FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Visual-Inertial Based Autonomous Navigation of an Unmanned Aerial Vehicle in GPS-Denied Environments

Francisco de Babo Martins

FOR JURY EVALUATION

Mestrado Integrado em Engenharia Electrotécnica e de Computadores

Supervisor: Luís Teixeira

Co-Supervisor: Rui Nóbrega

June 29, 2015

Resumo

Nos últimos tempos, desenvolvimentos tecnológicos tais como controlo de estabilidade, autonomia de baterias e redes de sensores permitiram a criação de veículos aéreos não tripulados (UAVs) baratos e acessiveis a uma grande diversidade de aplicações.

Para UAVs, a navegação em interiores é um problema desafiante devido ao limite de carga que estes podem carregar. Devido a uma autonomia reduzida da bateria, apenas sensores com um peso reduzido, tal como câmeras monoculares, que não têm um impacto elevado na vida útil da bateria nem nas limitações de peso, podem ser anexados ao UAV. Estas câmaras no entanto apresentam um problema relativamente a navegação autónoma e estimação de posição, devido à ausência de percepção de profundidade, que só pode ser obtida com utilização de câmeras estereoscopicas e outros sensores adicionais. Esta dissertação aborda este problema através do desenvolvimento de um método destinado a permitir que um UAV possa navegar com segurança num ambiente previamente desconhecido, onde não existe cobertura de GPS, ao mesmo tempo que detecta as portas dos corredores e mantem o drone centrado nesse mesmo corredor.

Por outras palavras, o objectivo é o de produzir um sistema de navegação e estimação de posição robusto e autónomo para um veículo aéreo não tripulado.

A fim de conseguir cumprir este objectivo, o vídeo vindo em tempo real do UAV é processado por três módulos diferentes: navegação, detecção de portas e estimativa de posição. Para a parte da navegação, o sistema baseia-se na detecção do ponto de fuga da imagem capturada, utilizando a transformada de Hough para detectar e prevenir a colisão com as paredes. A parte de detecção de portas baseia-se não só na detecção dos contornos das portas, mas também na detecção das reentrâncias de cada porta. O sistema usa estes dois diferentes tipos de detecção a fim de detectar as portas com uma precisão mais elevada, utilizando a detecção de reentrâncias como o detector principal e o detector de contornos para uma validação adicional. Para a parte de estimativa de posição, o sistema baseia-se em informação pré-codificada do piso em que o UAV está a navegar, e na velocidade linear do UAV fornecida pela IMU do mesmo.

A fim de validar o método desenvolvido, os resultados desta abordagem foram obtidos e avaliados em diferentes corredores usando um quadricoptero, o Parrot AR. Drone. O drone foi capaz de navegar com segurança nesses mesmos corredores, detectando as portas do mesmo e estimando a sua posição aproximada relativamente ao seu ponto de partida. Isto prova que os métodos de navegação e detecção de portas desenvolvidos são robustos e permitem uma navegação autónoma de um veículo aéreo sem qualquer intervenção humana.

Abstract

In recent years, the latest developments in technologies such as stability control, batteries and sensing systems have allowed manufactures to produce Unmanned Aerial Vehicles (UAVs) both affordable and accessible to a vast diversity of applications.

For UAVs, indoor navigation is a challenging problem due to the limited payload they can carry. Because of a limited battery capacity, only light weight sensors such as monocular cameras, which do not take a toll on battery life and weight limitations, can be attached to the UAV. These cameras however present a problem towards autonomous navigation and position estimation due to the lack of depth perception which can only be obtained with vision disparity using stereo cameras and other additional sensors. This dissertation addresses this problem by developing a method intended to allow an UAV to safely navigate in a previously unknown and GPS-denied environment and focus on the detection of doors of corridors while keeping the drone aligned with the center of the same corridor.

In other words, the goal is to produce a robust and autonomous navigation and position estimation system for an unmanned vehicle.

In order to achieve this, the video input from a drone is processed by three different modules: navigation, door detection and drone position estimation. For the navigation part, the system relies on the detection of the vanishing point using the Hough transform for wall detection and avoidance. The door detection part relies not only on detection of the contours but also on the recesses of each door. The system uses these two different detection types in order to detect doors with a higher precision, using the recess detection as the master detector and the contour detection as an additional validation. For the position estimation part, the system relies on precoded information of the floor in which the drone is navigating, and the velocity of the drone provided by its IMU.

To validate the developed method, results of this approach were obtained and evaluated in different corridors using a quad-rotor drone, the Parrot AR. Drone. The drone was able to safely navigate in those corridors while detecting evident doors and estimate its approximate position relatively to its starting point. This shows that developed vision navigation and door detection procedures are reliable and enable an aerial vehicle to fly without the need of human intervention.

Acknowledgements

I would like to thank my supervisor Luís Teixeira and my co-supervisor Rui Nóbrega for their guidance and influential supervision throughout this project. I would also like to thanks my parents Agostinho e Felisbela for their constant support and faith throughout my master study. A special appreciation to all my friends and colleges for their enthusiasm and encouragement towards my project.

Francisco de Babo Martins

"Roads?
To where we are going, we don't need...roads!"

Dr. Emmett Lathrop Brown

Contents

List of Figures

List of Tables

Abbreviations and Symbols

UAV	*Unmanned Aerial Vehicle*
MAV	*Micro Air Vehicle*
GPS	*Global Positioning System*
IMU	*Inertial Measurement Unit*
HD	*High Definition*
FPS	*Frames per Second*
ROS	*Robot Operating System*
SURF	*Speeded-Up Robust Features*
SVM	*Support Vector Machines*
HOG	*Histogram of oriented gradients*
SIFT	*Scale-invariant feature transform*
SLAM	*Simultaneous Localization and Mapping*
SVM	*Support Vector Machines*
SLAM	*Simultaneous Localization and Mapping*
HOG	*Histograms of Oriented Gradients*

Chapter 1

Introduction

Nowadays, the technological advances in both hardware and software bridge the gap between science fiction stories and physical reality. Regarding robot technology, sophisticated hardware and software are coming together in order to create machines capable of accomplishing and replicating human activities. Today, UAVs are used for tasks such as surveillance, surveys, search and rescue operations[1], traversal of paths inaccessible to ground vehicles, quick deployment, and military applications which would be dangerous and hazardous for human beings.

However, this represents a certain problem due to some tasks being simple and latent knowledge for human beings, but being non-trivial when it comes down to replicating them on machines. Technology in this domain is still in need of certain advancements in order to achieve human like intelligence. However, sub-tasks which can someday be integrated into a system, are being researched.

Walking down a corridor while avoiding colliding with the walls and at the same time perceiving the structure of the corridor (doors and windows), is a trivial activity for any human being. This dissertation presents a solution that will enable an aerial vehicle to possess this kind of perception and dead reckoning.

1.1 Context

In recent years, Unmanned Aerial Vehicles (UAVs) have become affordable and relevant in several research areas such as military applications and surveillance systems. These reliable, small-sized and low-cost devices are normally equipped with high definition cameras and can be used as an autonomous image gathering device. The captured images can be processed in order to extract and obtain useful information that may be used for a variety of tasks and applications.

Being able to hover, fly laterally and at low speeds, makes UAVs an optimal platform to accomplish different military and civilian tasks such as reconnaissance support in hazardous zones, visual surveillance and inspection. In addition, some relevant industries are starting to use drones for other tasks beyond surveillance (e.g Amazon's "Prime Air"[1]). Moreover, the most important

[1] www.amazon.com/b?node=8037720011

task in order to achieve UAV autonomy is autonomous navigation. This may prove useful in a near future for tasks in indoor environments such as indoor transportation, object retrieval (e.g a missing part in an assembly line), monitoring misplaced books in a library and autonomously reporting sports event[2].

Although years of research of GPS position and data tracking have improved outdoor navigation and localization, in environments such as indoors or dense urban areas where maps are unavailable and the GPS signal is weak, an UAV will operate in high hazardous regions, running the risk of becoming lost and colliding with obstacles.

Since the scope of this project consists of enabling an UAV to autonomously navigate in an unknown environment without resorting to GPS localization, the main challenge is using visual odometry and on-board IMU to develop navigation and position estimation algorithms to achieve an autonomous and robust navigation. A vast majority of UAVs depend on GPS for navigation, hence this project is more challenging since GPS coverage is not available.

1.2 Research Questions

The main research question that is being addressed in this dissertation is:

How to enable an aerial vehicle to possess a collision-free navigation and a door detection system, allowing a precise and robust position estimation of both the vehicle and the doors in a floor plan?

From this question it is possible to define the main addressed topics. These will be the use of *collision-free navigation*, *door detection* and *position estimation*.

From this broader problem several research questions arise. First, is it feasible to create a navigation and wall collision application just by using the vanishing point from a scene as feedback? Secondly, can the door detection method recognize doors with different formats? Finally, is the position estimation of both the drone and the doors accurate?

For each research question there are several hypothesis that can possibly be presented as a solution for each problem.

Research Question 1 *Is it feasible to create a navigation and wall collision application just by using the vanishing point from a scene as feedback?*

There are few projects that require only monocular cameras to capture all the information needed to build a navigation algorithm. In many cases, sensors like sonar and laser range finders are used for this purpose. Even stereo cameras and RGB-D are used in this sort of application due to their main advantage: image disparity which is used for range estimation.

Therefore, the goal is then to propose a new method that, by using only a monocular camera and by detecting the vanishing point in a video feed, will grant a wall avoidance navigation to an UAV.

Research Question 2 *Can the door detection method recognize doors with different formats?*

Different types of doors are normally detected with different methods, either by training a classifier or by detecting the shape of the door. Some doors however are not entirely visible when navigation down a corridor as only a recess is certain to be observable.

The goal is to identify different doors in a corridor, being normal doors or recessed doors.

Research Question 3 *Is the position estimation of both the drone and the doors accurate?*

The main question is if it is possible to estimate the position of an UAV, using only the information provided by its sensor system, and the position of visually detected doors using both the estimated position of the UAV and the information provided by a pre-coded map containing metric data of a floor.

There are a few projects that explore floor mapping and simultaneous localization and mapping (SLAM) solutions for similar problems, but few set out to accomplish a robust position estimation system based solely on metric parameters from an aerial vehicle and detection of structural elements in the scene. The goal is to create a system which estimates the position of both a drone and the doors of a corridor using only the metric data provided by the onboard sensors of the drone and a generalised door detection system.

1.3 Objectives

In order to address the research questions, this project focused on exploring methods that may allow autonomous flight indoors without GPS while using only on-board sensors. Therefore it sets out to study and implement an automatic video analysis framework responsible for enabling an aerial vehicle to possess these abilities. The main novelty of the proposed solution to the stated problem is the proposal of a system that uses only a monocular camera to acquired all the necessary data to both the navigation, door detection and position estimations methods to be developed.

Given the complexity of the stated problem, this project needs to be split into simpler problems:

- **Acquisition and quantification of sensory data:** Processing and evaluating the values from the UAV sensors is of the utmost importance. The readings from the sensors will prove to be a valuable asset when the development of the autonomous navigation algorithm requires sensory data to analyse the state of the UAV.

- **Adjustment of the UAV navigation parameters:** In order to create a navigation algorithm, a fully functional communication between the UAVs and the external processing unit (laptop) must be established. Prior to the creation of the navigation algorithm, it must ensured

that the laptop is fully capable of sending desirable commands to the UAV. These include adjustments to parameters such as yaw, pitch and roll.

- **Image data acquisition from the UAV camera:** All the image frames captured by the UAV camera will be used to compute all the necessary data for the navigation algorithm.
- **Image processing:** Images previously acquired should undergo a series of processes aimed at detecting points and regions of interest which are relevant to the detection of features in the indoor environment and the UAV navigation. Such processes may include: Binarization, Segmentation, Noise Reduction and Perspective Transformations.
- **Image feature detection and extraction:** All the acquired frames from the camera should be analysed in order to detect features of interest to be used on the classification stage. The techniques to be used may include:
  - Canny edge detection[3];
  - Hough transform[4];
  - SURF[5];

The desirable framework will be implemented using typical image processing methods commonly found in the literature. This framework will have a modular architecture, meaning that each module will work independently from one another. The gist of this approach is that of breaking new ground for modular image processing based drone navigation systems in which additional modules can be developed and added to the framework, creating an ever evolving system with several new capabilities. The base framework is going to be composed of the following stages:

- **Image Acquisition** through a monocular camera on-board an aerial vehicle;
- **Navigation Control** based only on the current position of the vanishing point of the scene;
- **Detection of structural features in corridors** such as doors;
- **Position Estimation** of both the drone and the doors by cross-referencing their estimated position with metric data of the floor plant;

In order to achieve this goal, the system needs to meet the following requirements:

- The computational complexity of the system should be independent of the environment size.
- Firm and robust door detection.
- Fast data processing to satisfy real time constrains.
- Brisk and Flexible navigation;

This project aspires to be the stepping stone towards an automatic and autonomous vision navigation and position estimation system with an image acquisition based on UAVs. The final product will be a complete and working framework built upon common methods paving the way to future works which may improve the performance of the system.

1.4 Contributions

Bellow are listed the various contributions that resulted from all the work developed on the scope of this dissertation:

- Development of a computer vision framework for autonomous navigation and position estimation in corridors.
- Study and implementation of a vanishing point and door detection methods: the former for navigation purposes and the latter for position estimation purposes.
- The corridors in which the tests were conducted and the videos were recorded had their dimensions, including the distance between its doors, measured and annotated, therefore creating a small dataset of the metric information from corridors.
- An article about this project and all the research done was written and submitted to *ROBOT'2015: Second Iberian Robotics Conference*[2], in particular, the special issue "Visual Perception for Autonomous Robots".

1.5 Summary and Document Structure

This project sets out to develop a modular algorithm in order to generate confident results regarding autonomous navigation and position estimation in indoor environments.

After this introductory chapter, a literature review and an analysis of other solutions for autonomous navigation will be presented on chapter 2.

In chapter 3, an overview of the framework of the system is presented and the major problems it faced are identified.

Chapter 4 touches upon the several stages of the framework and preliminary results and output examples are also presented for each stage.

In chapter 5, results from several simulations and practical experiments are presented and discussed in order to evaluate the robustness of the system.

Finally, chapter 6 concludes this dissertation by discussing the work developed, presenting the main conclusions, the answers to the research questions and discussing possible future work.

[2]https://web.fe.up.pt/ robot2015/index.php/special-sessions

Chapter 2

State of the art

This chapter will focus on the existing work on the field of computer vision systems for autonomous navigation and will present a literature review which is befitting for a better understanding of the problem under study.

Section 2.1 will focus on currently used algorithms for image processing, machine learning and navigation that are suitable methods for solving the problem this project set out to solve.

In section 2.2 data collected from researching quad-copters and a more profound analysis of the one that was found to best meet the requirements of this project.

Finally, in section 2.3, a brief description of the current development state of autonomous navigation using UAVs will be presented.

2.1 Computer Vision and Image Processing Methods and Algorithms

2.1.1 Image Acquisition

The first stage of a computer vision system, and one of the most important, is without a doubt the image acquisition phase. Architectures based on image acquisition normally differ from one another depending on the type of camera that is used (see section 2.2.1, 2.2.1 and 2.2.1) and the amount of cameras used[6, 7].

2.1.2 Pre-Processing

Following image acquisition, the necessary step that comes after is image pre-processing. This can significantly increase the reliability of an optical inspection.

An example of a pre-processing method is video stabilization which is often present in a computer vision system when there is a need to compensate undesired camera motion[8, 9]. This is of the utmost importance when spatial image coherence is required.

Several filter operations which intensify or reduce certain image details enable an easier or faster evaluation of the scene. Filtering is colloquially referred to as smoothing or blurring as is commonly used in order to reduce noise in an image.

The most used filters are linear, meaning that the value of an output pixel is determined as a weighted sum of input pixel values:

- **Normalized Box Filter:** Each pixel will have an output that corresponds to the mean of its kernel neighbours.
- **Gaussian Filter:** This type of filtering is accomplished by convolving each point in the input image with a Gaussian kernel and then summing them all to produce the output image.
- **Median Filter:** This filter runs through each element of an image and replaces each pixel with the median of the pixels from its neighbourhood.

Another type of filtering consists of morphological transformations which consist of a set of operations that process images based on shapes. These morphological operations apply a structuring element to an input image and generate an output image.

The most commonly used Morphological operations are:

- **Dilation:** Computes a convolution of an image with a kernel which can have any shape or size (normally a square or circle).
- **Erosion:** Similar to dilation. Compute a local minimum over the area of the kernel.
- **Opening:** Obtained by the erosion of an image followed by a dilation (normally used to remove small objects).
- **Closing:** Obtained by the dilation of an image followed by an erosion (normally used to remove small holes and dark regions).

2.1.3 Image Processing

Image processing is a field of imaging science in which the output of image processing results in yet other image or a set of parameters and characteristics related to the image.

Commonly, the first step after capturing an image is that of converting it into a greyscale image and then into a binary image, or changing its color space from RGB to HSV.

Concerning the the identification of connected components, a segmentation and contour detection is used and in order to reduce computational costs, objects smaller than a specific area are normally discarded.

Segmentation

Dividing an image in different regions of interest which possess common features is the process known as image segmentation. The most commonly used techniques are the watershed algorithm[10] and grabcut[11] for background subtraction.

Object Detection

The gist of object detection is the precise recognition of an object in a set of images. This is a very complex problem and nowadays still proves to be quite challenging[12]. Although the most used approach to detect objects in a video stream consists of using information from different frames, it delivers results with a high error rate. A typical workaround is that of analysing temporary information which was computed from a set of sequential frames, hence reducing the error rate[12].

Edge Detection

In image processing, an edge is defined as a discontinuity in rightness of an image. Therefore the problem of detecting edges consists of finding those discontinuities which may represent changes in properties of that same image such as depth, orientation and illumination.

The *Canny edge detector*[3] is one of the most used edge detection operators and is comprised of the following steps:

- Noise removal with Gaussian filer.
- Localization of the intensity gradients of the image.
- Non-maximum suppression application to eliminate false responses to edge detection.
- Double threshold application to resolve potential edges.
- Suppress the remaining edges that are week and are not connected to strong edges.

Feature Detection

The SURF algorithm[5] is a robust local feature detector built upon the SIFT detector and descriptor but is reported to perform better than the latter[13] when it comes down to speed and robustness to viewpoint and illumination changes.

This algorithm detects salient features and provides their image coordinates together with their descriptions. It is composed of 4 stages:

- Integral image generation
- Interest point detection
- Descriptor orientation assignment
- Descriptor generation

In a navigation algorithm the calculation of the Speeded up Robust Features may prove to be the most intensive and computationally demanding part, so it needs to be accelerated in order to obtain a real-time performance.

Object Tracking

Object tracking consists in the process of tracking an object in a sequence of frames or images. Its typical applications are in surveillance tasks and interaction systems.

Some typical challenges that object tracking tries to overcome and which directly influences its success are the following:

- The movement of the object.
- A change in the pattern, structure and background of the object.
- Camera movement.
- Object occlusion by other object or background.

The major algorithms used in object tracking are Point Tracking, Kernel Tracking and Silhouette Tracking[14].

When it comes down to object tracking the concept of **optical flow** is often used. Being the apparent motion between an observer and the environment, it is used to estimate the motion field. This is accomplished by estimating the location of each pixel from one frame in the second frame. However it does not always correspond to actual motion.

One of the major challenges in calculating optical flow is that of finding corresponding pixels as many of them can have the same color and even surrounding pixels with the same color value. This leads to the assumption that in order to calculate the optical flow, it is assumed that there is very little movement from one image to the other, leading to a "travelled distance" of just one pixel. This assumption poses a problem when dealing with fast moving vehicles due to delays in sending and interpretation images.

One possible workaround is that of using sparse optical flow. This is a process in which image features are tracked across different images[15].

Some navigation methods used on UAVs are based on a dead-reckoning system, which is the process of calculating the current position based on a previously determined one. These methods are used to estimate the speed of an aerial vehicle based on the optical flow in the image of its bottom camera. The value of these speeds are then integrated in order to get an estimate of the drone position[1].

Feature Extraction

One of the most used feature extraction techniques is the **Hough Transform**. It was originally designed to find lines in a image[16] but it has been modified in order to detect shapes such as circles and ellipses[17].

Many variants of the standard Hough Transform were developed, such as the Generic Hough Transform with Gradient, 2-1 Hough Transform and Fast Hough Transform[18].

Prior to shape detection, an image must undergo a preprocessing in order to acquire information of the edge of each object. Techniques such as those mentioned in 2.1.3 are used to compute this data which will later be processed using the Hough Transform to find a shape.

In order to find a line, which is represented by

$$y = mx + c \tag{2.1}$$

the first thing to do is to represent a line into a point, resulting in the following equation:

$$c = -mx + y \tag{2.2}$$

The Hough Transform converts a point in xy-space to a line in mc-space (the parameter space), which consists of representing the information of an image discrete points $(x_1, y_1)(x_2, y_2)$ in terms of the slope parameter m and the intercept parameter c. As the points from an edge of an image are therefore transformed into lines, some lines that are constructed from edges of the same object will intersect. This means that the intersection of lines in in mc-space correspond to information of a line in xy-space.

However, a vertical line proves to be a problem as m has an infinite value, requiring infinite memory to store the data in mc-space. Therefore, another representation is used to avoid this problem.

In this new representation, a line is represented by the distance p of a line from the point (0,0) and the angle θ of a line to the x-axis. Thus, a line is represented by the following equation:

$$p = x.cos(\theta) + y.sin(\theta) \tag{2.3}$$

Where x and y is a point passed by the line.

In this method, a line in xy-space will be transformed into a point in $p\theta$-space and a point in xy-space will be transformed into a sinusoidal curve[19].

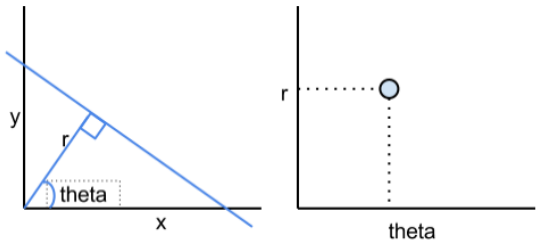


Figure 2.1: Representation of a line

Using this method, the Hough Transform uses the values of x, y and R to recognize circles in a image. The Hough Transform will create circles with a certain radius in every pixel of an object and the calculation of the points with the maximum number of interceptions of those circles will

be computed. In the end, the point with the most interceptions will acknowledge as the center point.

The Hough transform has proven to be a strong candidate for this project as it has proven to be efficient and quite reliable in some other similar projects[20].

A team from the University of Indonesia used the Hough Transform method[18] to detect and track shapes in an indoor environment using the front-facing camera of the AR. Drone. They accomplish this by following these steps:

- Conversion of the image color model from RGB to HSV.
- Conversion of the video stream into a binary image: since this team detected objects based on its color and shape, they had to pay close attention to the range of the HSV components in order to obtain the ideal threshold.
- Following, using an implementation of the Hough Transform method from the OpenCV library, a circle shape is found.
- The tracking method then uses the information obtained in the previous step and sets a rectangle window inside the viewfinder of the input image. After that, the controller method uses this information and makes sure that the rectangle always contains the information of the desirable object to track.

2.1.4 Machine Learning

Machine learning is a sub-field of artificial intelligence that enables computers to act independently without being programmed to do so. Many applications that rely on machine learning are, for example, self-driving cars, practical speech recognition and effective web search. This is accomplished by constructing algorithms that are able to learn from available data.

There are different methods that can be categorized as follows:

- Supervised learning which use both the inputs and the outputs in order to replicate an intended model:
  - Parametric algorithms.
  - Non-Parametric algorithms.
  - Support Vector Machines (SVM).
  - Kernels.
  - Neural networks.
- Unsupervised learning, which are used when there is no information regarding the intended output:
  - Clustering.

- Dimensionality Reduction.
- Recommender Systems.
- Deep learning.

Support Vector Machines

In order to enable an UAV to possess the ability to recognize objects, a machine learning method that is considered to be of relevant use is Support Vector Machines (SVM).

SVMs are a kernel-based technique that aim to output an optimal separating hyperplane that categorizes previously labelled images. They are mostly used in pattern recognition tasks and require a training set of images in order to train the classifier. Its modus operandi is finding the hyperplane that maximizes the margin of the training data. This means that the optimal separating hyperplane gives the largest minimum distance to the training data.

SVMs have a fairly satisfactory success rate towards redundant attributes, have a reliable accuracy and classify data rather quickly.

2.2 Unmanned Aerial Vehicles (UAVS)

In recent years, UAVs such as quad-copters have increasingly gained interest in robotics and computer vision research. Commonly used as a flying camera, they are mostly employed for surveillance tasks and recording humanly impossible videos. Therefore, in order to navigate safely, these equipments rely on their on-board sensors and cameras to track their position autonomously. The advantages of using UAVs includes their easy manoeuvrability and ability to hover.

Whilst there has been substantial research about navigation algorithms on ground vehicles, these approaches can not be directly transferred to aerial vehicles:

- An indoor aerial vehicle cannot hold as many sensors as a ground vehicle.
- A flying vehicle has more degrees of freedom than a ground vehicle, which prevents the use of 2D navigation algorithms.
- The risk of failure is more catastrophic in a flying vehicle. If a navigation prediction failure occurs on a ground vehicle, it can be stopped by a user, whereas when the same occurs in a flying vehicle, it falls on the ground damaging its structural integrity.

It is of the important to enable an UAV to possess collision avoidance capabilities to avoid the risk of damaging the structure of the vehicle and the surrounding environment. This catastrophic scenario depends mostly on the altitude of the vehicle and the nature of the collision.

2.2.1 On-board Electronics

Different UAVs have an array of devices capable of providing accurate data which can prove to be useful for navigation, localization, landing and obstacle avoidance tasks.

GPS

The Global Positioning System [21] was funded by the United States Department of Defence and was initially designed for the United States military, although in 1980, it was made available for civilian use.

This system works anywhere in the world, 24 hours a day and in any weather conditions. Four GPS satellite signals are used in order to compute the positions in three dimension and also the time offset in the receiver clock.

IMU

The IMU is a single unit, that houses two sensors, which collects angular velocity and linear acceleration data and returns it to the main processor. It is used to acquire pitch, roll and yaw data from the UAV.

A brief description of these two sensors follows:

- **Accelerometer:** An accelerometer is a device that measures acceleration forces which may be static (the force of gravity), or dynamic (vibrating or even moving the accelerometer itself).

- **Gyroscope:** A Gyroscope is a device that measures the orientation of a device based on the angular momentum of that same device. It is used to acquire the angular rate of a certain vehicle[22].

Laser Range Finder

This sensor uses a laser beam to determine the distance of the source of emission to an object. Its normal mode of operation consists of sending a laser pulse to a certain object and measuring the distance that it takes the pulse to reflect off the object and bounce back to the emitter. They are commonly used for 3D-Modelling[23].

Ultrasonic Sensor

This sensor can be used as a sonar based altimeter, providing altitude estimation and vertical displacements. Their modus operandi is similar to a sonar which consists of generating high frequency sound waves and measuring the time that takes for the transmitter to received the echo of the emitted wave, thus determining the distance to an object.

Pressure Sensor

A pressure sensor is commonly used to provide stability to an UAV, allowing the on-board processing to automatically correct and maintain a still position while the vehicle is airborne. This is achieved regardless of altitude and wind intensity[1]. On-board pressure sensors provide unique

stability that will automatically correct and maintain a still position in the air regardless of altitude and wind intensity

Magnetometer

A Magnetometer is an electronic device capable of measuring the strength and direction of a magnetic field. Therefore it can be used in navigation applications by measuring the earth's magnetic field.

Monocular Camera

Even though this type of cameras have low weight, power consumption, size and cost, they still provide an immense amount of information, which is unmatched by any other type of sensor. Compared to other depth measuring devices, the range of a monocular camera is virtually unlimited. This feature proves to be useful when using a monocular SLAM system to operate in open environments[24].

In spite of those advantages, this device proves to be inefficient when it comes to determine the scale of the environment as there is no way to obtain depth perception. In this case, a vehicle must rely on another set of sensors such as those of the IMU (2.2.1).

Stereo Camera

This particular camera, containing two or more lenses with a image sensor per lens, is a direct solution to the disadvantage of the aforementioned monocular camera: extraction of depth information of an object in a image. It is possible to capture 3D images with these cameras due to the fact that the lenses simulate the human binocular vision. Hence, this process is called *stereo photography*.

Not all stereo cameras are used to acquire 3D photographs or video whereas depending on the lenses configuration, the acquired data may not contain stereoscopic information. For example a twin-lens reflex camera uses one lens to image to a focusing/composition screen and the other to capture the image on film (these are usually in a vertical configuration).

RGB-D Sensor

The RGB-D sensor provides the combined data of RGB color information and per-pixel depth information. The data acquired from these sensors can be represented as a *point cloud* which consists of a collection of points in three dimensional space. Additional features can be associated with the point from that space, meaning that color can be one of those features.

One device that provides these functionalities and is affordable, unlike the rest of the RGB-D sensors in the market, is the Microsoft Kinect[25].

2.2.2 Quad-copters

In recent years, UAVs have been used in several research projects and applications due to their simple manoeuvrability, mechanical simplicity and several improvements that have been made in technologies such as: stability, control, batteries and sensors.

Recent developments in drone technology have spawned different types of drones. The most common ones are listed bellow.

- **Flapping wing drones**
- **Fixed wing drones**
- **Rotor based drones**

Due to their low weight, small size and agility, flapping wing drones drones have the advantage of being able to fly and hover in a close proximity to objects [26] [27]. Fixed wing drones use a gliding mechanism: either built in linear propellers or an external propulsion mechanism. Unlike flapping wing drones, these are less agile but on the other hand have longer flight durations which makes then ideal candidates for survey and mapping applications [28]. The flight control of Rotor based drones is based on regulating the speed of the rotors, providing the drone with an ample degree of freedom during its flight. These drones normally have a larger payload due to carrying additional sensors as well as on-board cameras. These cameras receive visual signals that are used in several research areas such as obstacle avoidance, tracking and feature detection. However, camera based methods rely strongly on external light sources needed to illuminate the environment and on edge and texture features. Also, these methods may be limited in low or irregular visibility situations based on illumination.

Nowadays there are several available options in UAVs with an array of sensors and feedback hardware. For this project a quad-copter small enough to be used indoors was required. This vehicle would allow agile and responsive flight and effortless maintenance. Also, only quad-copters with frontal monocular cameras were considered due to the need of processing a video feed been captured in the front of the drone.

A large variety of drones is available in the market now more than ever. However, one quad-copter drone stands out as the most affordable, easy to use and to repair: the Parrot AR. Drone[1].

When compared to other drones such as the Dji Phantom[2] or the AscTec Firefly[3], the Parrot AR. Drone presents itself as the ideal solution for this and any other vision oriented project[2] due to several factors:

- **Low price**.
- **Open communications protocol:** allowing the development of autonomous flight projects with the help of a HD camera with high quality and transmission rate.

[1] ardrone2.parrot.com
[2] www.dji.com
[3] http://www.asctec.de/uav-applications/research/products/asctec-firefly/

- **Repairability:** Despite having a reduced robustness, the AR. Drone is fully repairable which is a characteristic of paramount interest since it will be used as a testing platform. Therefore, falls and impacts is certainly not avoidable.

2.2.3 Parrot AR. Drone

The Parrot AR. Drone (Figure 2.2) was initially conceived as a toy and therefore is quite popular and affordable.

Table 2.1: Technical Specifications of the Parrot AR. Drone 2.0

Structure	Carbon Fiber and Outdoor Hull
Weight	400 g
Autonomy	18 min
Sensors	Gyroscope,Accelerometer,Magnetometer,Pressure Sensor, Ultassounds, Vertical Camera for ground velocity
Communications	Wi-Fi
Camera	720p 30fps. Low latency Wi-Fi Transmission
Observations	Fully Repairable. Open and documented communications protocol
Price	300 euros

Being an off-the-shelf drone makes it easy to maintain and replace parts as well as providing specific features of on-board stabilization and accessible control design that makes it possible to focus on developing efficient software solutions without worrying about development for hardware.

A list of the most useful characteristics that come factory-fitted with this drone are shown on the list below.

- **Front and bottom facing cameras:** The Parrot AR. Drone comes equipped with a front facing camera and a bottom facing camera that provide live video streaming to the device controlling the drone. The front facing camera is used for piloting the drone while the bottom facing camera is used to see what is below the drone, horizontal stabilization and velocity estimation.
- **Automatic stabilization:** This drone provides an exceptional on-board stabilization system that makes use of the rotors, gyroscope and bottom facing camera for that purpose.
- **Wireless connectivity:** Wireless devices can easily connect to the AR. Drone by either using the official application provided by Parrot (both for iOS and Android) or the corresponding drivers such as the ardrone_autonomy (see chapter 3.4.3)

Quad-copter flight control

The control and manoeuvrability of a drone is directly influenced by its design. Such design consists if four rotor blades attached to a main body that is arranged in structure resembling 2x2

Figure 2.2: The AR. Drone quad-copter with the protection hull attached

matrix. In this structure each pair of rotors turn in the same direction meaning that, as seen in Figure 2.3, the rotor pair 1,4 and 2,3 turn in opposite directions: the former turns counter clockwise and the latter turns clockwise. This configuration is what enables the drone to possess the ability to hover in one place.

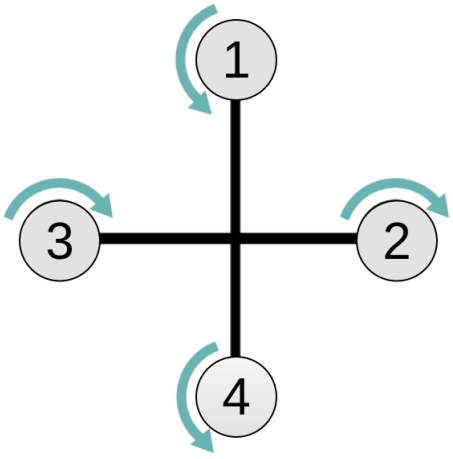


Figure 2.3: Rotation Direction of each of the four rotors found in a quad-copter

In order to move, the drone relies on a three-dimensional tilt/rotation system around the X, Y ad Z axis using differential torque and thrust amongst the rotors. As illustrated in Figure 2.4, rotation along each of the three axes allows the drone to move forward/backwards (pitch), left/right (roll) and turn left/right (yaw).

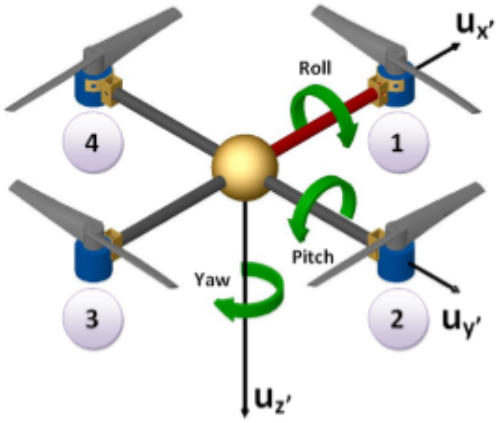


Figure 2.4: The rotation system on a quad-copter: yaw, pitch and roll

In order for a quad-copter to achieve linear movement, the rotation speed of two corresponding rotors varies relatively to the other two rotors. A quad-copter has the following degree of movement:

- **Forward/backward linear movement:** A differential speed between the front and the rear rotors achieves this movement (+/- pitch).
- **Up/down linear movement:** Applying the same speed on all four rotors achieves this movement. This means that lower speed lowers the drone and higher speed increases the hover hight (+/- height).
- **Left/right linear movement:** A differential speed between the right and the left rotors achieves this movement (+/- roll).
- **Left/right rotation movement:** A differential speed between the diagonal rotors achieves this movement (+/- yaw): the rotor pair 1,3 has a different torque than the rotor pair 2,4.

The speed and the direction of movement of the drone can be directly influenced by the degree of ration around either axes and the speed of each rotor.

Inertial Measurement Unit (IMU)

The Parrot AR. Drone contains an Inertial Measurement Unit (IMU) with six degrees of freedom that are measured using the following components:

- A 3-axis accelerometer that is used to measure acceleration int the X,Y and Z axes.
- A gyroscope used to measure,in degrees per second, roll and pitch (2-axis) and yaw (1-axis) by angular velocity.

In addition, the height estimation, stabilization and vertical speed of the AR. Drone is done by an Ultrasound Altimeter that is attached to the bottom of the drone. This altimeter is able to estimate the current height of the drone from a relatively flat surface bellow the drone by transmitting ultrasonic waves and "listening" to their echo on said surface.

On-board Processing

Due to the on-board processing software that comes pre-installed in the AR. Drone being closed-source and not publicly documented by Parrot, it was not modified throughout the fulfilment of this project. This software is responsible, among other things, for flight stability which requires the use of all the sensory and image data in order to maintain the position,awareness and state estimation of the drone.

Since it was designed as a toy, this drone has limited computational power which may prove a problem as image processing using computer vision algorithms require a lot of resources. One possible workaround is using a computer that is connected to the AR. Drone via wireless. The communication flow between the AR. Drone and the computer can be seen in figure 2.5.

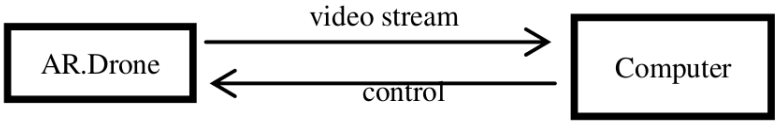


Figure 2.5: Data exchange between the AR.Drone and a Computer

The batteries of this UAV provide enough energy for a 15 minutes flight. However, its turbines are not powerful enough to keep a steady position in windy conditions, which limits its outdoor use.

All the documentation for the AR. Drone was made available by Parrot on their official website[4].

2.3 Related Work

The goal of developing autonomous UAVs is one of allowing them to perform complex manoeuvres and navigate independently in structured and unstructured environments.

2.3.1 Laser Assisted Navigation

A relevant project was that of a team from the Massachusetts Institute of Technology which used a laser range-finder sensor to enable an MAV to autonomously explore and map unstructured and unknown environments[29]. They used on-board sensors to estimate the position of the MAV and used SLAM algorithms to build a map of the environment around the vehicle.

[4] ardrone2.parrot.com/

The motion estimation of this project comprised of different problems: motion estimation using laser scan-matching and height estimation using the on-board IMU.

2.3.2 Sonar Assisted Navigation

Sonar technology used to be prevalent among UAVs and some projects still use it for field mapping using UAVs [30]. However, due to the fact that the time it takes for the sonar waves to travel decreases the UAV obstacle detection range, this technology can no longer be used.

The alternative is a way of detection images and avoid obstacles much sooner which can be achieve using a technology that does not depend on propagation waves, but rather uses light as medium to detect objects: optical cameras.

2.3.3 Camera Assisted Navigation

Pre-coded Maps

A team from the Czech Technical University in Prague developed a navigation system for a UAV using a monocular camera and odometry[31]. The navigation system was based on dead-reckoning techniques[32] to calculate the traversed segment length and the information from the camera to calculate the UAV yaw and vertical speed. This allowed the team to use a histogram voting scheme.

The developed navigation method consists of a "record and play" technique in which the UAVs is guided along the path that it will autonomously navigate later on. During the initial phase, a map is created by guiding the UAV along a path, using the front-facing camera to recognize and track salient features using the SURF[5] feature detector. In the second phase the UAVs navigates autonomously in the learned path by using the dead-reckoning system to calculate the traversed distance and retrieves the relevant landmarks from the precoded map in order to compare to the data currently being obtained through the camera. The difference of coordinates is then computed using a histogram voting technique and this information is then processed to the UAV yaw and altitude controllers to safely adjust the drone position. Briefly, using this method, the UAV builds a SURF-based landmark map in a guided tour and then uses the method described above to navigate autonomously in the mapped environment.

The only limitation of this project consists in the necessity of using a pre-autonomous-navigation phase in which the UAV is human-guided.

Autonomous Indoor Navigation

The implemented project is the spiritual successor of a project of a team from University of Texas [33] which aimed at navigating autonomously in indoor environments (corridors) and industrial environments (production lines) and detecting and avoiding obstacles (people). The navigation was accomplished by using the vanishing point algorithm, the Hough transform (2.1.3) for wall detection and avoidance and HOG descriptors using SVM classifiers for detecting pedestrians. The UAV used was the AR. Drone.

Firstly, edge detection is performed using the Canny edge detector followed by a Gaussian blurring to reduce noise and irrelevant edges. Using the points exported by the Canny edge detector, the Hough transform is used to detect the perspective lines using a voting scheme procedure which finds the lines that are closer to those same points. The following operation was that of removing some lines that were dependent on thresholds that influenced the Canny edge detector and the Hough transform (outliers). This was achieved by thresholding their angles (in radians).

The next phase was finding the line intersection of the lines which were detected by the Hough transform, which was achieved by using the Least square problem which estimates the minimum distance of a point from a set of lines. In corridors, such as those used in this project, the line interception is called the vanishing point. In this step there is a boundary box in the image used to distinguish between valid estimations of the vanishing point and invalid ones. If there is an invalid estimation, it is overlooked and the system awaits for another one.

As the AR. Drone lacks a laser or even an obstacle avoidance sensor, only the camera can be used to avoid collisions with the right and left wall. In this project it was assumed that the UAV should have the ability to know its relative position to the left and the right walls in order to avoid them while moving. The way they did it was by manually pre-estimating this ability which consisted of manually putting the camera of the drone to the right and left wall and analysing the angular value of the lines obtained from the Hough transform. It is important to mention that the angular values of the lines can be automatically estimated while the drone is at the center of a corridor. By exporting the value of the angles shown in figure 2.6 it was possible to determine when there was enough space for the UAV to safely navigate without hitting the side walls:

- When angle angle $B = D$, the drone is positioned at the center of the corridor.
- When the angle A is twice bigger than the the angle $B + D$, the AR-Drone has enough space to navigate without hitting any obstacles on the side walls or even the side walls themselves.

Figure 2.6: Hough Lines in the left and right wall

The human detection process implemented for this project used the Histogram of Oriented Gradient descriptors (HOG) and a linear SVM classifier (2.1.4). The SVM classifier was used to test the feature detector in the MIT pedestrian database which contains 1800 pedestrians images of different poses and backgrounds.

In order to avoid hitting the target (person), the boundaries of the side walls are considered a place of safe passage, meaning that when a person is detected by the AR. Drone, an estimation of the most large free sideways are is made in order to avoid hitting the moving target. The wall

detection algorithm is of the utmost importance as the position of the moving target is computed relatively to the boundaries of the side walls.

The main difference between this project the implemented project is that the latter sets out to improve on the navigation an wall avoidance algorithm and provide a feature detector to the drone. The original project only uses the information of the vanishing point to avoid wall collisions while the implemented project also uses this information to detect the end of a corridor as well as an obstacle blocking the way for the drone. Additionally, the implemented project provides video stabilization to compensate for abrupt movements and detects structural features of a corridor (doors) in order to estimate the current position of the drone.

Stereo and SLAM

Other approaches involved using stereo cameras[34] and the Microsoft Kinect RGB-D sensors[35] to solve the SLAM problem and find the best solution for a robust state estimation and control methods. These approaches perform local estimation of the vehicle position and build a 3D model of the environment (SLAM) to plan trajectories through an environment.

2.3.4 IMU Assisted Navigation

Some other projects recurred to Kalman filters and ultrasound sensors to compute the UAV pose estimation in a GPS denied environment[36]. Every time that the UAV has to reach a certain location, a path which provides a favourable observation density is computed from a predefined map, minimizing the risk of localization failures.

2.4 Summary

Recently, UAVs, more importantly quad-rotor drones, have become more affordable and consequently, available to wider range of applications and research. These vehicles, now equipped with on-board HD cameras are suitable for capturing images and recording videos in areas or situations that may or may not be possible for human beings. Their portability, effortless manoeuvrability and potential for autonomous flight have made them a image capturing low cost solution for research and consequently this project.

Several image processing techniques useful for this kind of project are available and some of them were used in the implementation of the developed solution for this project. The image processing techniques used are the *Hough Transform*, *Canny Edge Detection* and *Optical Flow*.

Chapter 3

System Overview

This chapter presents an overview of the system, explaining how the video feed is acquired and processed. The framework created for this project will be described succinctly.

The proposed method consists of a modular algorithm designed using image processing methods. The result of the algorithm is a generated map in which the position of the drone and the doors of a corridor are estimated and displayed.

3.1 Video Acquisition

The video feed used in this project comes from a 1280x720 (720p) resolution camera on-board the AR. Drone. The video is recorded at 30 frames per second (fps) in real-time while the drone navigates inside corridors (Figure 3.1) of the Department of Electrical and Computer Engineering[1] of the Faculty of Engineering of the University of Porto[2].

Figure 3.1: Drone navigating inside a corridor

[1] http://paginas.fe.up.pt/ deecsite/
[2] https://sigarra.up.pt/feup/

3.2 Video Processing

The analysis of the video feed is performed remotely in real-time using the OpenCV library for C++ (see section 3.4).

In the several stages of this process, different methods are used. The following sections explain how the developed framework and each of its modules operate and cooperate, with regard to input and output, as well as the main reason for using them.

3.2.1 System Framework

In this dissertation, a detailed framework was created to tackle the challenge of autonomous navigation and position estimation using a monocular camera. A graphical representation of the algorithm is shown on Figure 3.2.

This framework is devised into two parts:

- **Hardware Side:** The vehicle and all the data it provides (image and sensory data)
- **Software Side:** All the image processing methods.

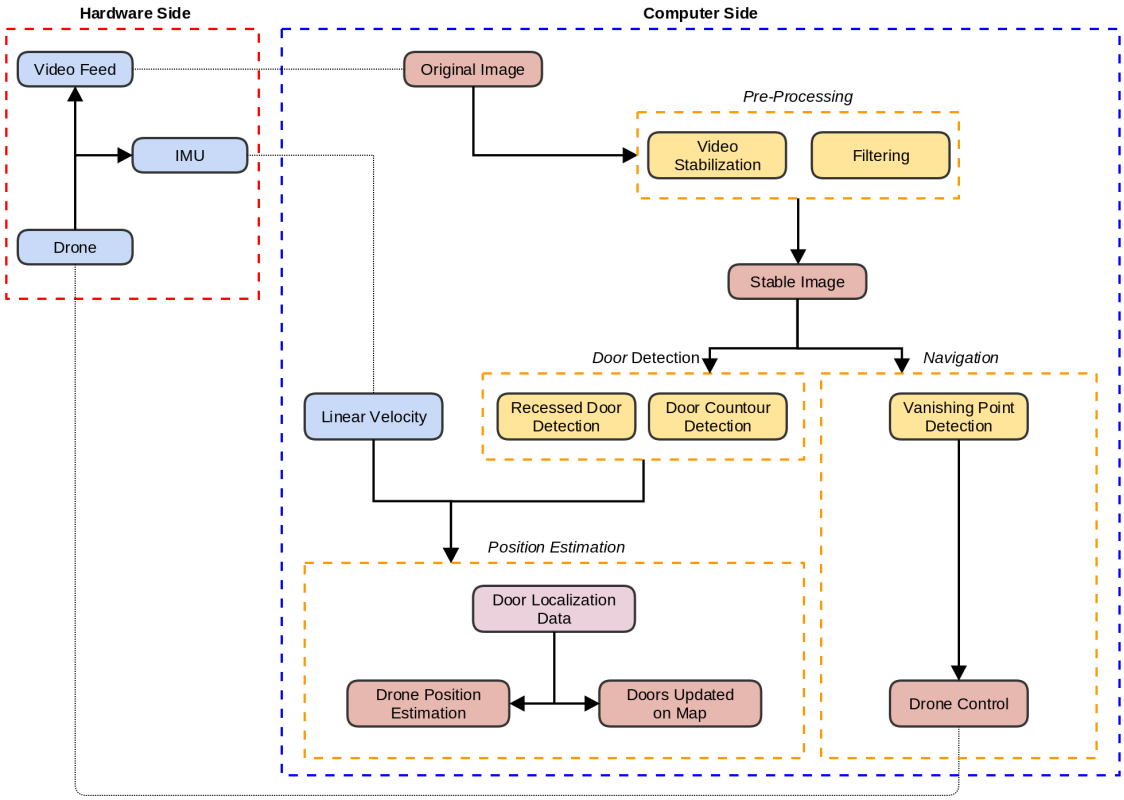


Figure 3.2: The developed framework which bridges the gap between the hardware of the drone and the external processing unit using an Wifi connection

In order to reduce the on-board payload data processing, the drone sends the acquired data to an external processing unit.

Regarding the hardware side, the drone mechanism and stability, the camera input and the medium through which the video feed is transmitted are considered. On the computer side a series of algorithms and methods that process the video feed and output the drone commands are present.

Pre-Processing

In order to extract useful information from the provided video feed, a series of steps are implemented. Beyond any doubt, the drone used for navigating in the corridors will be susceptible to oscillations caused by either external and uncontrollable factors (e.g different airflows) as well as internal factors such as its own dynamic. For this reason, the first module of the framework is image stabilization and filtering:

- Video stabilization will be used to compensate for abrupt camera shifts and oscillations;
- Filtering will be responsible for smoothing the frames and reducing noise.
- In order to speed up the operations and reduce the processing load on the computer running the software, each frame will be downscaled to a lower resolution. This operation by itself will also contribute to lowering the noise and unwanted features in the captured scene.

Door Detection

The door detection module will be responsible for detecting the doors in the video feed provided by the drone. Two different approaches will be used in order to develop a robust door detection system. These two methods will use a master/slave modus operandi, one being the detector that provides a better detection accuracy (*recessed door detection*) and the other the one (*door contour detection*) which although does not provided such a high detection accuracy, detects doors using an entirely different method. The conjugation of these two distinct detections will provide an even higher door detection accuracy.

Navigation

The navigation module works as follows:

- First, it acquires the location of the vanishing point (detected in the video feed provided by the drone);
- Then it creates adjustments that will correct the course of the drone.
- Finally, it sends these commands to the Drone.

A successful traversal of a corridor will be achieved if the drone can lock on the location of the vanishing point and always keep its orientation aligned with it.

Position Estimation

This module will use a set of data from different sources in order to estimate the position of the drone and the doors that are part of the corridor.

- Pre-coded metric information about the floor in which the drone is navigating.
- The linear velocity of the drone.
- The output of the door detection module.

3.3 Methods Evaluation

Having in mind the nature of the problem that this project aims to solve, the various steps of each method will be evaluated using quantitative and qualitative criteria:

- Quantitative Evaluation:
  - Vanishing Point Detection
  - Recessed Door Detection
  - Door Contour Detection
- Qualitative Evaluation
  - Video Stabilization

3.3.1 Quantitative Evaluation

"Vanishing Point Detection", "Recessed Door Detection" and "Door Contour Detection" are the framework stages in which a metric evaluation can be used.

Vanishing Point Detection

The metric used to evaluate the vanishing point detection method will be based on the mean value of all the detections that were made during each test flight.

For every test flight a manual annotation of the ideal vanishing point detected in each frame was performed. As this point will have similar coordinates during the flight it is expected that as more points are detected, they will be near the average value of all the points detected so far.

Therefore, using the manually detected vanishing point data, in each flight, the vanishing point detection method will prove to be robust if the each new detection does not deviate from the mean.

Door Detection

Both door detection methods, *Recessed Door Detection* and *Door Contour Detection* are evaluated by simply comparing the instant of time at which this module reports a door detection with the instant of time at which the drone actually passes next to a door.

Simply put, if the drone passes near a door and the method reports a door detection, and this situation happens for every door in the corridor, then this will prove the robustness of the method.

3.3.2 Qualitative Evaluation

Video Stabilization

In the field of computer vision the quantitative evaluation of video stabilization is not easy and trivial. Due to this fact, a qualitative evaluation using subjective criteria was performed based on long term accuracy and efficiency.

3.3.3 Test Scenarios

In order to test the developed methods and their accuracy and robustness in corridors, a series of videos were initially recorded with a 720p camera. The purpose was to cover different trajectories and circumstances which an aerial vehicle would encounter in those same corridors. The plan was to initially test the image processing stage of the framework and then, after developing the drone navigation method, compared the results from the simulation with those obtainable from various test flights in the same test environments.

The following three different corridors, all located on the electronics department at FEUP, were used to test the framework:

- **Corridor 1[3]:** This corridor is located on the ground floor. The trajectory starts at the northern end of the corridor.

Figure 3.3: Beginning, middle and end of corridor number 1

- **Corridor 2[4]:** This corridor is located on the ground floor and is the same as corridor number one but the trajectory starts at the southern end of this corridor.

[3] https://youtu.be/znE0zcuQgqc
[4] https://youtu.be/uUZ6uwh6lLI

Figure 3.4: Beginning, middle and end of corridor number 2

- **Corridor 3**[5]**:** This corridor is located on the second floor. The trajectory stars at the northern end of the corridor.

Figure 3.5: Beginning, middle and end of corridor number 3

3.4 Tools

A variety of tools were used during the implementation of this project and for latter experiments. The Parrot AR. Drone was used to test the developed framework, the OpenCV in C++ was the core processing library, and ROS with the ardrone_autonomy driver to bridge the communication between the drone and the computer.

3.4.1 Robot Operating System(ROS)

The experiments conducted throughout this project and presented in this dissertation required the use of the Robot Operation System (ROS) software framework as a link between the robot used in this project, the Parrot AR. Drone, and the computer where ROS is installed and running the developed algorithm.

ROS was originally developed in 2007 by the Stanford Artificial Laboratory as an operating system for a variety of robots and hardware [37]. Several services are provided such as hardware abstraction, process management, message passing, packet management and implementation of commonly used functionalities for devices. This operating system is released under terms of BSD license [6] and is freely available as a open source software for commercial as well as research use.

The usage of this software was influenced by the availability of drivers and support for the Parrot AR. Drone. While ROS provides several features for message passing, monitoring and

[5] https://youtu.be/xQC2eku8FcI
[6] http://www.linfo.org/bsdlicense.html

visualization tools, only the message passing system and the node framework is used to bridge the communication gap between the drone and the computer.

3.4.2 OpenCV

OpenCV is a famous computer library which provides tools focused around real-time computer vision applications[38]. This library was officially launched in 1999 as an Intel Research Initiative and made available initially in C programming language and later in C++ and Python programming language as a cross platform suite. OpenCV possesses a BSD license, being available for both commercial and research use as an open source solution.

OpenCV offers several state of the art implementations of useful computer vision algorithms and applications. The functionalities provided by this open source library used in this project are the following:

- Image processing (*Hough Transform*, *Canny edge detector*, etc)
- Video analysis
- 2D features framework
- Kalman Filters

This library also provides detailed information and various tutorials on its website[7]. OpenCV is the library of choice for researchers which makes it a suitable choice due to the fact that it is freely available and vastly used with a plethora of official[8] and unofficial forums.

3.4.3 AR. Drone Driver for ROS

In order to establish data exchange and communication between the AR. Drone and the computer where the developed method is running, a third-party driver custom made for this drone was used. The reason for using a third-party driver rather than of a built-in driver is due to the fact that the development team behind ROS has yet to build one. The Ardrone_autonomy[9] is a ROS driver based on the official AR. Drone SDK 2.0 [39], developed by Mani Monajjemi from the Simon Fraser University[10] among other contributors. This driver package also possesses a plenitude of tutorials and documentation available on several official and unofficial forums.

For this project, the most useful control parameters provided by this driver are the following:

- Front camera;
- Take-off and landing;

[7] http://docs.opencv.org/
[8] http://answers.opencv.org/questions/
[9] https://github.com/AutonomyLab/ardrone_autonomy
[10] http://sfu.ca/ mmonajje

- Flat trim: a service that calibrates the drone based on rotation estimates assuming that the drone is on a flat surface;
- Linear velocity along the X,Y and Z axes;
- Linear acceleration along the X, Y and Z axes;
- Three-dimensional rotation along the X (left and right tilt), Y (forward and backward tilt) and Z (orientation) axes.

All these parameters allow a favourable control over the drone from the software side of the project which will be running as the pilot of the drone.

3.5 Summary

Having in mind the purpose of this project, a straightforward approach was chosen in order to process the video feed from a vehicle traversing a corridor and respond with appropriate manoeuvres.

An important aspect worth mentioning is that the approach to the original problem was to create and develop a modular framework, meaning that each module works independently from one another. This also means that modules can be added and removed to and from this framework without conditioning the overall performance of the system. This paves the way for future improvements for this project.

The chapters that follow will describe and shed a light on the various stages of the framework as well as the various challenges that emerged and the subsequent methods proposed to deal with them.

Chapter 4

Vision Based Autonomous Navigation and Position Estimation

In this chapter the methods developed to bring the designed framework to life will be presented and explained. Some intermediary results will also be presented. As most indoor environments satisfy the Manhattan World assumption [40], i.e., most planes lie in one of three mutually orthogonal orientations, all the work developed made use of the this assumption. In other words, this work is within the constrained space of Manhattan-world scenes (scenes consisting predominantly of piece-wise planar surfaces with dominant directions).

4.1 Video Stabilization

When there is an undesired motion in a video feed, the need for video stabilization arises. In 1995, Canon created the first stabilization system[1] based on moving lens with a 16-bit microcomputer controlling an ultrasonic motor. This approach proved to be quite expensive and not suitable for the majority of cameras. For this reason, digital image stabilization is used instead, and it is widely used for image registration methods[41] and the development of computational resources[42].

Essentially, this system requires camera stabilization in order to reduce strong oscillations that might occur when a vehicle is traversing a corridor.

4.1.1 Proposed Method

The aim of the image stabilization method developed for this project was providing useful stabilization for a forward-moving and panning video stream.

The first step is to find the transformation from the previous to the current frame. This transformation is basically a rigid Euclidean transform and is achieved by using optical flow for all frames.

[1] cpn.canon-europe.com/content/education/infobank/lenses/image_stabilisation.do

In order to detect corner feature points for optical flow tracking, the OpenCV implementation of "goodFeaturesToTrack"[2] was used. This implementation is based on the Shi-Tomasi corner detector algorithm[43]. Retrieved feature points consist of 2D pixel locations that are matched in consecutive frames and afterwords are tracked using the OpenCV Lucas-Kanade Optical Flow implementation[3].

The second step is getting the trajectory for x,y and the angle at each frame by accumulating the frame-to-frame transformations.

The third step consists of smoothing the trajectory by using a sliding average window.

The final step consists of creating a new transformation and applying it to the current frame from the video. This new transformation is obtainable as follows:

$$new\ transformation = transformation + (smoothed\ trajectory - trajectory) \tag{4.1}$$

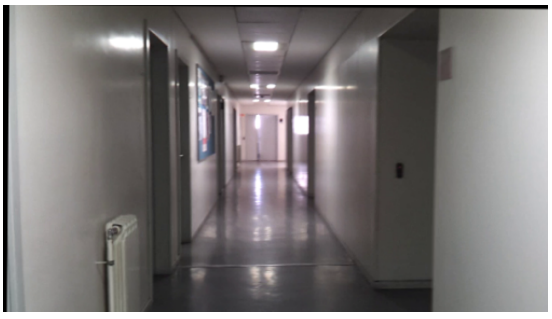
4.1.2 Results

As seen in Figure 4.1, the developed stabilization method sets out to perform as intended. When the video suffers from a strong oscillation, this method compensates this undesirable motion.

(a) Example number 1 of the original frame

(b) Example number 1 of the stabilized frame

(c) Example number 2 of the original frame



(d) Example number 2 of the stabilized frame

Figure 4.1: Result of the Stabilization method when a strong oscillation occurs
http://youtu.be/clsiq7GzUjw

[2] docs.opencv.org/modules/imgproc/doc/feature_detection.html?highlight=goodfeaturestotrack#goodfeaturestotrack
[3] http://docs.opencv.org/modules/video/doc/motion_analysis_and_object_tracking.html

4.1.3 Conundrums

The visible black border that appears in the frame when there is motion compensation could be removed by simply increasing area that is cropped out when this situation occurs. However, by doing so, a lot of visual information would be lost and the other methods (*Vanishing Point Detection* and *Door Detection*) would not work properly.

For this reason, the amount of cropped area that was applied to the frames was obtained by testing the algorithm in order to obtain the best cropping value that would not badly influence the other methods.

4.2 Filtering

After video stabilization and before releasing the frame to the other modules, some relevant image operation needs to take place.

First and foremost, in order to reduce the computational power required by the system to process each frame independently, each frame is scaled down to a resolution of $640x360$. This resolution was chosen after testing the performance of the framework with different resolutions and concluding that this was the minimal resolution with which the several methods still worked properly. Bellow this value there were not enough features to be used/detected.

Even after obtaining the best resolution per performance ratio, in order to reduce some noise in each frame, some image smoothing (blurring) was applied. This was done in two steps:

1. Applying a morphology operations know as *Opening*. This operation consists of dilating and image after eroding it:

$$dst = open(src, element) = dilate(erode(src, element)) \tag{4.2}$$

   It is particularly useful for removing small objects in the frames from the video, therefore providing a cleaner image to the other modules.

2. After applying a morphological operation to the current frame, a smoothing (blurring) by Gaussian was still possible without deteriorating the performance of the other modules. On the contrary, by applying a simple Gaussian blur, methods such as the *Vanishing Point Detection* which resorted to the Hough Transform for line detection, were able to perform even better.

   The Gaussian filter used had the following characteristics:

   - **Size of the kernel (neighbours to be considered):** [1, 1]
   - **Standard deviation of the Gaussian distribution:**
     - In X direction: 225
     - In Y direction: 255

All these numerical values were obtained by visual inspecting the result of the several methods of the framework.

4.3 Vanishing Point Detection

When traversing a corridor or a hallway, one can easily observe four lines drawn to the ends of that corridor or hall. These four lines are formed by the intersection of the floor and the walls and the subsequent intersection of these lines will be the vanishing point of that image.

The detection of the vanishing point in a video feed has been widely used in autonomous navigation of aerial and ground vehicles in environments such as corridors and hallways[44, 45, 46]. Following the vanishing point will keep the vehicle, in this case the drone, aligned with the centre of the path, consequently avoiding collisions with the walls.

4.3.1 Proposed Method

The process of detection the vanishing point in a corridor consists of the following steps:

- **Step 1:** Detecting the lines in the current frame.
- **Step 2:** Finding the slope of those lines.
- **Step 3:** Filtering out horizontal and vertical lines using the slope as a filter.
- **Step 4:** Filtering out unnecessary lines in order to obtain only diagonal lines.
- **Step 5:** Calculating the intersection of all the diagonal lines.
- **Step 6:** Obtaining the vanishing point which is the point with the highest number of intersections.

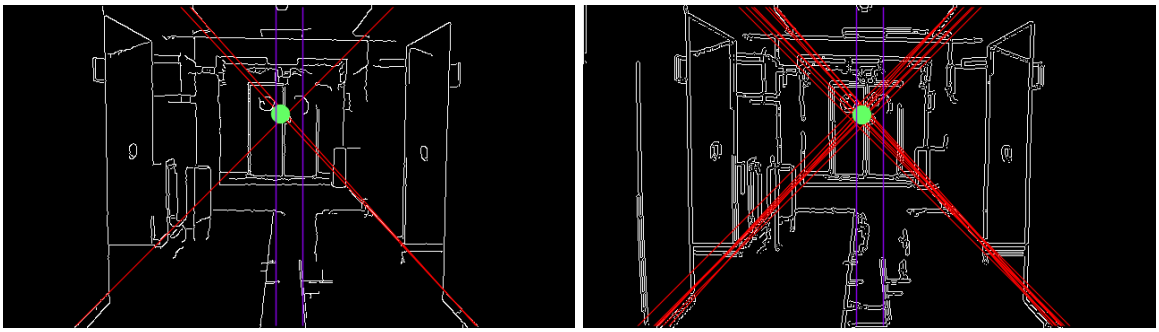
This is achieved by using the Hough Transform in order to detect lines in each frame. Although being a computationally expensive method for line detection, the Hough Transform is extremely robust even when working in cases of occlusion and noise[4].

In order for the Hough Transform to work properly, the edges of the current frame must be found and only then the Hough Transform should be applied to the matrix containing those edges. This is accomplished by using the OpenCV implementation of the Canny edge detector[4] which finds the edges in a image using the Canny algorithm for edge detection[3].

It is important to mention that the threshold used in the Canny edge detector is not static. Using a static threshold value would pose no problems if the purpose of this work was just to detect lines in an image. However, it is important to take into account that it is a video feed that is being processed which means that the threshold value used to detect the edges in a previous frame might not be ideal to detect the edges in the current frame or even the one that follows. For this reason,

[4] http://docs.opencv.org/modules/imgproc/doc/feature_detection.html?highlight=houghlines#canny

a dynamic threshold value is used, meaning that this value will start off as a certain value, then it will be reduced when the Hough Transform does not return enough lines to detect the Vanishing Point. Figure 4.2 illustrates this particular situation.



(a) Higher Threshold: enough lines were detected to acquire the vanishing point (the threshold value therefore remains intact)

(b) Lower Threshold: not enough lines were detected to acquire the vanishing point (the threshold value therefore decreases)

Figure 4.2: Example of the dynamic threshold used in the Canny edge detector

In order to obtain the slope of the lines we convert the polar coordinates of a line in the Hough space (ρ, θ) into Cartesian coordinates (m, b):

$$m = -cos(theta)/sin(\theta) \tag{4.3}$$

$$c = \rho * (1/sin(\theta)) \tag{4.4}$$

After this conversion, to filter out the unnecessary lines, all the lines whose absolute value of m (slop) is not within the interval $[0.5; 5]$ are discarded. This interval approximately corresponds to:

$$(\theta > 170^{o}\ OR\ \theta < 10^{o})\ AND\ (\theta > 80^{o}\ OR\ \theta < 100^{o}) \tag{4.5}$$

The values were hand picked after various trial and error attempts at find the correct slope interval to use as a filter. The Cartesian coordinates were used to filter out the lines since they allowed for a higher level of precision.

Regarding the intersection of the detected lines, as mentioned above, the threshold value used in the Canny edge detector is decreased when there are not enough lines to detect the vanishing point in the current frame. Obviously, being the Vanishing Point the intersection of the detected lines, the minimal number of lines that need to exist in order for this to work properly is 2. Therefore, if the number of detect lines is lower than 2, the threshold used in the canny edge detector is decreased and another attempt at acquiring enough lines for the vanishing point detection takes place.

On the other hand, if there are enough lines (more than 2) then the intersection point is determined by this simple method:

- Acquiring the equation of the lines detected in each frame: $y = mx + c$.
- Having in mind that at the point of intersection, the two equations will have the same values of x and y, the following step is to set the two equations equal to each other. This results in an equation that is solvable for x.
- Substituting the x value in one of the line equations and solving it for y results in the x and y coordinates of the intersection.

Obtaining the point with the most intersection of the lines is not enough when processing a live video feed. In some situations there might be a interference either by some physical destabilization occurring in the vehicle, by some person walking in front of the vehicle or even by some posters that might be on the walls. Due to this, some outliers might appear and so, they need to be removed.

Outlier removal is achieved by simply using a Kalman filter, introduced in the early 1960's by Rudulf Emil Kalman[47], to maintain the vanishing point stable throughout the whole video stream. This means that abrupt changes in the coordinates of the detected vanishing point will be smoothed by the Kalman Filter. Basically, the Kalman filter that was used is a 2D tracker of the coordinates of the detected vanishing point that is more immune to noise. For this reason, the Kalman filter is set up with 2 dynamic parameters and 2 measurement parameters (no control), where the measurements are the 2D location of the vanishing point. This also makes the transition matrix simple.

Tracking with Kalman filters involves the initialization of the OpenCV *CvKalman* structure and using the following sequence of operations:

- Predict by using *cvKalmanPredict*.
- Associate with a measurement.
- Correct using the measurement by calling *cvKalmanCorrect*

Initialization of the Kalman filter consists of constructing the transition matrix 4.6, the process noise covariance matrix, the measurement covariance matrix and the measurement transition matrix.

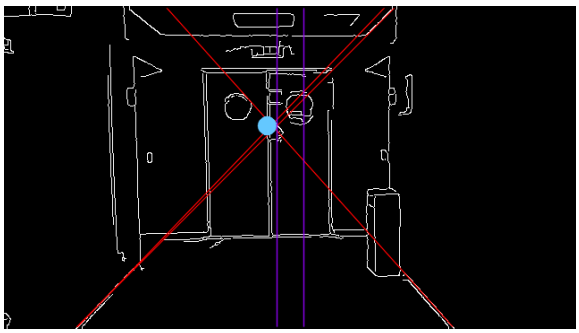
$$\begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.6}$$

When a new vanishing point is detected, the x and y coordinates of that point, the measurement parameters are associated with the Kalman tracking structure.

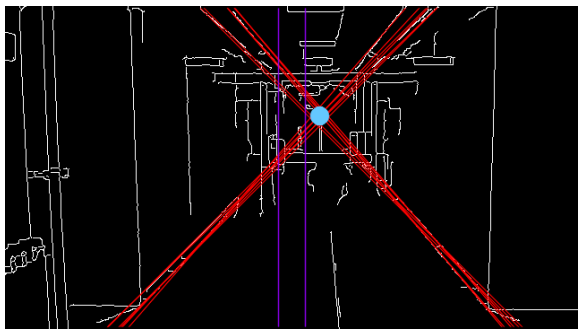
In the end, instead of having the vanishing point widely oscillating in the video feed, the Kalman filter estimates where the vanishing of the new frame is, leading to a much smother detection.

4.3.2 Results

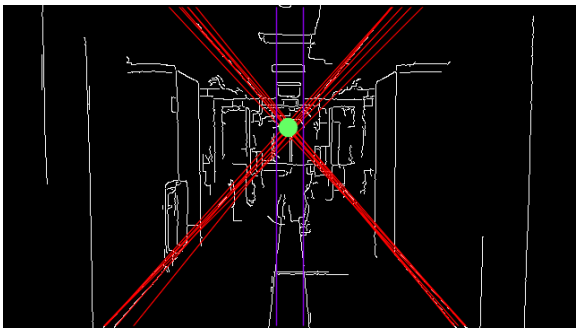
Figure 4.3 shows the result of this module. The vertical lines are only representations of the acceptable limits in which the vanishing point is supposed to be placed.



(a) Vanishing point is on the left margin



(b) Vanishing point is on the right margin



(c) Vanishing point is within the acceptable limits

Figure 4.3: Example of the vanishing point detection method
https://youtu.be/LoF001DMSoU

This information will later be used to order the drone to adjust its yaw to the left or to the right depending on the location of the vanishing point:

- If the vanishing point is on the left margin, then the yaw is adjusted to the left;
- If the vanishing point is on the right margin, then the yaw is adjusted to the right;
- If the vanishing point is within the acceptable limits, then there is no need to adjust the yaw;

4.3.3 Conundrums

Since this method is strongly dependent on the Hough transform for line detection, when some corridors have, for instance, boards, posters and, worst of all, boards with posters, some major interference occurs in the detection of the vanishing point as seen in Figure 4.4.

However, thanks to Kalman filtering this situation does not present a real problem since the these are sporadic events and the vanishing point is quickly restored to its stable position in the following frames.

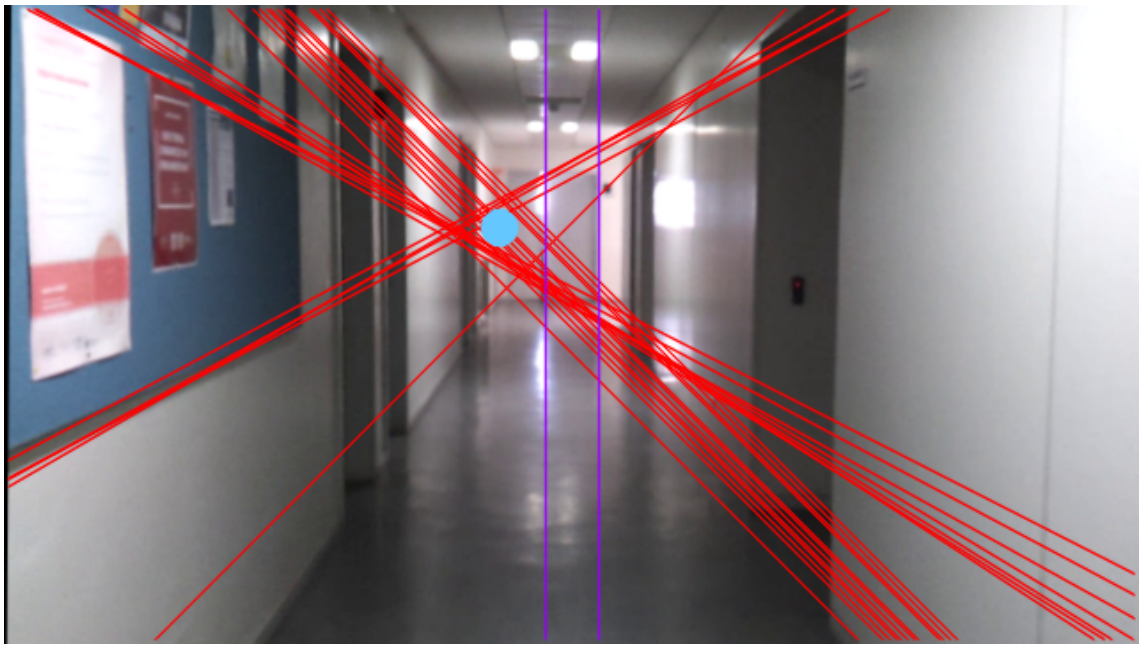


Figure 4.4: Example of a bad detection of the vanishing point due to the interference of a board with posters

4.4 Door Detection

As a means to achieve localization estimation, the approach taken was that of detecting structural elements of the corridors in which the drone would be navigating. From all the corridors at FEUP, the only structural element that was a constant in all the corridors were doors. For this reason, and since the drone would not be able to safely navigate in every corridor (as explained further in section 5), in order to estimate the position of the drone, only doors were as a structural element to be detected.

In order to detect the doors of a corridor while the drone is traversing it, a single detection method was perceived as not being enough. The reason behind this is that, while traversing the corridor, the front facing camera of the drone does not see the doors, but instead sees recessed doors such as those in Figure 4.5

In order for a door detection method to detect this kind of doors and since the objective of this project was to build a robust door detection module, the approach taken to solve this problem was based on a dual validation system consisting of the following steps:

- **Detecting the recess in the walls:** This will indicate that a door or even another corridor might be there.

- **Detecting the contours in the corridor:** The contour of a door is the largest rectangle whose height is bigger than its width.

Figure 4.5: Example of two recessed doors as seen by the drone

- **Assigning weights to each detection:** Since the recessed door detection is more noise resistant and robust than the door contour detection, the former was given a weight of 0.6 and the latter a weight of 0.4.

- **Dual validation system:** A validation system takes place when a recess is detected in either the left or the right side of the frame. It was stipulated that the recessed door detection would have a validation weight of 6 and after detecting a door, it awaits a valid contour detection which has a validation weight of 4:

  - If a contour is detected after a recess detection, than the door will be detected with 100% accuracy.

  - Otherwise, if after a small period of time, the only detection that occurred was that of the recess, than the door is detected with only 40% accuracy. When a recess is not detected, hardly ever is a door contour detected, hence the lower validation weight assigned to this method).

4.4.1 Recessed Door Detection Method

This method aims at detecting the recesses in the corridor which would mean that a door or even a corridor was detected.

Implementation

In order to detect the left and right recesses in the corridor, this method was divided into two parts: Floor Segmentation and Corner Detection. The objective was that of isolating the floor from the rest of the frame and then detecting the recesses in the floor.

Segmentation

To achieve this, the watershed segmentation implementation of the OpenCV library was used[5], which performs a non-parametric marker-based image segmentation[10].

The basic concept of this algorithm is that a grayscale image can be considered a topographic surface where a high and low intensities stand for hills and valleys respectively. Every isolated valleys is a local minima and is filled with coloured water (labels). Water from different valleys (different labels) will begin to merge when the water stars to rise, which is dependent on the hills (gradients) nearby. In order to avoid merging the water from different valleys, a barrier must be built in the location where the merge occurs. Continuously filling water and building barriers until the hills are underwater will then lead to the segmentation result defined the barriers previously created.

As the gist of this step is to segment the floor from the rest of the scene, a marker matrix is created in which the desired regions that are going to be segmented are outlined. This means that every region is represented as one or more connected components. In other words, this is an interactive image segmentation in which different labels are assigned to the known objects in the frame. Since the floor is the foreground and the rest of the frame is background, they are labelled accordingly resulting in the marker that is going to be applied to the watershed algorithm. Figure 4.6 shows the markers used in order to segment the floor from the corridor.

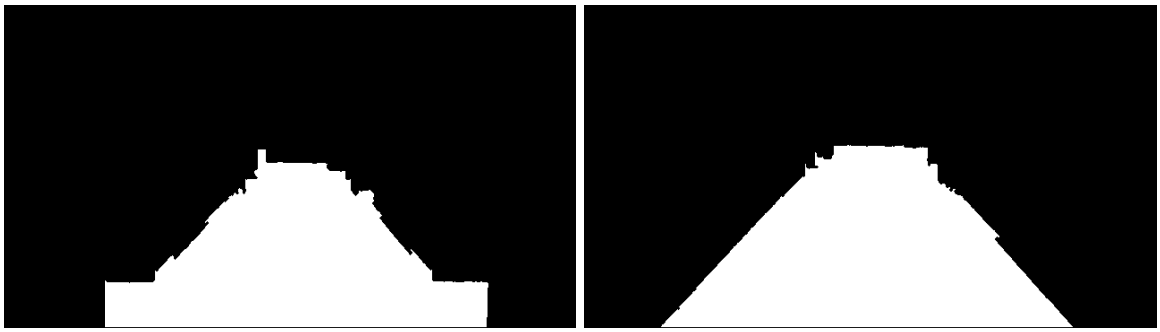
(a) Markers (b) Markers on the corridor

Figure 4.6: Markers used to select foreground (floor) and background (everything else)

As seen in Figure 4.7, the floor is correctly segmented by applying the algorithm described above.

Paying close attention to Sub-figure 4.7a, the recess of both doors (left and right) is present in the form of a rectangle and is quite noticeable. For this reason, the next step is to find the vertex of

[5]http://docs.opencv.org/modules/imgproc/doc/miscellaneous_transformations.html#watershed



(a) Floor Segmentation near the doors (b) Floor segmentation away from the doors

Figure 4.7: Examples of successful floor segmentations
https://youtu.be/3M78qyrev1k

those rectangles, meaning that the vertex in the far left and far right side of the frame corresponds to the a recess in the left and the right side of the corridor.

Vertex Detection

The second part of this method is comprised of four steps.

The **first step** consists of applying a threshold to the image obtained in the previous step.

The **second step** consists of using the Hough transform[4] to detect horizontal and vertical lines. Detected horizontal lines are considered to have a slope m within the interval $[0;0.001]$ and detected vertical lines are considered to have a slope m bigger than 1000. Therefore, by filtering out all the lines whose slope m does not meet this criteria, only the horizontal and vertical lines are left.

The **third step** consists of finding useful intersections using the method previously explained in 4.3.1. Since the door detection is only useful when the drone is near a door, in order to prevent a recessed door detection that is too far from the drone, only detections that occur below one fourth of the frame are considered as valid. Figure 4.8 shows an horizontal line on the frame (after segmenting the floor and applying a threshold) which separates the area where the useful intersections will be located (below the line) from the area where the intersections will be ignored.

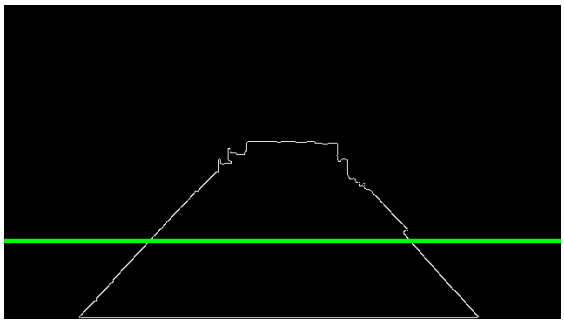


Figure 4.8: Floor segmentation, thresholding and the useful intersection area

In the **final step**, after acquiring all the useful intersections, only those that are the ones in the far left and far right side of the frame are considered as being doors.

In the video feed, when the drone is approaching a door, several consecutive recesses are detected. In order to refine the recessed door detection method, temporal information is used. Figure 4.9 shows the algorithm that is used to detect recessed doors:

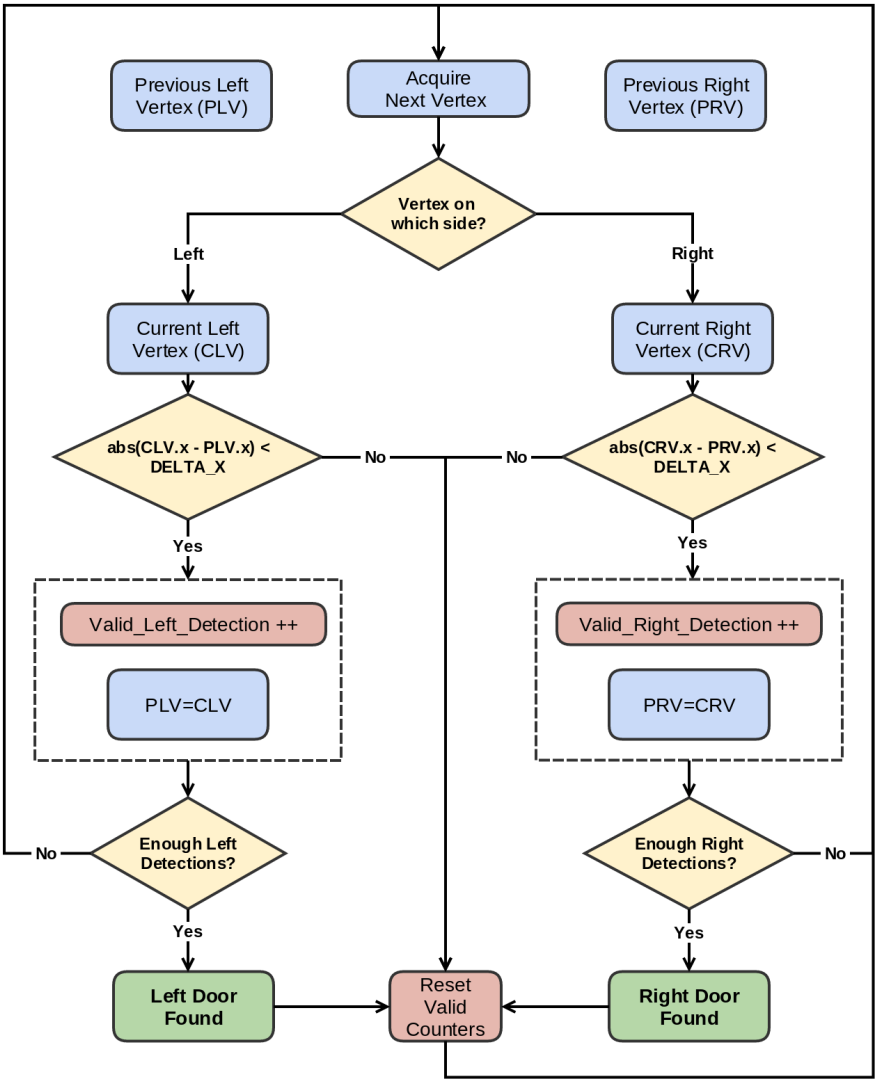
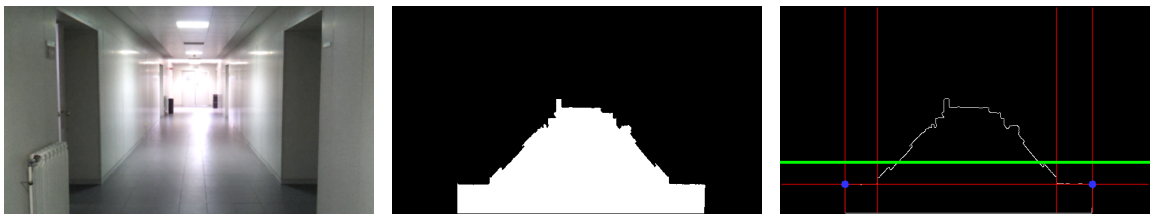


Figure 4.9: Recess door detection vertex temporal validation algorithm

If the absolute value of the difference between the X coordinates of the detected vertex previous one, is smaller than a stipulated value ($DELTA\_X$) than this vertex is not considered a suitable "recess candidate" being therefore discarded. Only after enough consecutive valid detections (that value being adjustable depending on the desired precision) is a recessed door detected.

Results

As seen in Figure 4.10,which shows the results of the developed method,the vertexes of both doors are successfully detected.



(a) Original frame (b) Floor segmentation (c) Successful detection

Figure 4.10: The various steps of the recessed door detection method
https://youtu.be/alToFq8htFk

Conundrums

Two different types of errors originate from this approach.

Figure 4.11 is an example of a vertex detection that does not correspond to a recess, hence the need for temporal information in order to eliminate these faulty detections: accumulating consecutive recess detections, and establishing a successful door detection based on a fixed number of consecutive detections, instead of interpreting each detection as a success. The latter would lead to many erroneous detections due to artefacts in the video stream.

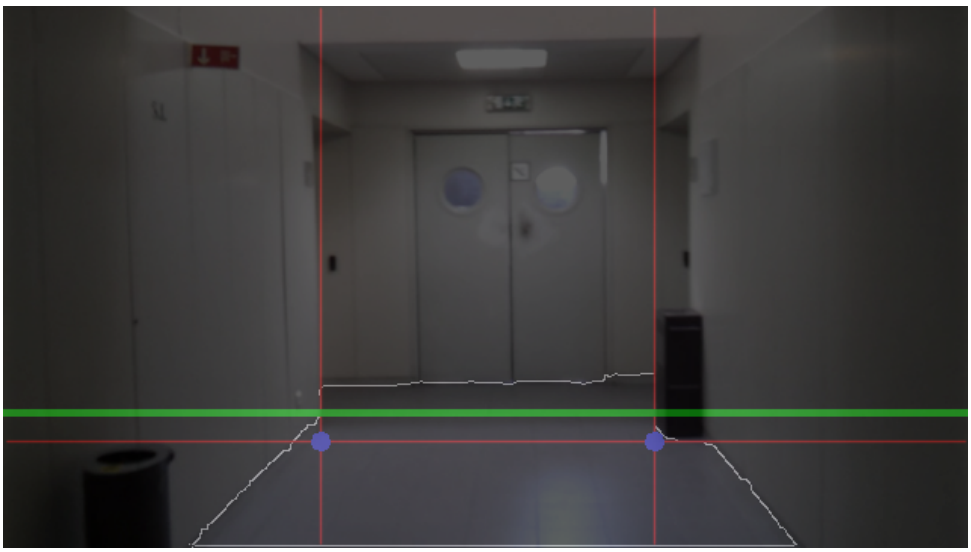


Figure 4.11: Example of a sporadic invalid recessed door detection

Due to reflections on the floor, an invalid floor segmentation (using the watershed algorithm) can occur as seen in Figure 4.12.

(a) Original frame and markers (b) Invalid floor segmentation (c) Unsuccessful detection

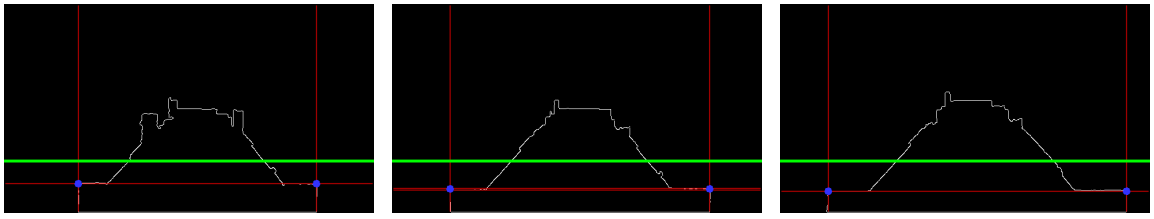
Figure 4.12: Unsuccessful left recessed door detection due to floor reflections

An invalid segmentation due to reflections on the floor can have catastrophic outcomes when it comes to recess detection. Figure 4.13 shows an example of an unsuccessful detection due to an invalid segmentation.

(a) Original frame and markers (b) Invalid floor segmentation (c) Unsuccessful detection

Figure 4.13: Unsuccessful left recessed door detection due to floor reflections

However, it is important to point out that these problems caused by undesirable floor segmentations do not affect the robustness of the developed method thanks once again to the use of temporal information. This information consists of the coordinates of the valid detected vertexes that are considered to be close to each other on consecutive frames. This means that, in order to determine if a set of consecutive vertex detections are considered as a door, during a certain period of time, if successive detections occur, each new detection is cross-validated with the previous one and if the coordinates of the newly detected vertex are not within a certain range of the previously detect vertex, it is considered a bad detection. Therefore, only consecutive vertex detections whose coordinates form a diagonal line in the frame are considered valid detections, meaning that a door was successfully detected. Figure 4.14 shows and example of valid consecutive vertex detections.



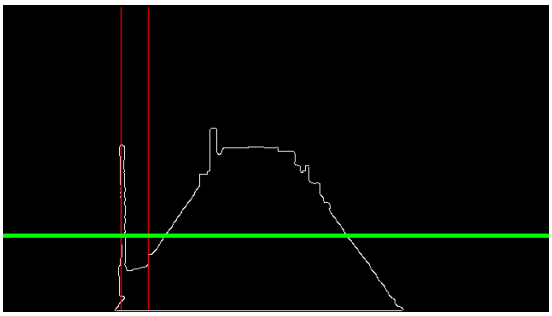
(a) First valid vertex (b) Second valid vertex (c) Third valid vertex

Figure 4.14: Example of a sequence of valid vertex detections

Another situation in which the recess detection does not work properly is when corridors have

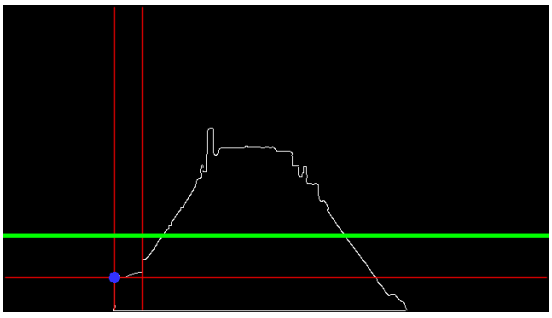
doors with narrow recesses. Figure 4.15 shows what happens in this specific situation: even though the recess detection works in some frames of the video feed, it does not detect the same recess in enough consecutive frames in order for the temporal information to detect a recessed door.

(a) Example number 1 of a door with a small recess



(b) Example of an unsuccessful recess detection

(c) Example number 2 of a door with a small recess



(d) Example of successful recess detection

Figure 4.15: Result of the Stabilization method when a strong oscillation occurs

Nevertheless, as explained in chapter 5.1 this would only prove to be a problem during simulations in these corridors since they did not met the conditions that would allow the drone to fly in it.

4.4.2 Door Contour Detection Method

This method aims at detecting the contours of doors in corridors, filtering out the unnecessary contours in order to detect only those who correspond to doors: those whose height is bigger than its width.

Implementation

In order to find the contours of the doors in each frame, the following sequence of operations was used:

- Apply a threshold to the current frame in order to obtain a binary image.
- Detect the edges in the current frame using the Canny edge detector[3].
- Retrieve the contours from the binary image using the algorithm [48]. This method retrieves all of the contours and compresses horizontal, vertical, and diagonal segments leaving only

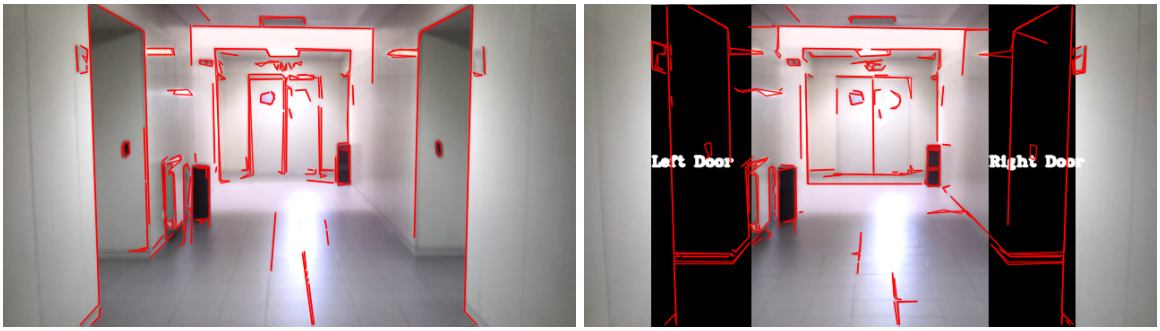
their end-points. This means that, for example a rectangular contour is encoded with 4 points.

- Approximate the contours to a rectangle, which will make the contour sides be a lot more regular, using the Douglas-Peucker algorithm[49].
- Segment the rectangular contours using geometry. Since the contour of a door is a rectangle whose height is bigger than its width, all the other rectangles are discarded. The remaining rectangles are filtered out if a contour of a door has a width that is too low (the rectangle would be to narrow), which would result in false door detections.

Resembling the recessed door detection method, the door contour detection method also uses temporal information in order to avoid unreliable detections such as those show on section 4.4.2. This is achieved by simply keeping a counter for booth left and right door contour detections. When that counter reaches a specific value (which is adjustable depending on the desirable detection accuracy) a door is detected and the corresponding counter is reset.

Results

In the end, the contours of the doors in the video feed are detected properly as seen in figure 4.16 (the red line represents the detected contours outlines drawn into the frame using the OpenCV method *drawContours*[6]).



(a) Before the detection (b) After the detection

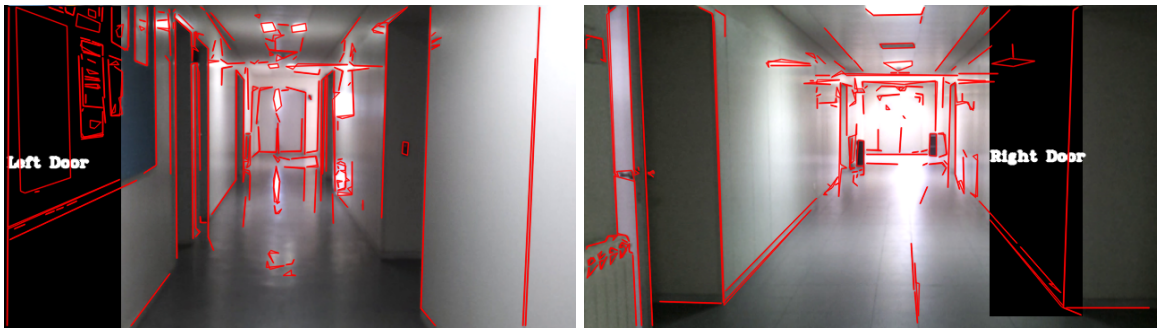
Figure 4.16: Example of the door contour detection method
https://youtu.be/lhktcVTLNrg

Conundrums

Once again, posters and boards present a problem. This time around, shapes from these structures are wrongly identified as doors just because they meet the criteria sporadically. Figure 4.17 presents an example of this situation.

This emphasizes the fact that using only this method to detect doors is not enough: it is susceptible to errors such as those already mentioned. Although the erroneous detections are not that

[6]http://docs.opencv.org/modules/imgproc/doc/structural_analysis_and_shape_descriptors.html?highlight=drawcontours#drawcontours



(a) Invalid detection on the left (b) Invalid detection on the right

Figure 4.17: Example of problematic door contour detections

far off, using only this method would lead to errors in the position estimation of the vehicles as it would be detecting doors out of place

4.5 Position Estimation

The final module of the developed framework is responsible for the position estimation of the vehicle traversing the corridor.

4.5.1 Proposed method

In order to estimate the position of the drone in the corridor, the following data is used:

- The velocity of the drone.
- Precoded Map Information.
- The door detection information.

The **velocity of the drone** is either a real-time value provided by the vehicle being used, or, assuming that the vehicle is moving at a steady pace, a constant value.

The **precoded map information** is a text file containing relevant metric data from the corridor in which the drone will be traversing. Before each flight the framework loads the corresponding floor map data containing the following information (all the measurements are in centimetres):

- Storey.
- Length of the corridor.
- Point of origin: the initial location (point of departure) of the drone. Can be either *north* or *south*.
- Total number of rooms in the corridor.
- For each room: ID and distance to the point of origin.

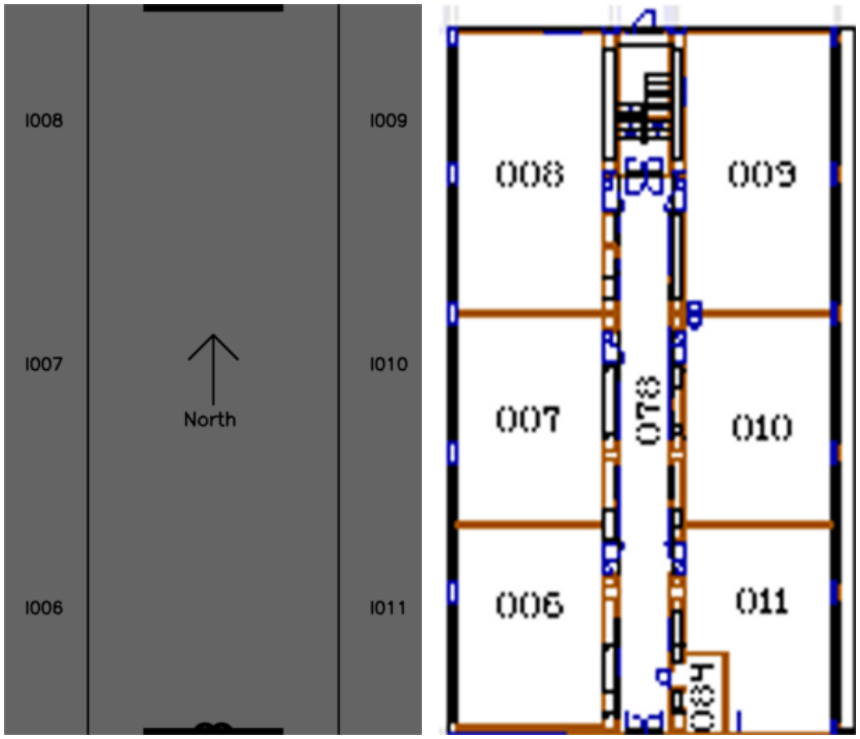
Table 4.1 shows an example of data file used to load the framework with all the data from the corridor.

Table 4.1: Example of a corridor data file

storey	0	
length	2300	
entrance	north	
rooms	6	
left	I006	500
right	I011	500
left	I007	1700
right	I010	1700
left	I008	2300
right	I009	2300

The **door detection information** is obtainable from the door detection module.

After reading the floor information provided by the data file, a visual representation of the floor map is created. Figure 4.18 shows a comparison between the representation of a map created by the framework (using the data file shown in Table 4.1 and its original schematic[7]).



(a) Rendered Map (b) Original Schematic

Figure 4.18: Comparison between the rendered representation of a corridor and its schematic

[7] sigarra.up.pt/feup/pt/instal_geral.edificio_view?pv_id=1414&pv_num_piso=0

The map itself never changes, however, two types of information are updated/drawn on the map: the position of the drone and the location of the detected doors.

The position of the drone is updated depending on its current velocity and the initial point of departure.

The location of the detected doors are drawn on the map when the current position of the drone is near the location of a door and the door detection module actually detects a door. This means that a confidence interval is used regarding the location of the door. For instance, using the data from Table 4.1, if the position of the drone is known to be 600 centimetres from the initial point of departure and a left door is detected, since there is a door at 500 centimetres, the drone is considered to be near that door leading to a valid detection of the left door *I*006. Figure 4.19 is representative of this situation.

Figure 4.19: Updating the position of the Drone and the detected door

4.5.2 Results

Using all the information provided by the precoded map data, the velocity of the vehicle and the door detection module, the result of the this module, which uses all the previously developed modules, can be seen in Figure 4.20.

4.5.3 Conundrums

Since this module is the final output of the framework, showing a graphical representation of the result of the door detection, the problems and errors that cause an invalid estimation of the position of the doors are due to the same errors that cause an invalid door detection as mentioned in 4.4.2. In other words, the problems that surface from this module can be succinctly described as being directly influenced by the behaviour of the door detection module as seen in Figure 4.21

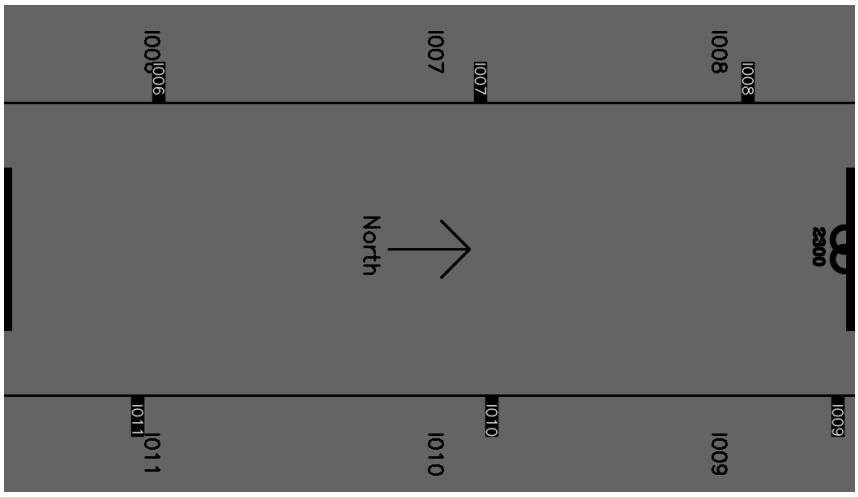


Figure 4.20: Example of a successful door detection and position estimation
https://youtu.be/HfYTlzBVStk

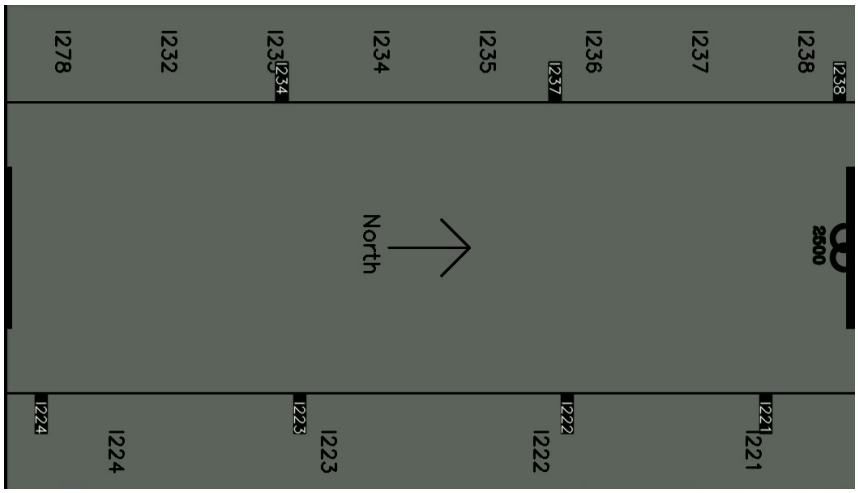


Figure 4.21: Example of a not so successful door detection

4.6 Summary

All in all, the developed framework was built upon the principle of "divide and conquer" in that each module provides separate and independent information that is later used either for navigation or location estimation.

Overall, the video stabilization method worked as intended by simply compensating strong oscillations occurring in the video feed while preserving the original video when no compensation was required.

The vanishing point detection method works flawlessly which leads to a satisfactory navigation orientation for the drone. Even in situations where an invalid vanishing point is detected, either by external interferences (e.g posters hanged in the walls) or by the lack of visual information available, this module is able to compensate these disparities by either using Kalman filters to

smooth the oscillation of the detected vanishing point or evaluating the time period during which the vanishing point is not detected: for safety reasons, if the detection is interrupted, the drone will hover and wait for another set of valid detections and, if it takes too much time to do so, the drone will land.

The door detection method works as expected, detecting the doors when they are in close proximity to the drone. By making use of the information from both door detections methods as well as temporal information to avoid invalid sequential detections, a robust and coherent door detection module was created.

The position estimation module proved to work as intended by using all the information provided by the other modules and using it, alongside the pre-coded map, to estimate the position of the drone as well as the detected doors.

Although some problems were detected with the different modules, which in some situations would bleed through to other modules, precautions were taken so that the modules could work even in the most unfavourable situations.

Chapter 5

Experiments and Discussion

In this section, the results of the experiments and simulations that were conducted are presented. The videos used for simulating the trajectory of the AR. Drone were recorded with a video camera similar to the one on-board the drone, while the actual AR. Drone was use in the experiments.

5.1 Experimental Setup

All the experiments were carried out inside the Department of Electrical and Computer Engineering[1] in the Faculty of Engineering of the University of Porto[2].

A laptop computer (Clevo P751ZM), connected over Wi-Fi to the AR. Drone, is used to control the movement of the drone based on information processed from the video feed provided by the drone itself. The technical specifications of the laptop are:

- **Processor:** Intel® Core™ i7-4790K quad-core Processor @4.0GHz
- **Memory:** 16 GB DDR3 @1600MHz
- **Operating System:** Linux Mint 17.1 Rebecca
- **Wireless Card:** Intel® Dual Band Wireless-AC 7265

5.1.1 Aditional Considerations

Initially the drone was supposed to be able to navigate in two different corridor types: one wider than the other. Figure 5.1 shows the difference between these corridors.

However after testing simple take-off, hover and navigation commands on the Parrot AR. Drone, it was noticeable that it could not maintain a stable trajectory on the narrower corridor. The reason for the loss of stability was due to different airflows:

- Originating from the various rooms behind the doors in the corridor.

[1] www.fe.up.pt/deec
[2] fe.up.pt

(a) Wider Corridor (b) Narrower Corridor

Figure 5.1: Two different types of corridor

- Caused by the rotors of the drone. Since the corridor is so narrow, the airflow caused by the drone does not disperse uniformly causing the drone to destabilize.

In some situations, the AR. Drone would lean backwards at an alarming speed trying to compensate for the airflow that was interfering with its ability to stabilized. Simple tasks such as hovering were impossible and the drone would just hit the walls and crash.

Therefore, even though the narrower corridors were used to simulate the behaviour of the developed framework in a real situation, it was not used when testing the framework with the Parrot AR. Drone.

5.1.2 Corridor Dimentions and Charateristics

The corridor used to test the developed framework on the AR. Drone was the corridor on the DEEC Ground floor. It has a length of 23 meters and 6 rooms: 3 on the left and 3 on the right (all with recessed doors).

Two different trajectories were used as test scenarios:

- Northern entrance as the starting point.
- Southern entrance as the starting point.

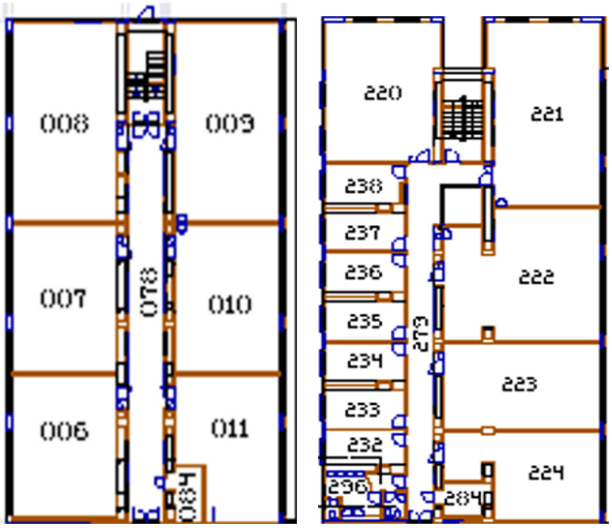
The corridor which was used only for simulations was the one on the DEEC 2^{nd} floor, using the northern entrance as the starting point, having a length of 25 meters and 12 rooms: 8 on the left and 4 on the right (only the latter were recessed doors).

Figure 5.2 shows the schematics of these two corridors.

5.2 Navigation and Position Estimation

All the simulations and the experiments were conducted in optimal conditions, meaning that:

- The corridor was empty, meaning that there were no people walking along the corridor.



(a) Ground Floor (b) 2^{nd} Floor

Figure 5.2: Floor Schematics

- All the doors were completely shut in order to reduce airflow interference (only valid during the experiments).

The following sections will present the results from both the simulation and the experiments and compare them.

5.3 Flight Experiments

Several flight experiments were conducted to test the performance of the developed method and in order to highlight both success and failure conditions.

In order to evaluate the behaviour of all the framework modules, a control window containing the output of the different modules, is displayed in the laptop. In Figure 5.3 an example of this control window can be observed.

As already mentioned, all the modules are displayed in this window.

- Position Estimation (Map).
- Video Stabilization.
- Vanishing Point Detection.
- Door Detection:
  - Recessed Door Detection
  - Door Contour Detection

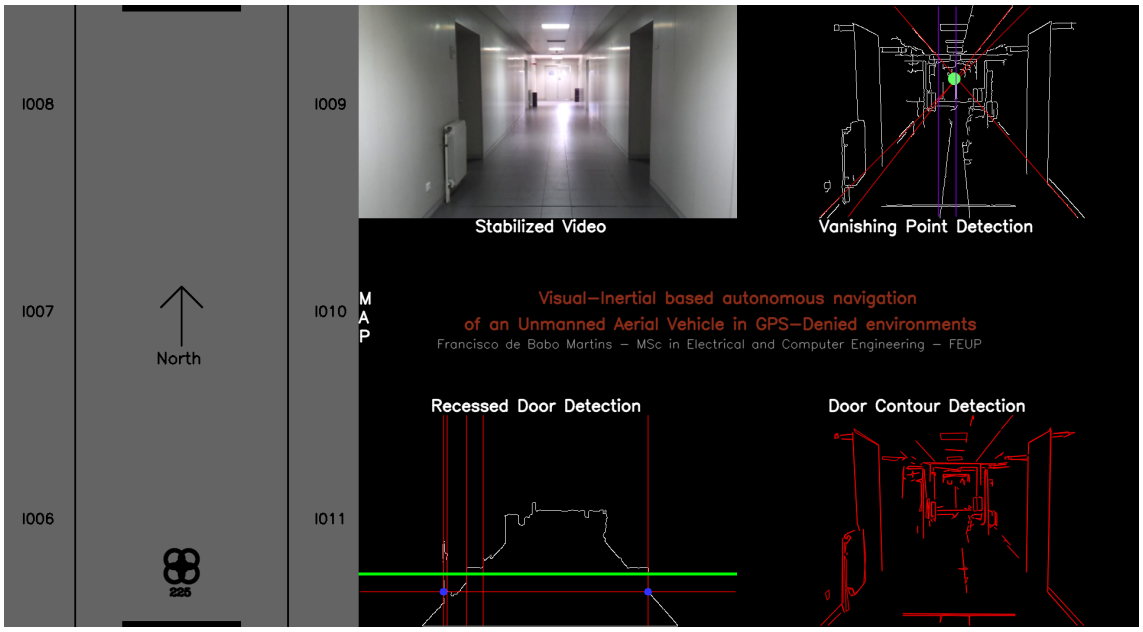


Figure 5.3: Example of the control window provided by the framework

Several tests required a change in the temporal parameters of the door detection method, namely the ones responsible for the minimal number of consecutive recesses and contour detections (which lead to a valid door detection as explained in chapter 4).

For each experiment, several test runs were executed in order to validate the effectiveness of the developed system.

5.3.1 Experiment 1

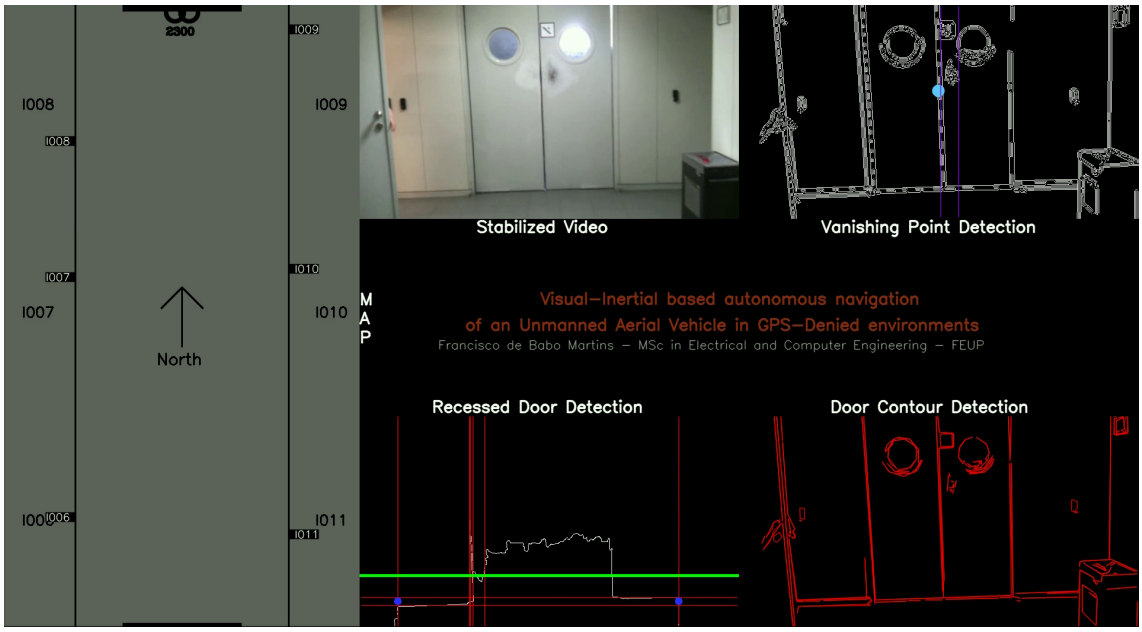
For experiment 1, the drone attempts to traverse the corridor located on the DEEC ground floor by taking-off near the northern entrance and moving towards the southern entrance.

As seen in Figure 5.4, which shows both the result of the simulation and the experiment of this particular test run, the drone was able to safely navigate from one point to the other while correctly detecting all the doors.

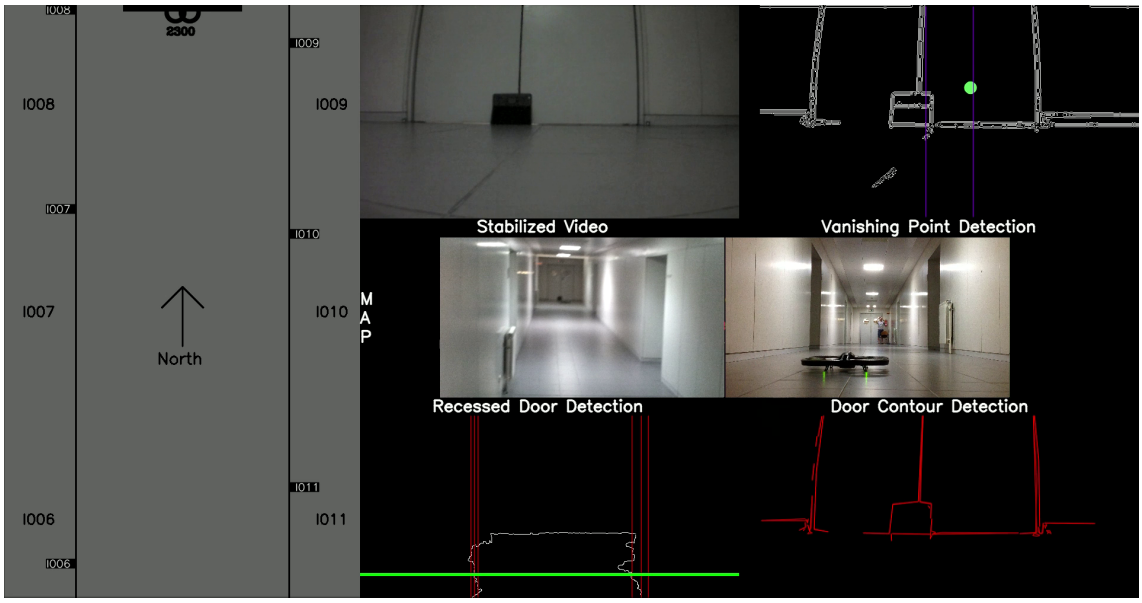
If the drone was able to navigate from one point of the corridor to the other without hitting the walls, it means that the vanishing point was successfully detected during the journey and the movement of the drone was compensated in situations where the vanishing point would get out of focus.

Figure 5.5 shows the coordinates of the vanishing point detected while the drone is traversing the corridor. The results were the ones expected since the vanishing point converged near a certain point, demonstrating that the developed method was working properly.

In order to emphasize the robustness of the vanishing point detection, table 5.1 presents both the standard deviation (in pixels) and the mean value of the coordinates of the vanishing point



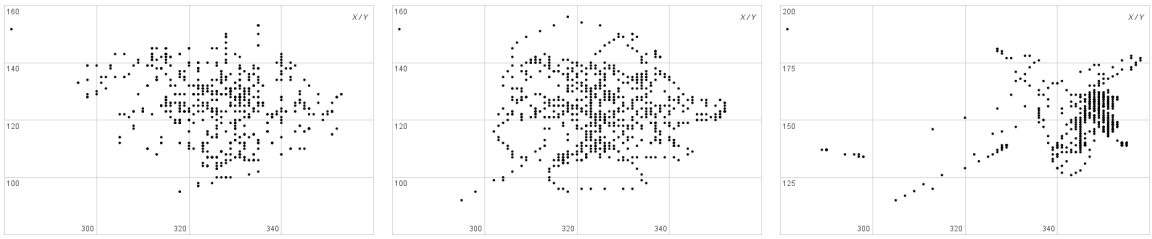
(a) Simulation
http://youtu.be/dFCO8Ow2EKg



(b) Experiment
http://youtu.be/xjIpChc9Oio

Figure 5.4: DEEC Ground Floor Framework Simulation and Experiment number 1

detected throughout this test run. It is observable that although the results from the manual annotation, simulation and experiment are almost identical, which proves the precision of the developed detection algorithm, the results from the flight experiment were the worst of the lot and the ones that were manually annotated were the best. This is understandable since the data that was manually acquired was done by a human being, which perceives the vanishing point in an image with a lot more precision and the data from the simulation and the experiment was acquired by the



(a) Annotation (b) Simulation (c) Experiment

Figure 5.5: Vanishing Point dispersion map for flight experiment number 1

developed system. However, the overall result of the experiment did not substantially differentiate from the ones obtained in the simulation and these were not far off from those acquired manually.

Table 5.1: Standard deviation and median value Vanishing Point from experiment 1

	Mean Point	Standard Deviation
Annotation	[327;126]	[10.64;11.39]
Simulation	[325;123]	[11.31;11.47]
Experiment	[350;146]	[11.47;10.88]

Table 5.2 shows the results of several simulations and test runs. From the starting take-off point to the final landing site, the drone is supposed to detect all the doors from the corridor, 3 on the left and 3 on the right.

Table 5.2: Door Detection data from simulation and experiment 1

	#Run	Left Doors Detected	Right Doors Detected	Door Detection Success Rate
Simulation	1	3	3	93.33 %
	2	3	3	
	3	3	2	
	4	3	3	
	5	2	3	
Experiment	1	3	3	86.67 %
	2	2	2	
	3	2	3	
	4	3	3	
	5	3	2	

During the simulations, only on two specific occasions did the door detection failed to detect wall the doors. However the overall result was quite satisfactory. On the other hand, more failed detections occurred during the several test runs in the corridor used for this experiment, leading to a lower success rate. Nevertheless, both the simulations and the experiments scored above 80% on the door detection test.

For each experiment several test runs were conducted in order to evaluate the success of the developed navigation module that was developed. In Table **??** different results for each conducted

test run are presented.

Table 5.3: Navigation data from experiment 1

	#Run	Travel Time (s)	Number of yaw corrections	Arrival	Arrival Success Rate
Experiment	1	35	8	yes	80%
	2	40	11	yes	
	3	33	6	yes	
	4	60	23	yes	
	5	15	2	no	

The *yaw corrections* strongly varied from one test run to the other due to external interferences like the airflow originating from the several doors and even the airflow caused by the rotors of the drone.

Although the majority of the test runs proved that the navigation method works as intended, the test run number 4 and 5 are due to some important observations:

- **Test run number 4:** During this test run, the lights of the corridor went off, causing the drone to hover while awaiting to lock on a valid vanishing point. After the lights turned back on, the drone resumed its trajectory and finished the course.

- **Test run number 5:** This was a failed test run caused by a strong oscillation in the drone which the navigation module could not compensate. The yaw failed to be correctly adjusted resulting in the drone bashing into a wall.

5.3.2 Experiment 2

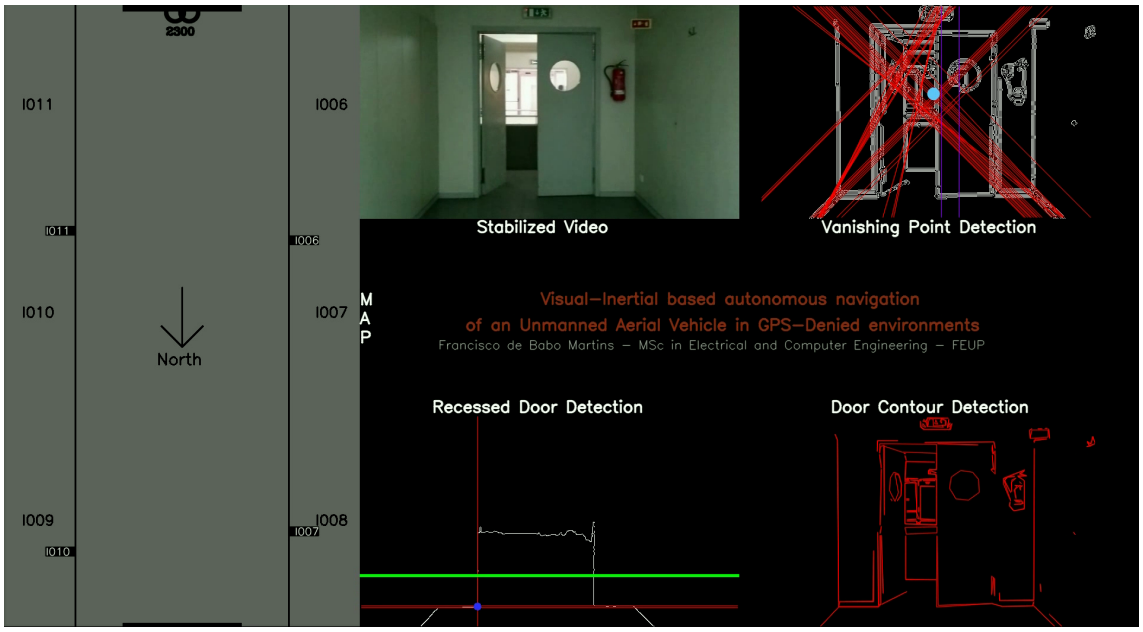
For experiment 2, the drone attempts to traverse the corridor located on the DEEC ground floor by taking-off near the southern entrance and moving towards the northern entrance.

Figure 5.6 shows both the result of the simulation and the experiment. Its important to notice that while this corridor possesses 6 doors, the drone is incapable of seeing the first two, meaning that only 4 doors were considered as valid for this experiment.

Resembling the previous experiment, a successful detection of the vanishing point provided the navigation module with the right data to successfully guide the drone down the corridor. All this while still managing to adjust the yaw of the drone in order to keep it aligned with the center of the corridor.

Figure 5.7 shows the coordinates of the vanishing point detected while the drone is traversing the corridor. The results were the ones expected since the vanishing point converged near a certain point, demonstrating that the developed method was working properly.

Table 5.4 presents both the standard deviation (in pixels) and the mean value of the vanishing point detected throughout this run. Similarly to the first experiment, it is noticeable that the results from the manual annotation and simulation are mostly identical, being the experiment results the



(a) Simulation
http://youtu.be/luWOo-Jtles

(b) Experiment
http://youtu.be/FVnKTEFN89E

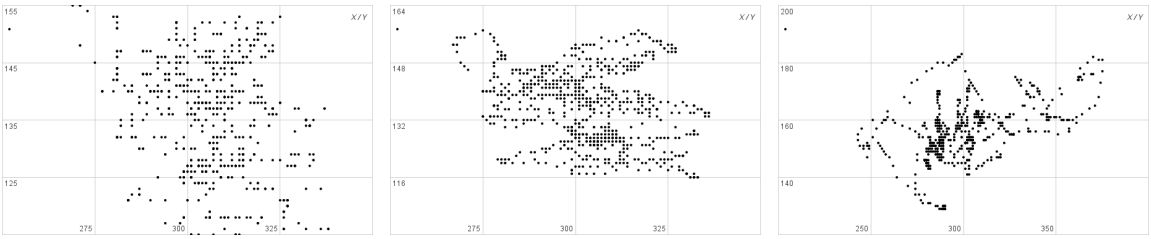
Figure 5.6: DEEC Ground Floor Framework Simulation and Experiment number 2

ones with the higher standard deviation. The overall result still allowed for a safe and collision-free navigation.

although the deviation from the experiment is much higher than before. Nevertheless

In Table 5.5 the results of the simulations an test runs conducted for this experiment are displayed.

As mentioned before, only 4 doors (2 on the left and 2 on the right) were considered as being



(a) Annotation (b) Simulation (c) Experiment

Figure 5.7: Vanishing Point dispersion map for experiment number 2

Table 5.4: Standard deviation of the Vanishing Point for experiment 2

	Mean Point	Standard Deviation
Annotation	[306;137]	[13.65;9.76]
Simulation	[304;136]	[15.14;10.29]
Experiment	[290;194]	[25.24;10.85]

Table 5.5: Door Detection data from simulation and experiment 2

	#Run	Left Doors Detected	Right Doors Detected	Door Detection Success Rate
Simulation	1	2	2	95%
	2	1	2	
	3	2	2	
	4	2	2	
	5	2	2	
Experiment	1	2	2	90%
	2	2	2	
	3	2	1	
	4	2	2	
	5	2	1	

part of the corridor since from its starting point, the camera of the drone cannot perceive the existence of the doors right next to it. Only in one simulations did the door detection module failed to deliver a complete detection of all the existing doors. However during the test runs, two detections failed, both on the right side, which was mainly due to the reasons stated in chapter 4.4. Nevertheless, the overall result was satisfactory as both simulations and test runs achieved above 90% of successful detections.

Like in the first experiment, Table 5.6 presents the results of all the test runs conducted for this experiment.

Similarly and having in mind that the corridor used for this experiment was the same as the one used for the previous one, the yaw corrections strongly varied between test runs due to the different airflows.

Although these test runs proved that the navigation method works as intended, the test run number 3 failed to accomplish a successful navigation. This was due to the back door of the cor-

Table 5.6: Navigation Data from experiment 2

	#Run	Travel Time (s)	Number of yaw corrections	Arrival	Arrival Success Rate
Experiment	1	41	5	yes	80%
	2	50	14	yes	
	3	27	3	no	
	4	38	10	yes	
	5	39	8	yes	

ridor being wide open, which strongly contributed to destabilizing the drone. This destabilization was way beyond the ability of the navigation method to stabilized the drone and resulted in the drone catastrophically bashing into the left wall and crashing (the protective hull even had to be repaired as seen in Figure 5.8).

Figure 5.8: Repairing the external hull of the AR. Drone with a double sided adhesive tape

5.3.3 Experiment 3

As previously explained, due to the lack of conditions the drone was not able to fly in this corridor and therefore only simulations were made. Figure 5.9 shows the result of the simulation.

Figure 5.10 shows the coordinates of the vanishing point detected while the drone is traversing the corridor. Since there was no practical experiment in this corridor, only the dispersion map of the manually annotated vanishing points and the simulation are shown.

Table 5.7 shows the standard deviation and the mean of the coordinates of the acquired vanishing point during the traversal of this corridor.

It is noticeable that there is a higher deviation in the simulation which is expected since this corridor presents features that interfere with a stable detection of the vanishing point like poster and boards.

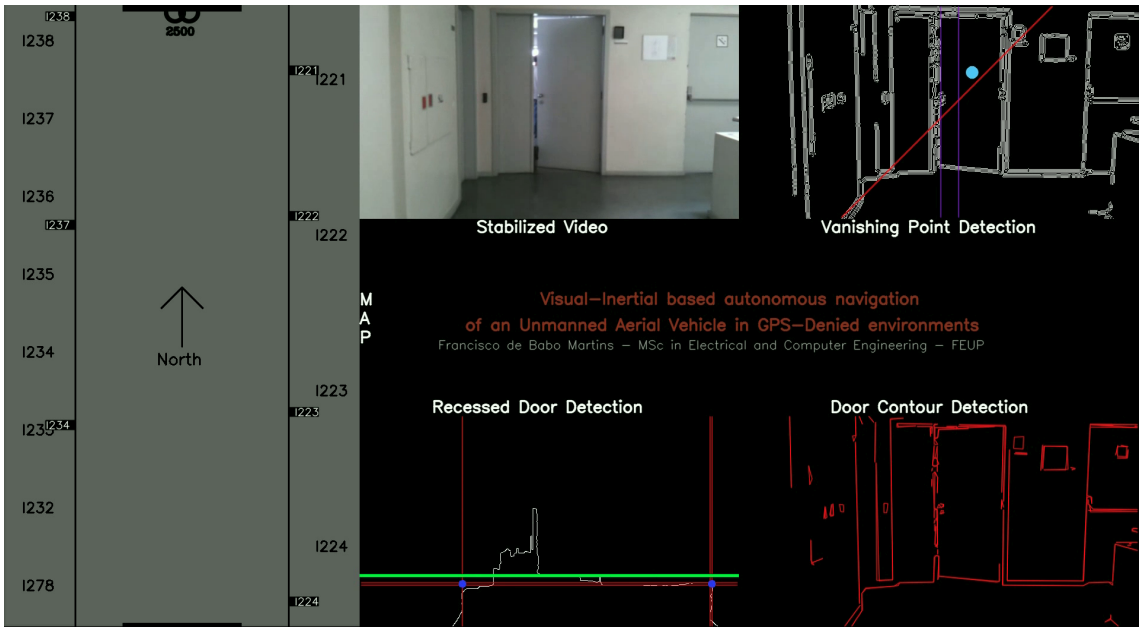
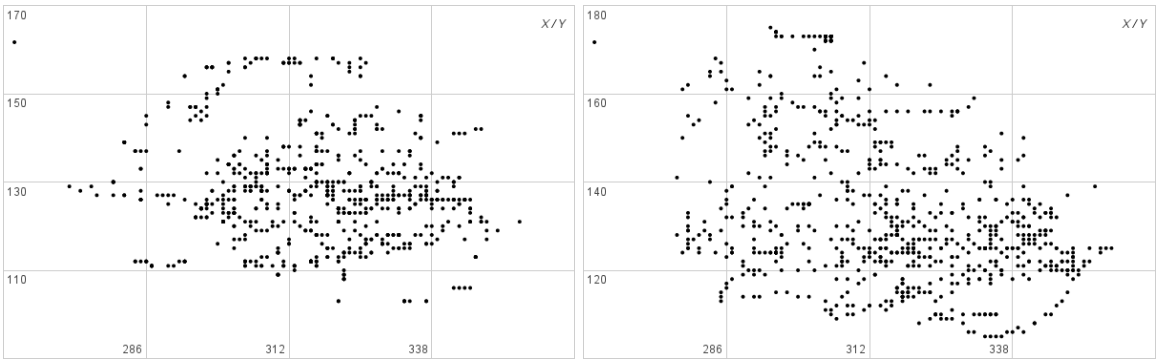


Figure 5.9: Simulation
http://youtu.be/0Q-owmC5N4s



(a) Manual Annotation (b) Simulation

Figure 5.10: Vanishing Point Detection from the simulation and experiment 2

Table 5.7: Standard deviation of the Vanishing Point for experiment 3

	Mean Point	Standard Deviation
Annotation	[317;139]	[15.93;11.95]
Simulation	[316;128]	[18.61;16.91]

This corridor presented a challenge due to the format of its doors: all the doors on the right are recessed doors whilst the doors on the right are not recessed doors. This lead to a not so accurate door detection as seen in the results on Table 5.8.

Table 5.8: Door Detection data from simulation 3

	#Run	Left Doors Detected	Right Doors Detected	Door Detection Success Rate
Simulation	1	3	4	53.33%
	2	2	4	
	3	2	3	
	4	4	4	
	5	3	3	

In the majority of the test runs, almost all the recessed doors were detected while the same could not be said about the remaining doors.

5.3.4 Discussion

In both the simulations and the flight experiments, the majority of the doors were successfully detected. However, the main difference between them is the following:

- **Simulation:** the doors were detected by both methods (recess and contours) almost all the times.
- **Experiment:** the door contour detection proved to be not so robust, leading to a door detection that was almost totally dependent on the recess detection module.

Despite these differences between simulation and experiment, the developed modules proved to work accordingly and as desired although some incoherences were noticeable. Table 5.9 provides a summary of the 3 experiments conducted.

Table 5.9: Summary of the experiments

	Experiment		Simulation
	#1	#2	#3
Vanishing Point Standard Deviation	[11.47 ; 10.88]	[25.23 ; 10.85]	[18.60 ; 16.91]
Door Detection Success Rate	86.67%	90%	53.33%
Arrival Success Rate	80%	80%	N/A

The experiment with the best overall results was the first one due to its lower *vanishing point standard deviation* and although having an apparent lower *door detection success rate* it is in fact higher as the total number of detectable doors in this experiment is 6 in contrast to only 4 in the second experiment. Sadly no real life data from the third experiment was able to be acquired as previously explained, although even if the drone was able to fly in the narrower corridor, if the simulation results are already the worst of the 3 simulations, the devised test runs for that corridor would not be better. This assumption is based on the observation that for experiment number one and number 2, the test runs always returned worst results than the simulation.

Also, in a vast majority of the experiments, the drone was able to keep aligned with the center of the corridor without colliding with the walls. This proves that the navigation module that was

implemented was working properly by acquiring robustly the vanishing point of each frame and keeping the drone aimed at it.

5.4 Summary

A satisfactory performance of the developed framework was observed for each and every experiment carried out in this dissertation.

During the realization of the experiments it was possible to notice some influence of lighting and texture conditions. The developed method was noted to have a better performance during the day due to the presence of day light instead of artificial light from the lamps in the ceiling of the corridors.

Regarding situations where the developed method is tested in narrow corridors a significant contrast in performance is noticeable since there are less recessed doors and a lot of features that interfere with the detection of the vanishing point. Also, due to the physical limitations of the drone, only the simulations were carried out in that corridor, which lead to a lack of data in order to test the performance of the framework with a real vehicle.

Therefore a failed run in experiment 3 shows that, while the method performs appropriately when the video is previously recorded, the physical dimensions of the corridor and the limitations of the drone take a toll on the result.

However, as it was observable in the simulations, the framework not only works with UAVs but would also be perfectly suitable for ground vehicles in which the video feed would be the same as the one used in the simulations.

Chapter 6

Conclusions

In this dissertation a solution for a real problem was developed and a framework with a modular architecture was proposed. The main goal was to develop a solution which would allow for future upgrades and improvements through the addition of yet more modules.

A deep research has contributed with several ideas and approaches to deal with the problem at hand employing a diversity of existing methods and tools to conquer the challenge that originated from the original problem. During this dissertation a closer look was given at methods such as the Hough Transform and the Canny Edge Detector in order to develop a framework that strongly detects the vanishing point and the doors of a video feed provided by an UAV.

6.1 Results

A series of flight experiments were conducted in order to infer strengths and weaknesses of this solution and the results were satisfactory. Either using pre-recorded videos of a vehicle traversing a corridor, or using a drone to traverse that same corridor, this project turned out to work properly and as expected.

The main research question of this project was answered, as it was in fact possible to enable an UAV to possess a system capable of providing a collision-free navigation, a door detection system and a robust position estimation of both the vehicle and the doors in a floor plan.

Using only the vanishing point, the drone was able to fly with a wall collision-free navigation. When the Drone initially takes-off it will hover indefinitely until it can lock on a stable vanishing point. On the other hand, when in movement, the drone automatically starts hovering if the vanishing point has not been detected for a certain amount of time. This module is constructed so that if the vanishing point is not detected during a certain period of time, the drone will hover and wait until it can lock on the vanishing point once again. However, if that period of time expires, the drone will land. This works as a safety precaution for when someone stands in front of the drone, the lights go out or the drone is near the end of the corridor or facing a wall.

Doors with different formats were successfully identified thanks to the use of two different detection methods, one that detected doors based on their geometric shape (a rectangle) and other

that detected doors based on recesses in the walls. The conjugation and fusion of the results of both these methods led to a detection of different types of doors.

Both the position of the drone and the position of the doors in the floor map were estimated with minimal errors and were quite satisfactory. Although some calibration problems lead in some particular situations to the misplacement of some estimations, the overall performance was favourable.

Issues with both the vanishing point and door detection caused by some existing objects and features in the corridors were observed. However, no critical situations that could render the developed framework useless and inappropriate were encountered.

6.2 Discussion

The developed framework is constructed in such a way that the inclusion of additional modules would be a simple and accessible task. Since each module is independent of one another, sharing just one video feed among each other, it would be as simple as switching a specific module on or off. This feature enables the framework to have a considerable degree of flexibility as it is not bound to a specific type of robot, being versatile to the point of working properly on either aerial or ground vehicles.

All the simulations that were carried out could have been actual tests with a ground vehicle since the video feed would be practically the same. A ground vehicle would have much more stability and would not be susceptible to external interferences like airflows allowing for a more steady and abrupt motion free video feed to be transmitted to the remote computer running the developed framework. Furthermore, the main reason why the processing could not be done on-board an UAV was due to its weight and autonomy limitations. By using a ground vehicle, such constraints would not be met, removing the need for an immovable processing stations. This would lead to a all-in-one on-the-go solution.

Concluding, the proposed method has proven to be flexible and versatile enough in order to be considered a positive asset in applications such as: auto-pilot systems; mobile surveillance; traffic management; domestic, industrial and military applications; and search and rescue missions.

6.3 Future Work

By improving the image processing and detection methods the developed framework will have a better performance. For example, by refining the door detection and vanishing point detection modules, the door detection success rate would increase for corridors without recessed doors and the vanishing point detection would not be susceptible to errors in corridors with poster and boards hanged the walls.

A list of additional features that can be researched in the future is presented bellow:

- Using a drone with a faster data exchange rate would speed up the video input stream processing and therefore making and providing quicker navigation decisions.

- Processing frames and image operations with the GPU instead of the CPU using Nvidia CUDA technology [50] would lead to an improved performance.
- Adding a collision avoidance and person detector module would provide a secure and safe navigation for both the drone and people walking on the same area.
- Using additional hardware (external or internal to the drone), such as range-finders, would provide additional data regarding possible obstacles pillars, windows and foreign objects to the area.
- Adding modules which would allow underwater reckon of shipwrecks and also navigation and mapping using dead reckoning systems.

Conclusively, the developed method supports a variety of possible extensions which are not restricted by applicability in any scenario.

Appendix A

List of QR Codes of the videos

List of QR Codes from the videos presented throughout this dissertation All the videos are hosted on Youtube[1] and can be seen directly on ant browser. The QR codes were created using qrickit[2] and can be used to launch the videos from tablets or smartphones. Possible reader applications include QR Code Reader[3] for Android, QR Code Reader[4] for iOS and QR Code Reader[5] for Windows Phone.

Figure A.1: QR code of Video Stabilization (figure 4.1)

Figure A.2: QR code of Vanishing Point Detection (figure 4.3)

[1] https://www.youtube.com/
[2] http://qrickit.com/qrickit_apps/qrickit_qrcode_creator_url.php
[3] https://play.google.com/store/apps/details?id=me.scan.android.client
[4] https://itunes.apple.com/us/app/qr-code-reader-by-scan/id698925807?mt=8
[5] https://www.windowsphone.com/en-us/store/app/qr-code-reader/e21dee2d-9c1c-4f25-916f-c93d25da8768

Figure A.3: QR code of Floor Segmentation (figure 4.7)

Figure A.4: QR code of Recessed Door Detection (figure 4.10)

Figure A.5: QR code of Door Contours Detection (figure 4.16)

Figure A.6: QR code of Position Estimation (figure 4.20)

Figure A.7: QR code of Simulation #1 (figure 5.4a)

Figure A.8: QR code of Flight Experiment #1 (figure 5.4b)

Figure A.9: QR code of Simulation #2 (figure 5.6a)

Figure A.10: QR code of Flight Experiment #2 (figure 5.6b)

Figure A.11: QR code of Simulation #3 (figure 5.9)

References

[1] Pierre-Jean Bristeau, François Callou, David Vissière, and Nicolas Petit. The Navigation and Control technology inside the AR . Drone micro UAV. *Proceedings of the 18th IFAC World Congress, 2011*, 18:1477–1484, 2011.

[2] Filipe Oliveira Ramos Trocado Ferreira. Video analysis in indoor soccer with a quadcopter. 2014.

[3] John Canny. A computational approach to edge detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, (6):679–698, 1986.

[4] Richard O Duda and Peter E Hart. Use of the hough transformation to detect lines and curves in pictures. *Communications of the ACM*, 15(1):11–15, 1972.

[5] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *Computer vision–ECCV 2006*, pages 404–417. Springer, 2006.

[6] Pezhman Firoozfam and Shahriar Negahdaripour. A multi-camera conical imaging system for robust 3d motion estimation, positioning and mapping from uavs. In *Advanced Video and Signal Based Surveillance, 2003. Proceedings. IEEE Conference on*, pages 99–106. IEEE, 2003.

[7] Hyondong Oh, Dae-Yeon Won, Sung-Sik Huh, David Hyunchul Shim, Min-Jea Tahk, and Antonios Tsourdos. Indoor uav control using multi-camera visual feedback. *Journal of Intelligent & Robotic Systems*, 61(1-4):57–84, 2011.

[8] Andrey Litvin, Janusz Konrad, and William C Karl. Probabilistic video stabilization using kalman filtering and mosaicing. In *Electronic Imaging 2003*, pages 663–674. International Society for Optics and Photonics, 2003.

[9] Yasuyuki Matsushita, Eyal Ofek, Weina Ge, Xiaoou Tang, and Heung-Yeung Shum. Full-frame video stabilization with motion inpainting. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 28(7):1150–1163, 2006.

[10] Kari Saarinen. Color image segmentation by a watershed algorithm and region adjacency graph processing. In *Image processing, 1994. Proceedings. ICIP-94., IEEE international conference*, volume 3, pages 1021–1025. IEEE, 1994.

[11] Carsten Rother, Vladimir Kolmogorov, and Andrew Blake. Grabcut: Interactive foreground extraction using iterated graph cuts. *ACM Transactions on Graphics (TOG)*, 23(3):309–314, 2004.

[12] Liming Wang, Jianbo Shi, Gang Song, and I-fan Shen. Object detection combining recognition and segmentation. In *Computer Vision–ACCV 2007*, pages 189–199. Springer, 2007.

[13] David G Lowe. Object recognition from local scale-invariant features. In *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, volume 2, pages 1150–1157. Ieee, 1999.

[14] Alper Yilmaz, Omar Javed, and Mubarak Shah. Object tracking: A survey. *Acm computing surveys (CSUR)*, 38(4):13, 2006.

[15] Jianbo Shi and Carlo Tomasi. Good features to track. In *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR'94., 1994 IEEE Computer Society Conference on*, pages 593–600. IEEE, 1994.

[16] Gary Bradski and Adrian Kaehler. *Learning OpenCV: Computer vision with the OpenCV library*. " O'Reilly Media, Inc.", 2008.

[17] Dana H Ballard. Generalizing the hough transform to detect arbitrary shapes. *Pattern recognition*, 13(2):111–122, 1981.

[18] Futuhal Arifin, Ricky Arifandi Daniel, and Didit Widiyanto. Autonomous detection and tracking of an object autonomously using ar. drone quadcopter. *Jurnal Ilmu Komputer dan Informasi*, 7(1):11–17, 2014.

[19] John Illingworth and Josef Kittler. A survey of the hough transform. *Computer vision, graphics, and image processing*, 44(1):87–116, 1988.

[20] Cooper Bills, Joyce Chen, and Ashutosh Saxena. Autonomous mav flight in indoor environments using single image perspective cues. In *Robotics and automation (ICRA), 2011 IEEE international conference on*, pages 5776–5783. IEEE, 2011.

[21] Navigation Center general information on gps, *http://www.navcen.uscg.gov/?pageName=gpsmain*, 2015-02-09.

[22] Gyroscopes uses for gyroscopes *http://www.gyroscopes.org/*, 2015-02-09.

[23] Hartmut Surmann, Kai Lingemann, Andreas Nüchter, and Joachim Hertzberg. A 3d laser range finder for autonomous mobile robots. In *Proceedings of the 32nd ISR (International Symposium on Robotics)*, volume 19, pages 153–158, 2001.

[24] Jakob Engel, Jürgen Sturm, and Daniel Cremers. Camera-based navigation of a low-cost quadrocopter. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 2815–2821. IEEE, 2012.

[25] Kinect for windows, *http://www.microsoft.com/en-us/kinectforwindows/*, 2015-02-13.

[26] Stéphane Doncieux, Jean-Baptiste Mouret, Laurent Muratet, and Jean-Arcady Meyer. The robur project: towards an autonomous flapping-wing animat. *Proceedings of the Journes MicroDrones, Toulouse*, 2004.

[27] Robert C Michelson and Steven Reece. Update on flapping wing micro air vehicle research-ongoing work to develop a flapping wing, crawling entomopter. In *13th Bristol International RPV/UAV Systems Conference Proceedings, Bristol England*, volume 30, pages 30–1, 1998.

[28] Guowei Cai, Kai-Yew Lum, Ben M Chen, and Tong Heng Lee. A brief overview on miniature fixed-wing unmanned aerial vehicles. In *Control and Automation (ICCA), 2010 8th IEEE International Conference on*, pages 285–290. IEEE, 2010.

[29] Abraham Bachrach, Samuel Prentice, Ruijie He, and Nicholas Roy. Range–robust autonomous navigation in gps-denied environments. *Journal of Field Robotics*, 28(5):644–666, 2011.

[30] Mohamad Farid bin Misnan, Norhashim Mohd Arshad, and Noorfadzli Abd Razak. Construction sonar sensor model of low altitude field mapping sensors for application on a uav. In *Signal Processing and its Applications (CSPA), 2012 IEEE 8th International Colloquium on*, pages 446–450. IEEE, 2012.

[31] Tomáš Krajník, Matías Nitsche, Sol Pedre, Libor Přeučil, and Marta E. Mejail. A simple visual navigation system for an UAV. In *International Multi-Conference on Systems, Signals and Devices, SSD 2012 - Summary Proceedings*, 2012.

[32] Ariane S Etienne, Roland Maurer, Josephine Georgakopoulos, and Andrea Griffin. Dead reckoning (path integration), landmarks, and representation of space in a comparative perspective. 1999.

[33] Alexandros Lioulemes, Georgios Galatas, Vangelis Metsis, Gian Luca Mariottini, and Fillia Makedon. Safety challenges in using ar. drone to collaborate with humans in indoor environments. In *Proceedings of the 7th International Conference on PErvasive Technologies Related to Assistive Environments*, page 33. ACM, 2014.

[34] Markus Achtelik, Abraham Bachrach, Ruijie He, Samuel Prentice, and Nicholas Roy. Stereo vision and laser odometry for autonomous helicopters in gps-denied indoor environments. In *SPIE Defense, Security, and Sensing*, pages 733219–733219. International Society for Optics and Photonics, 2009.

[35] Albert S Huang, Abraham Bachrach, Peter Henry, Michael Krainin, Daniel Maturana, Dieter Fox, and Nicholas Roy. Visual odometry and mapping for autonomous flight using an rgb-d camera. In *International Symposium on Robotics Research (ISRR)*, pages 1–16, 2011.

[36] James F Roberts, Timothy Stirling, Jean-Christophe Zufferey, and Dario Floreano. Quadrotor using minimal sensing for autonomous indoor flight. In *European Micro Air Vehicle Conference and Flight Competition (EMAV2007)*, number LIS-CONF-2007-006, 2007.

[37] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y Ng. Ros: an open-source robot operating system. In *ICRA workshop on open source software*, volume 3, page 5, 2009.

[38] Learning OpenCV. Computer vision with the opencv library. *GaryBradski & Adrian Kaebler-O'Reilly*, 2008.

[39] Mani Monajjemi. ardrone autonomy: A ros driver for ardrone 1.0 & 2.0, 2012.

[40] James M Coughlan and Alan L Yuille. The manhattan world assumption: Regularities in scene statistics which enable bayesian inference. In *NIPS*, pages 845–851, 2000.

[41] Lucio Marcenaro, Gianni Vernazza, and Carlo S Regazzoni. Image stabilization algorithms for video-surveillance applications. In *Image Processing, 2001. Proceedings. 2001 International Conference on*, volume 1, pages 349–352. IEEE, 2001.

[42] Marius Tico and Markku Vehvilainen. Image stabilization based on fusing the visual information in differently exposed images. In *Image Processing, 2007. ICIP 2007. IEEE International Conference on*, volume 1, pages I–117. IEEE, 2007.

[43] David Stavens. The opencv library: computing optical flow, 2007.

[44] Shih-Ping Liou and Ramesh C Jain. Road following using vanishing points. *Computer vision, graphics, and image processing*, 39(1):116–130, 1987.

[45] Frank A Van Den Heuvel. Vanishing point detection for architectural photogrammetry. *International archives of photogrammetry and remote sensing*, 32:652–659, 1998.

[46] Duncan P Robertson and Roberto Cipolla. An image-based system for urban navigation. In *BMVC*, pages 1–10. Citeseer, 2004.

[47] Greg Welch and Gary Bishop. An introduction to the kalman filter. 2006. *University of North Carolina: Chapel Hill, North Carolina, US*, 2006.

[48] Satoshi Suzuki et al. Topological structural analysis of digitized binary images by border following. *Computer Vision, Graphics, and Image Processing*, 30(1):32–46, 1985.

[49] Mahes Visvalingam and J Duncan Whyatt. The douglas-peucker algorithm for line simplification: Re-evaluation through visualization. In *Computer Graphics Forum*, volume 9, pages 213–225. Wiley Online Library, 1990.

[50] Kari Pulli, Anatoly Baksheev, Kirill Kornyakov, and Victor Eruhimov. Real-time computer vision with opencv. *Communications of the ACM*, 55(6):61–69, 2012.