. Given their performance in both experiments and simulations, they could be brought to real world applications.

The directions for future work focus on two fronts, the first being the completion of the experimental testing of the trajectory tracking controller, currently in its final stages of planning. Afterwards, there are the considerations concerning the improvement of the developed algorithms or the approaches themselves. A significant amount of the effort behind this thesis was put on treating the data from the LiDAR. This data, more than often, presented new challenges and required an adjustment of the methodology that had been planned so far. The use of more accurate LiDAR sensors, when they become available, may lead to more predictable observations and therefore to an easier and more reliable processing of the information. Besides that, there are many more strategies in place to deal with similar situations, from which only a selected few were tested within this scope. A proposal would be to investigate what other available methodologies have to offer, in terms of an increase in performance.

Although one of the main concerns, while testing every method, was to consider the a large number of scenarios to make them robust, there are always unforeseen cases that can eventually cause an exception in the routines. With the goal of making the transition of these controllers to real applications, further testing in even more adverse conditions may also be a point of interest. Furthermore, employing LiDAR sensors with a higher accuracy can also bring back the discussion on cylindrical structures and therefore open the solutions developed in this thesis to a wider range of possibilities. As mentioned in Section 2.3.1, the curvature of such structures makes it difficult to obtain meaningful results with the current sensors, so future technological advancements in this area may benefit those scenarios.

Bibliography

[1] *Unmanned aircraft systems (UAS)*. Cir 328, AN/190. ICAO, 2011.

[2] FAA. Unmanned aircraft systems (UAS) frequently asked questions. Retrieved 08/08/15, from http://www.faa.gov/uas/faq/#qn1.

[3] U. E. Franke. Civilian drones: Fixing an image problem? ISN Blog, ETH Zurich, January 2015.

[4] B. P. Tice. Unmanned aerial vehicles - the force multiplier of the 1990s. *Airpower Journal*, 1991.

[5] G. Goyer and R. Watson. The laser and its application to meteorology. *Bulletin of the American Meteorological Society*, 44(9):564–575, 1963.

[6] A. P. Cracknell. *Introduction to remote sensing*. CRC press, 2007.

[7] J. R. Sklaroff. Redundancy management technique for space shuttle computers. *IBM Journal of Research and Development*, 20(1):20–28, 1976.

[8] M. Atwater. Structural inspection reaches new heights while inspector stays grounded. Engineering.com, September 2013.

[9] B. J. N. Guerreiro. *Sensor-Based Control and Localization of Autonomous Vehicles in Unknown Environments*. PhD thesis, Instituto Superior Técnico, Universidade de Lisboa, 2013.

[10] M. Teixidó, T. Pallejà, D. Font, M. Tresanchez, J. Moreno, and J. Palacín. Two-dimensional radial laser scanning for circular marker detection and external mobile robot tracking. *Sensors*, 12(12): 16482–16497, 2012.

[11] R. Halır and J. Flusser. Numerically stable direct least squares fitting of ellipses. In *Proc. 6th International Conference in Central Europe on Computer Graphics and Visualization. WSCG*, volume 98, pages 125–132. Citeseer, 1998.

[12] A. Fitzgibbon, M. Pilu, and R. B. Fisher. Direct least squares fitting of ellipses. In *Pattern Recognition, 1996., Proceedings of the 13th International Conference on*, volume 1, pages 253–257. IEEE, 1996.

[13] V. Nguyen, A. Martinelli, N. Tomatis, and R. Siegwart. A comparison of line extraction algorithms using 2d laser rangefinder for indoor mobile robotics. In *Intelligent Robots and Systems, 2005. (IROS 2005). 2005 IEEE/RSJ International Conference on*, pages 1929–1934. IEEE, 2005.

[14] T. Pavlidis and S. L. Horowitz. Segmentation of plane curves. *IEEE transactions on Computers*, 23 (8):860–870, 1974.

[15] C. Leys, C. Ley, O. Klein, P. Bernard, and L. Licata. Detecting outliers: do not use standard deviation around the mean, use absolute deviation around the median. *Journal of Experimental Social Psychology*, 49(4):764–766, 2013.

[16] C. A. Groenwall and M. C. Millnert. Vehicle size and orientation estimation using geometric fitting. In *Aerospace/Defense Sensing, Simulation, and Controls*, pages 412–423. International Society for Optics and Photonics, 2001.

[17] L. Shapiro and G. C. Stockman. Computer vision. 2001. *ed: Prentice Hall*, 2001.

[18] J. Jensen. Hough transform for straight lines. *Miniproject in Image Processing*, 2007.

[19] F. L. Markley. Attitude determination using vector observations and the singular value decomposition. *The Journal of the Astronautical Sciences*, 36(3):245–258, 1988.

[20] R. M. Murray, Z. Li, S. S. Sastry, and S. S. Sastry. *A mathematical introduction to robotic manipulation*. CRC press, 1994.

[21] S. Brás, R. Cunha, C. J. Silvestre, and P. J. Oliveira. Nonlinear attitude observer based on range and inertial measurements. *Control Systems Technology, IEEE Transactions on*, 21(5):1889–1897, 2013.

[22] H. K. Khalil and J. Grizzle. *Nonlinear systems*, volume 3. Prentice hall New Jersey, 1996.