TECHNICAL UNIVERSITY OF CRETE

LOUKOPOULOS DIPLOMA THESIS

Embedded System for Real-time Detection of Escape Openings Towards Unmanned Aerial Vehicle Navigation

Author:
GEORGIOS LOUKOPOULOS CHATZIGIOSIS

Thesis Committee:
Prof. APOSTOLOS DOLLAS
Prof. MICHAIL LAGOUDAKIS
Prof. PANAGIOTIS PATRTSINEVELOS

A thesis submitted in fulfillment of the requirements for the diploma of Electrical and Computer Engineer

in the

School of Electrical and Computer Engineering
Microprocessor and Hardware Laboratory

February 23, 2023

TECHNICAL UNIVERSITY OF CRETE

Abstract

School of Electrical and Computer Engineering

Electrical and Computer Engineer

Embedded System for Real-time Detection of Escape Openings Towards Unmanned Aerial Vehicle Navigation

by GEORGIOS LOUKOPOULOS CHATZIGIOSIS

Unmanned aerial vehicles (UAVs), also known as drones, have emerged as versatile tools for various applications, including surveillance, inspection, delivery, and search and rescue missions. The increasing popularity of UAVs has led to a growing demand for advanced autonomous navigation systems to enable them to perform complex tasks independently. Autonomous navigation refers to the ability of UAVs to navigate and maneuver without human intervention, relying on onboard sensors, algorithms, and decision-making processes.

Two of the critical tasks in UAV autonomous navigation is localization, which refers to determining the drone's position and orientation relative to its environment, and mapping which is the ability of the drone to build a map of its surroundings. Another important aspect is the detection of escape openings, which are critical in emergency situations, allowing the UAV to navigate through narrow or complex environments.

In this work, we present an approach that addresses the challenges of localization, mapping, and escape opening detection in UAV autonomous navigation, while minimizing cost, energy consumption, and resource usage. Our solution leverages the power of Computer Vision algorithms and sensors, enabling accurate and robust localization and mapping, as well as an efficient escape opening detection.

ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ

Περίληψη

Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών

Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών

Ενσωματωμένο Σύστημα Πραγματικού Χρόνου για Εύρεση Ανοιγμάτων Διαφυγής με Σκοπό την Πλοήγηση Μη Επανδρωμένου Εναέριου Οχήματος

Λουκοπουλος Χατζηγιωσης Γεωργιος

Τα μη επανδρωμένα αεροσκάφη (UAV's), γνωστά και ως drones, αποτελούν πολύ χρήσιμα εργαλεία για διάφορες εφαρμογές, συμπεριλαμβανομένων της παρακολούθησης, του ελέγχου, της παράδοσης και των αποστολών έρευνας και διάσωσης. Η αυξανόμενη ενσωμάτωση των UAVs έχει οδηγήσει σε μια αυξανόμενη ζήτηση για προηγμένα συστήματα αυτόνομης πλοήγησης. Η αυτόνομη πλοήγηση αναφέρεται στην ικανότητα των UAV's να πλοηγούνται και να μανοβράρουν χωρίς ανθρώπινη παρέμβαση, βασιζόμενα σε αισθητήρες, αλγόριθμους και διαδικασίες λήψης αποφάσεων που βρίσκονται στο εσωτερικό τους.

Δύο από τις πιο κρίσιμες προκλήσεις στην αυτόνομη πλοήγηση των UAVs είναι η τοποθέτηση, και ο χαρτογραφικός προσανατολισμός, που αφορά τη δυνατότητα του δρονε να δημιουργήσει ένα χάρτη του περιβάλλοντός του προκειμένου να πλοηγηθεί σε αυτόν. Ένα άλλο σημαντικό στοιχείο είναι η ανίχνευση των ανοιγμάτων διαφυγής, τα οποία είναι κρίσιμα σε επείγουσες καταστάσεις, επιτρέποντας στο UAV να πλοηγηθεί μέσα από στενά ή πολύπλοκα περιβάλλοντα.

Σε αυτή την εργασία, παρουσιάζουμε μια προσέγγιση που αντιμετωπίζει τις προκλήσεις της τοποθέτησης, του χαρτογραφικού προσανατολισμού και της ανίχνευσης ανοιγμάτων απόδρασης στην αυτόνομη πλοήγηση των UAVs , επιχειρώντας την ελαχιστοποίηση το κόστους, της κατανάλωσης ενέργειας και της υπολογιστικής ισχύος ενσωματώνοντας αλγορίθμους μηχανικής όρασης και αισθητήρες, διασφαλίζοντας ακριβή και ανθεκτική τοποθέτηση και χαρτογράφηση, καθώς και αποδοτική ανίχνευση ανοιγμάτων απόδρασης.

Acknowledgements

I am deeply thankful to my supervisor, Professor Apostolos Dollas, for his unwavering guidance, support, and assistance in completing my Dissertation. I am also extremely grateful to Professor Panagiotis Partsinevelos, Mr. Angelos Antonopoulos and the rest of the Sense Lab team for their inspiration, contributions, and provision of equipment. Lastly, I thank my loved ones, family and friends, for their constant encouragement and love throughout this whole journey.

Contents

List of Figures

List of Abbreviations

ALU	**A**rithmetic **L**ogic **U**nit
ASIC	**A**pplication **S**pecific **I**ntegrated **C**ircuit
BRAM	**B**lock **R**andom **A**ccess **M**emory
CPU	**C**entral **P**rocessor **U**nit
CS	**C**omputer **S**cience
DDR4	**D**ouble **D**ata **R**ate type texbf4 memory
DRAM	**D**ynamic **R**andom **A**ccess **M**emory
DSP	**D**igital **S**ignal **P**rocessor
FF	**F**lip **F**lops
FPGA	**F**ield **P**rogrammable **G**ate **A**rray
GDDR6	**G**raphics **D**ouble **D**ata **R**ate type **6** memory
GPU	**G**raphic **P**rocessor **U**nit
HBM	**H**igh **B**andwidth **M**emory
HDL	**H**ardware **D**escription **L**anguage
HLS	**H**igh **L**evel **S**ynthesis
HPC	**H**ight **P**erformance **C**omputing
LUT	**L**ook **U**p **T**able
MPSoC	**M**ulti **P**rocessor **S**ystem **o**n **C**hip
PL	**P**rogrammable **L**ogic
PS	**P**rocessing **S**ystem
RAM	**R**andom **A**ccess **M**emory
SDK	**S**oftware **D**evelopment **K**it
SIMD	**S**ingle **I**nstruction **M**ultiple **D**ata
SSE	**S**treaming **S**IMD **E**xtensions
SSD	**S**olid **S**tate **D**rive
TDP	**T**hermal **D**esign **P**ower
URAM	**U**ltra **R**andom **A**ccess **M**emory
USD	**U**nited **S**tates **D**ollar

Dedicated to my family and friends…

Chapter 1

Introduction

Looking at how living beings operate to navigate in an unknown space and what stimulus the brain requires to make decisions about the next move helps us understand how to approach a solution on how robots would achieve the same goal. Being aware of the surrounding space is the first and most crucial prerequisite for navigation. Furthermore, the ability to detect several other attributes (doors, windows) leads to improved movement flexibility (moving along rooms), extra information about the environment, and better spatial perception.

Simultaneous Localization and Mapping (SLAM) is the computational problem of constructing or updating a map of an unknown environment while simultaneously keeping track of an agent's location within it [1]. The research on SLAM dates back to 1986, when the idea of using estimation-theoretic methods for robot localization and mapping were first discussed in IEEE Robotics and Automation Conference held in San Francisco.

Today, many variations of SLAM have been introduced by researchers regarding different ways of approaching SLAM problems depending on the conditions of each problem.

1.1 Motivation and Scientific Contributions

In UAV autonomous navigation, a Visual SLAM system leverages 3D vision to perform location and mapping functions when neither the environment nor the location of the sensor is known.It works by tracking set points through successive camera frames to triangulate their 3D position, while simultaneously using this information to approximate camera pose. Basically, the goal of these systems is to map their surroundings in relation to their own location for the purposes of navigation. This, can provide a basic navigation and path planning system.

Though, when it comes to buildings inspection where more complex environments need to be traversed and analysed, the SLAM output may not be enough. The ability of the system to recognize several features inside a 3D space can lead in a more robust navigation.

In this study, we propose:

- An additional feature to the navigation system that would have to do with detecting escape openings inside the surrounding indoor environment providing the system with the ability to navigate through different spaces or escaping through narrow openings on emergency situations. Also another challenge of the work was to try to minimze the systems energy workload, the hardware modules used and the required computational resources.
- This is achieved by integrating an Optical Sensor, an embedded system and a Flight Controller and developing and utilizing SLAM and other Computer Vision Algorithms in order to obtain the required data for the autonomous navigation .
- The system was tested by assembling a real custom-made drone under realistic circumstances inside buildings in Technical University of Crete and the results are based on these datasets.
- The Results are sufficiently accurate regarding the localization and mapping, especially considering the hardware modules utilized and the integrated algorithms. The Opening Detection feature provides accurate and efficient identification of escape routes in complex environments. However, it's important to note that the effectiveness of this feature may be limited in certain circumstances, such as low light and obscured environments.

1.2 Thesis Outline

- **Chapter 2 - Theoretical Background:** In this chapter, we aim to provide a comprehensive explanation of the basic terms and theories that are essential for our work. The purpose of this chapter is to lay the foundation for a clear understanding of the concepts that are relevant to our research. We will delve into the relevant literature and present the most important theories that have shaped our understanding of the topic.

- **Chapter 3 - Related Work:** In this chapter, we present an overview of the related work in our field of study. Our goal is to provide a comprehensive overview of the various approaches and strategies that have been used to address similar problems or research questions. By examining the previous work, we aim to highlight the similarities and differences between our study and other studies in the field. We will also analyze the different strategies that we took and explain why we chose these strategies. This chapter is an important part of our research as it provides insights into the current state of the field and sets the stage for the next chapters.
- **Chapter 4 - Theoretical Modeling:** In this chapter, we delve into the details of the methods and theoretical background that we used to approach the problems of this work. The purpose of this chapter is to provide a comprehensive explanation of our approach and the reasoning behind our choices. We will present the mathematical models and algorithms that we used to solve the problems and describe the steps that we took to integrate these models.
- **Chapter 5 - System Level Design:** In this chapter, we focus on the design and development of the system. We analyze the hardware and tools that we used to build the system and describe how we integrated them to develop our final solution. The purpose of this chapter is to provide a comprehensive explanation of the hardware and software components that we used to build the system, and how we integrated them to create a complete and functional system. We will provide detailed descriptions of the hardware and software components, including the specifications, features, and design trade-offs. We will also describe the process of integrating these components, including the challenges that we faced and how we overcame them.
- **Chapter 6 - Results:** In this chapter, we present and analyze the results of our work. The purpose of this chapter is to provide a comprehensive evaluation of the performance of the system and to showcase the results of our efforts. We will present the data and metrics that we used to evaluate the system, including experiments and tests that we

conducted. We will also provide a detailed analysis of the results, including the strengths and weaknesses of our system and any insights that we gained from the data.

- **Chapter 7 - Conclusions and Related Work:** In this chapter, we summarize the main findings and conclusions of our work and discuss future directions for further research. The purpose of this chapter is to reflect on the results of our work and to provide insights into how our work contributes to the field. We will summarize the main contributions of our work, including the results and performance of the system. We will also discuss the limitations of our work and any areas for improvement. Finally, we will present our vision for future work in the field, including any potential avenues for further research and improvement.

Chapter 2

Theoretical Background

The objective of this chapter is to furnish a detailed elucidation of the fundamental terms and theories that are crucial for our study. Its aim is to establish the basis for a lucid comprehension of the ideas that pertain to our research. We will examine the pertinent literature and highlight the key theories that have influenced our perception of the subject matter.

2.1 Aircraft

An aircraft is a vehicle or machine that is able to fly by gaining support from the air. It counters the force of gravity by using either static lift or by using the dynamic lift of an airfoil, or in a few cases the downward thrust from jet engines. Common examples of aircraft include airplanes, helicopters, airships (including blimps), gliders, paramotors, and hot air balloons [2].

FIGURE 2.1: Cessna 172 Skyhawk

Source: www.nycaviation.com

2.2 Unmaned Aerial Vehicles

An unmanned aerial vehicle (UAV), commonly known as a drone, is an aircraft without any human pilot, crew, or passengers on board. UAVs are a component of an unmanned aircraft system (UAS), which includes adding a ground-based controller and a system of communications with the UAV. The flight of UAVs may operate under remote control by a human operator, as remotely-piloted aircraft (RPA), or with various degrees of autonomy, such as autopilot assistance, up to fully autonomous aircraft that have no provision for human intervention.[3]

FIGURE 2.2: Mavic-3: www.dji.com.

Source: www.dji.com

2.3 Inertial Measurement Units (IMUs)

An inertial measurement unit (IMU) is an electronic device that measures and reports a body's specific force, angular rate, and sometimes the orientation of the body, using a combination of accelerometers, gyroscopes, and sometimes magnetometers.. An IMU system is essential as it comes to flying or generally maneuvering for the reason that it provides the pilot or the path planing system vital information regarding the attitude of the aircraft real-time. Modern IMUs provides us with three-dimensional measurements. A modern IMU typically consists of:

- **Accelerometers:** A 3-axis accelerometer measures linear acceleration along the three principle axis X (longitudinal) Y(lateral) and Z(vertical). The corresponding rotations are called roll, pitch and yaw

- **Gyroscopes:** A three axis gyroscope measures the angular velocity of its rotation in a reference frame along its three sensitivity axes. For an ideal sensor, these outputs are equal to the projections of the rotation angular velocity on the sensitivity, that is, intrinsic coordinate system axes..
- **Magnetometers:** Three axis magnetometers provide orientation data to the magnetic north pole.

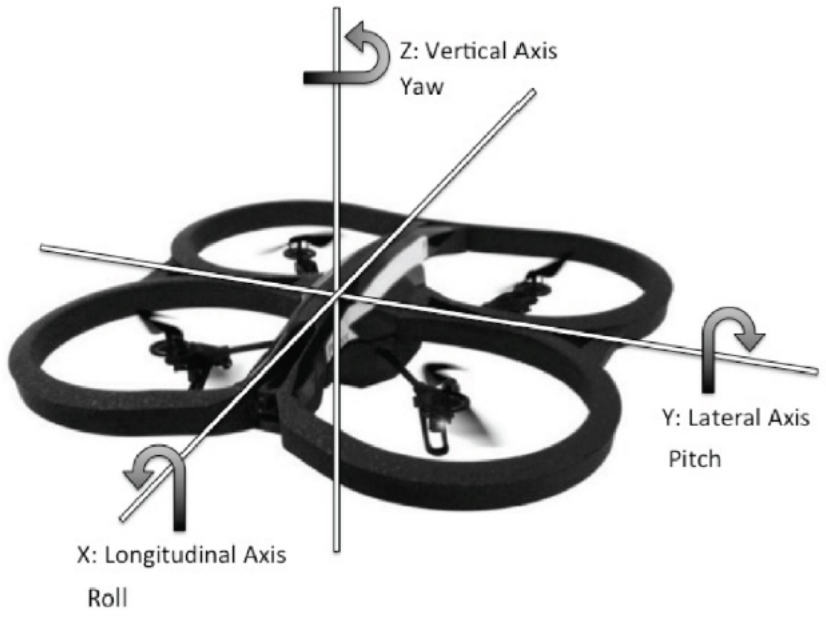


FIGURE 2.3: The principal axis

Source: A.R. Parrot 2.0

2.4 Visual Simultaneous Localization and Mapping (vSLAM) algorithm

Simultaneous localization and mapping (SLAM) is one of the most fundamental, yet most challenging problems in mobile robotics. To achieve full autonomy a robot must possess the ability to explore its environment without user-intervention, build a reliable map, and localize itself in the map. In particular, if global positioning sensor (GPS) data and external beacons are unavailable, the robot must somehow, by itself, determine what are appropriate reference points, on which to build a map. [4]

The vSLAM algorithm uses odometry data and image feed in order to calculate the robots pose. This is being achieved by recognizing specific landmarks using computer vision algorithms, estimating their distance and by fusing the odometry data,constantly updating and correcting its pose and map of its surroundings.

The inputs to the system are odometry and images, and the output is the robot pose and an abstract vSLAM map.

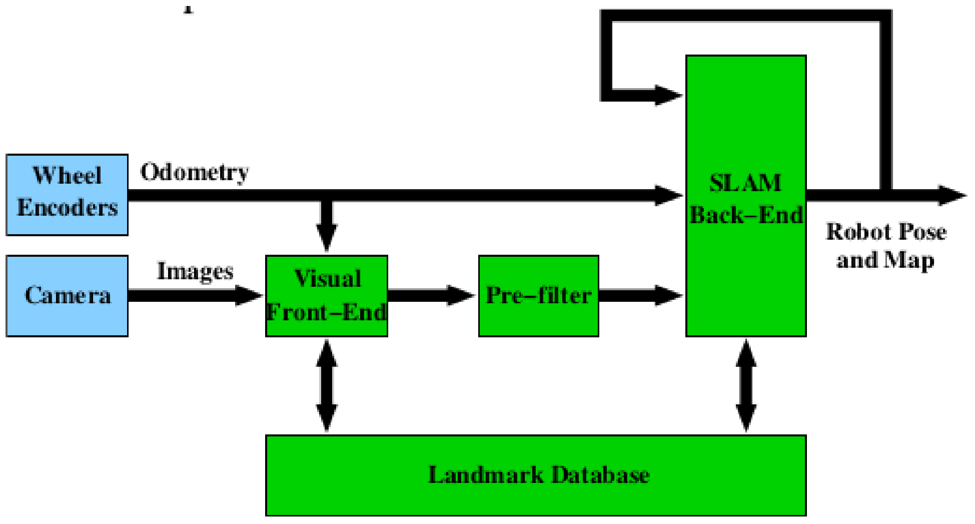


FIGURE 2.4: Block diagram of vSLAM

Source: [4]

The visual landmarks are "created" along the path of the robot. Each landmark is associated with a landmark pose, defined as the robot pose (x, y, and θ) when the landmark was created. In the first module of vSLAM, the images are processed and compared with previously created landmarks. If a recognition is made, an estimate is computed on where the robot is located relative to where it was located when the landmark was created. If a recognition is not made, the module attempts to create a new visual landmark.If a visual landmark was recognized in the Visual Front- end and a visual measurement is computed, then the second module, the Pre-filter, assess the reliability of the measurement. If the measurement is considered unreliable, it is rejected and thrown away. However, if it is accepted, it is used as an input to the SLAM module. The SLAM module is a feedback system taking odometry data and relative pose measurements as inputs, to be fused and compared to a current map. First the robot pose is estimated based on the most up-to-date map and the two inputs. Then, the map is updated to reflect the new information.[4]

2.5 Stereo Vision

Stereo vision is a triangulation-based range-finding technique in which two (or more) cameras are used to reconstruct the three-dimensional structure of a scene, as illustrated in figure 2.5.. The fundamental computational problem in stereo is the *correspondence problem*, which requires finding *correspondence points* p_l and p_r in the two images. Given such points and the relative geometry of the cameras, it is a simple matter to compute the depth, or the distance from the cameras, of the associated world point P. In principle, we can find the distance to every point in the scene by finding the corresponding point in the right image for every pixel in the left image. The resulting representation of scene depth at every pixel in the image, known as a depth map, is a starting point for computing a 3-D model of the scene.[5]

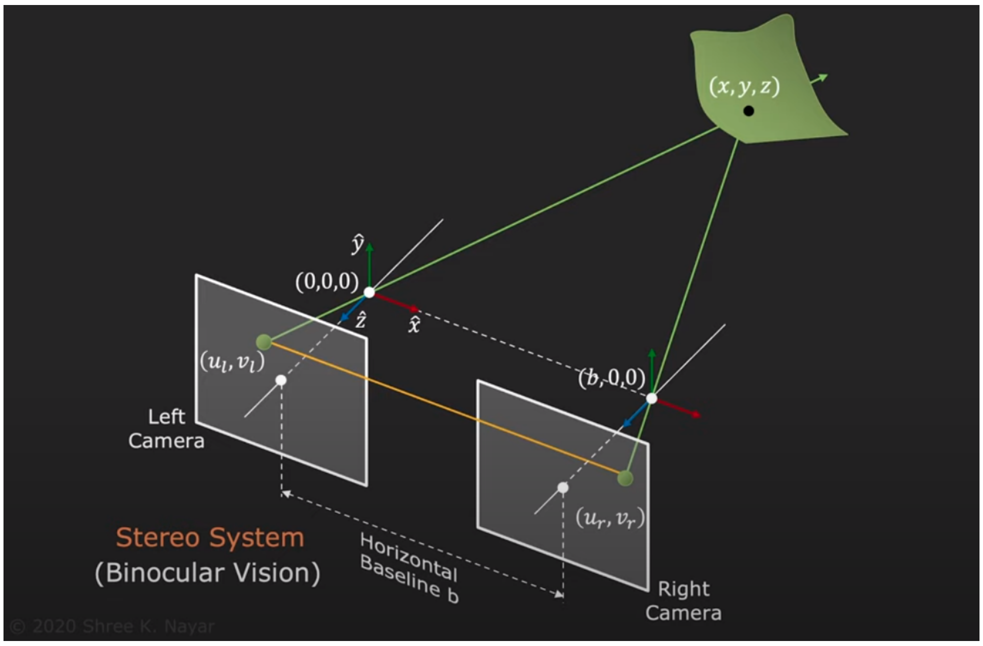


FIGURE 2.5: Stereoscopic Vision

Source: First Principles of Computer Vision

2.6 Stereo Block Matching

In order to deal with the *correspondence problem*, many approaches have been developed. The basic idea of Stereo Block Matching is to segment the target image into fixed size blocks and find for each block the corresponding block that provides the best match from the reference image as shown in figure 2.6. The distance along the X-Axis between the matching block in the left image and the one in the right image is called Disparity. Depth at any given point can be computed if the disparity at that point is known. Disparity measures the displacement of a point between two images. The higher the disparity, the closer the object.

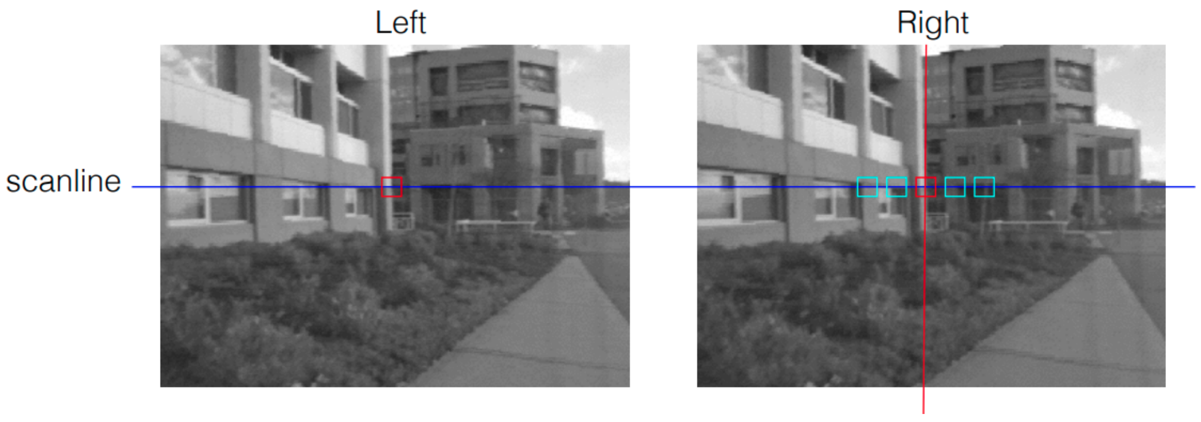


FIGURE 2.6: Stereo Block Matching

Source: www.cs.cornell.edu/

Disparity estimation algorithms fall into two broad categories: local methods and global methods. Local methods evaluate one pixel at a time, considering only neighboring pixels. Global methods consider information that is available in the whole image. Local methods are poor at detecting sudden depth variation and occlusions, and hence global methods are preferred. Semi-global matching uses information from neighboring pixels in multiple directions to calculate the disparity of a pixel. Analysis in multiple directions results in a lot of computation. Instead of using the whole image, the disparity of a pixel can be calculated by considering a smaller block of pixels for ease of computation. Thus, the Semi-Global Block Matching (SGBM) algorithm uses block-based cost matching that is smoothed by path-wise information from multiple directions.[6]

2.7 SGBM Algorithm

The SGBM algorithm takes a pair of rectified left and right images as input. The pixel data from the raw images may not have identical vertical coordinates because of slight variations in camera positions. Images need to be rectified before performing stereo matching to make all epi-polar lines parallel to the horizontal axis and match vertical coordinates of each corresponding pixel.

The SGBM algorithm implementation has three major modules: Matching Cost Calculation, Directional Cost Calculation and Post-processing.

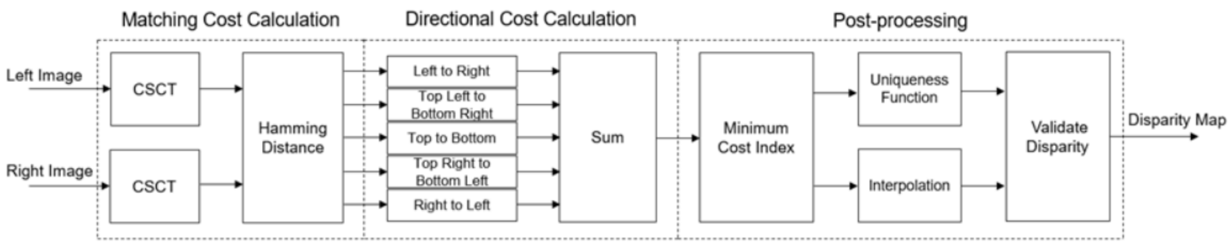


FIGURE 2.7: SGBM block diagram

Source: docs.opencv.org

2.7.1 Matching Cost Calculation

On this stage , at first the model computes the **Center-Symmetric Census Transform** on each of the left and right images using a sliding window. For a given pixel. CSCT for the center pixel in that window is estimated by comparing the value of each pixel with its corresponding center-symmetric counterpart in the window. If the pixel value is larger than its corresponding center-symmetric pixel, the result is 1, otherwise the result is 0.

Afterwards, In the **Hamming Distance module**, the CSCT outputs of the left and right images are pixel-wise XOR'd and set bits are counted to generate the matching cost for each disparity level. To generate D disparity levels, D pixel-wise Hamming distance computation blocks are used. The matching cost for D disparity levels at a given pixel position, p, in the left image is computed by computing the Hamming distance with (p to D+p) pixel positions in the right image. The matching cost, C(p,d), is computed at each pixel position, p, for each disparity level, d. The matching cost is not computed for pixel positions corresponding to the first D columns of the left image.[6]

2.7.2 Directional Cost Calculation

The second module of SGBM algorithm is directional cost estimation. In general, due to noise, the matching cost result is ambiguous and some wrong matches could have lower cost than correct ones. Therefore additional constraints are required to increase smoothness by penalizing changes of neighboring disparities. This constraint is realized by aggregating 1-D minimum cost paths from multiple directions. It is represented by aggregated cost from r directions at each pixel position, $S(p,d)$, as given by[6]

$S(p,d) = \sum_r L_r(p,d)$

The 1-D minimum cost path for a given direction, $L_r(p,d)$, is computed as shown in the equation.

$L_r(p,d) = C(p,d) + min(L_r(p-r,d), L_r(p-r,d-1) + P1, L_r(p-r,d+1)) + minL_r((p-r,i) + P2) - minL_r(p-r,k)$

where

$L_r(p,d)$ = current cost of pixel p and disparity d in direction r

$C(p,d)$ = matching cost at pixel p and disparity r

$L_r(p-r,d-1)$ = previous cost of pixel in r direction at disparity $d-1$

$L_r(p-r,d+1)$ = previous cost of pixel in r direction at disparity $d+1$

$min_i L_r(p-r,i)$ = minimum cost of pixel in r direction for previous computation

$P1, P2$ = penalty for discontinuity

2.7.3 Post Processing

The third module of SGBM algorithm is Post-processing. This module has three steps: minimum cost index calculation, interpolation, and a uniqueness function. Minimum cost index calculation finds the index corresponding to the minimum cost for a given pixel. Sub-pixel quadratic interpolation is applied on the index to resolve disparities at the sub-pixel level. The uniqueness function ensures reliability of the computed minimum disparity. A higher value of the uniqueness threshold marks more disparities unreliable. As a last step, the negative disparity values are invalidated and replaced with -1.[6]

2.8 Canny Edge Detector

The Canny Edge Detector is an edge detection operator that uses a multi-stage algorithm to detect a wide variety of edges in images. It was developed in 1986 by John F. Canny [7]. With the use of the Canny edge detection technology, the amount of data that needs to be processed can be drastically reduced while still extracting meaningful structural information from various vision objects. It is frequently used in many computer vision systems. According to Canny, the prerequisites for applying edge detection to various vision systems are largely the same. Thus, a solution for edge detection that meets these needs can be used in a variety of contexts. The standard parameters for edge detection include of:

- Edge detection that has a low error rate, which implies that it should correctly identify as many edges as possible in the picture.
- The operator's edge point detection should precisely locate on the edge's center.
- If at all feasible, picture noise should not produce false edges, and an edge in the image should only be marked once.

Canny utilized the calculus of variations, a method for locating the function that best optimizes a given functional, to fulfill these objectives. The best fit for Canny's detector is given by the sum of four exponential components, however the first derivative of a Gaussian can be used to approximate it.

The process of Canny edge detection algorithm can be broken down to five different steps:

2.8.1 Gaussian Filter

Since edge detection is susceptible to noise in the image, first step is to remove the noise in the image by convolving the image with a Gaussian kernel. This step will slightly smooth the image to reduce the effects of obvious noise on the edge detector. The equation for a Gaussian filter kernel of size $(2k+1) \times (2k+1)$ is given by:

$H_{ij} = \frac{1}{2\pi\sigma^2} exp(-\frac{(i-(k+1))^2+(j-(k+1))^2}{2\sigma^2})$

2.8.2 Finding Intensity Gradient of the Image

Smoothened image is then filtered with a Sobel kernel in both horizontal and vertical direction to get first derivative in horizontal direction (Gx) and vertical direction (Gy). From these two images, we can find edge gradient and direction for each pixel as follows:

$Edge\_Gradient(G) = \sqrt{G^2x + G^2y}$

$Angle(\theta) = \tan^{-1}(\frac{G_y}{G_x})$

Gradient direction is always perpendicular to edges. It is rounded to one of four angles representing vertical, horizontal and two diagonal directions.

2.8.3 Non-maximum Suppression

Once the gradient's magnitude and direction have been determined, the entire image is scanned to set apart any extraneous pixels that might not be the edge. This is accomplished by checking each pixel to see if a local maximum exists in the area around it in the gradient's direction.

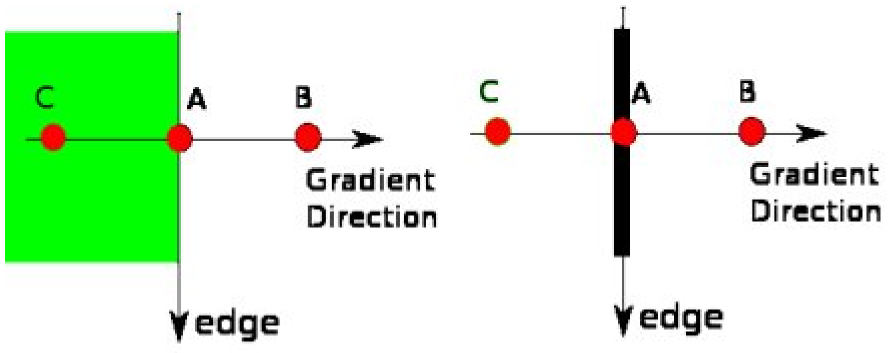


FIGURE 2.8: Non-maximum Suppression

Source: docs.opencv.org

Point A is on the edge (in vertical direction). Gradient direction is normal to the edge. Point B and C are in gradient directions. So point A is checked with point B and C to see if it forms a local maximum. If so, it is considered for next stage, otherwise, it is suppressed (put to zero). The result of this stage is to obtain "thin" edges

2.8.4 Hysteresis Thresholding

At this stage, it is determined which edges are actually edges and which ones are not.There are two required values for this stage. minVal and maxVal. Any edges with gradients of intensity greater than maxVal are certain to be edges, and any below minVal are certain to be non-edges, so they should be discarded. Based on their connectivity, those who fall between these two thresholds are categorized as edges or non-edges. They are regarded as being a part of edges if they are linked to "sure-edge" pixels. If not, they are also thrown away.[8]

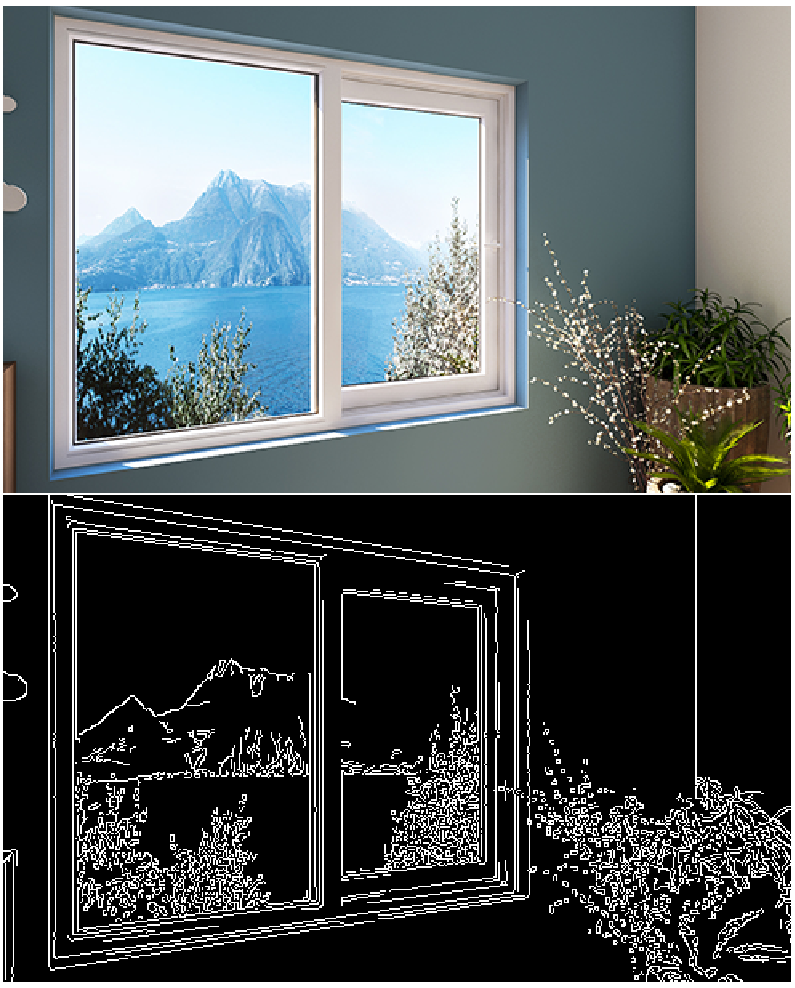


FIGURE 2.9: Canny Edge Detector

2.9 Hough Line Transform

The **Hough transform** is a feature extraction technique used in image analysis, computer vision, and digital image processing. The purpose of the technique is to find imperfect instances of objects within a certain class of shapes by a voting procedure. This voting procedure is carried out in a parameter space, from which object candidates are obtained as local maxima in a so-called accumulator space that is explicitly constructed by the algorithm for computing the **Hough transform** [9].

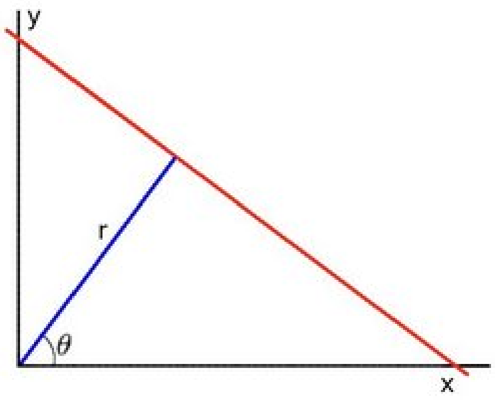
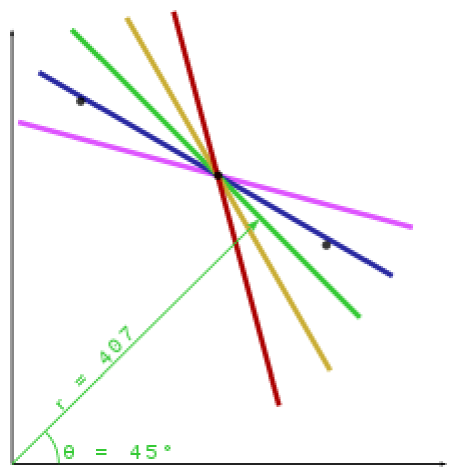


FIGURE 2.10: (r,theta) line representation

Source: docs.opencv.org

The simplest case of Hough transform is detecting straight lines. The linear Hough transform algorithm estimates the two parameters that define a straight line. The transform space has two dimensions, and every point in the transform space is used as an accumulator to detect or identify a line described by $r = x\cos\theta + y\sin\theta$ 2.10. Every point in the detected edges in the image contributes to the accumulators 2.11.



θ	r
15	318.5
30	376.8
45	407.3
60	409.8
75	385.3

FIGURE 2.11: Accumulators

Source: docs.opencv.org

Taking into account the quantized values of r and in the pair (r, θ), the dimension of the accumulator is equal to the number of unknown parameters, or two. The Hough transform method assesses if there is sufficient proof of a straight line at each pixel at (x, y) and its surrounding areas. If so, it will determine the (r, θ) parameters of that line, then look for the accumulators bin where the parameters lie, and increase the value of that bin. The most likely lines can be retrieved, together with their geometric definitions, by seeking for the bins with the greatest values, often by searching for local maxima in the accumulator space. By using a threshold, which controls which lines are detected and how many, it is easiest to locate these peaks. Finding which parts of the image correspond to which lines is frequently important in the following step because the lines returned do not contain any length information. The output of the linear Hough transform is a two-dimensional array (matrix) is similar to the accumulator, where one dimension is made up of the quantized angle θ.and the other is made up of the quantized distance r. The value of each matrix component is equal to the total of the points or pixels that are situated along the line denoted by the quantized parameters

(r, θ). Accordingly, the element with the highest value represents the straight line that appears the most in the input image.

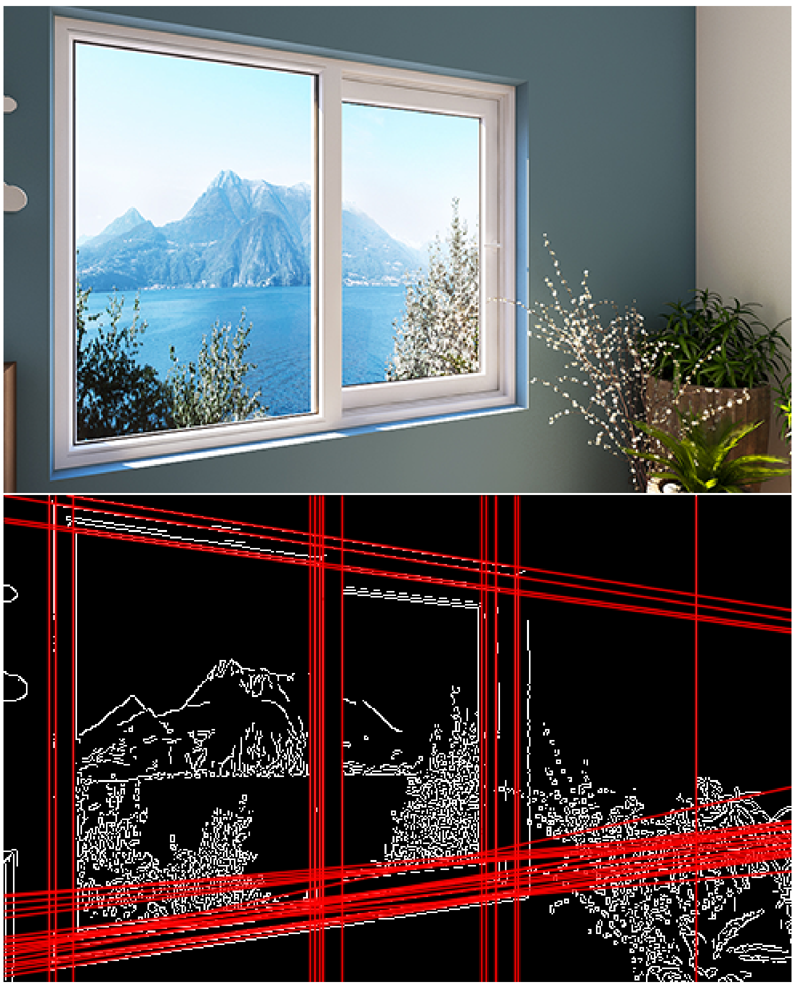


FIGURE 2.12: Hough Line Transform

Chapter 3

Related Work

On this chapter, we aim to present a thorough overview of the different techniques and methods that have been employed to tackle comparable issues or research queries. Through an examination of past research, we endeavor to underscore the similarities and divergences between our study and others in the field. We will also scrutinize our own approach and explicate the reasoning behind our chosen strategies. This chapter holds a crucial position in our research as it furnishes insights into the present condition of the field and establishes the foundation for the ensuing chapters.

3.1 Indoor UAV Navigation

While autonomous UAV navigation on indoor environments provides many capabilities to UAV's and offers sollutions to very important problems, many approaches have been developed in order to achieve sufficient results. Regarding the environment circumstances, prior knowledge and target use of the system, different approaches and technologies can be integrated to achieve the best possible results. The most common approaches are Vision-based combined with inertial sensor localization, sollutions based on Lidar integratrion and finally by utilizing SLAM through stereo or mono optical sensors.

3.1.1 Vision-based and inertial sensor-based localization approaches

The environment simply has to be minimally instrumented for vision-based approaches, which merely need to employ affordable, everyday cameras as perception sensors. The latest 3D tracking methods in dynamic contexts, such building sites, were developed by 2D monitoring the same objects of interest in two or more cameras and 3D triangulating the tracked 2D observations[10]. These solutions, however, rely on permanent cameras that are placed in locations with wide baselines, require complicated calibration procedures, have a fixed field of vision, and are inevitably obstructed by objects like equipment or temporary constructions.

3.1.2 Lidar-based SLAM

Lidar-based SLAM has drawn more and more attention from researchers, particularly those who are interested in localization and mapping in dynamic interior situations like construction sites, because of its excellent accuracy and an ever-increasing number of open source implementations.

This work [11] presents a method for generating and visualizing floor plans in real-time using portable laser range-finders (LIDAR) and simultaneous localization and mapping (SLAM). The authors demonstrate the effectiveness of their approach on a backpack mapping platform that achieves real-time mapping and loop closure at a 5 cm resolution while operating under limited computational resources. To achieve real-time loop closure, the paper proposes a branch-and-bound approach for computing scan-to-submap matches as constraints. Experimental results and comparisons to other established techniques show that the proposed approach is competitive in terms of quality. In comparison to our works goal, in order for a drone to navigate effectively requires a 3D representation instead of 2D.

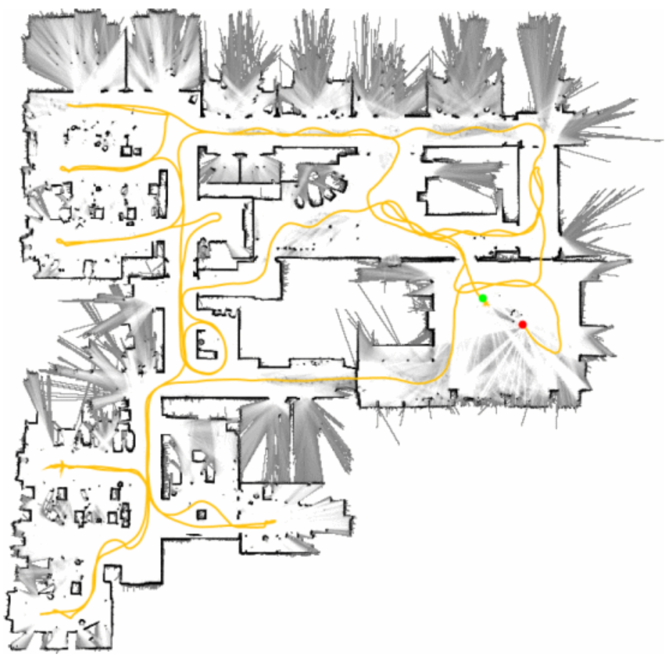


FIGURE 3.1: 2D loo[closure Lidar SLAM]

Source: [11]

Another study's [12] main goal is to create a highly automated system that can produce an ideal 3D-reconstructed map for the construction industry with a minimal amount of static scanning and little processing time. The system's output is a fully registered point cloud with color information. The architecture 3.4 proposed consists of five 2D Lidars in specific layout in order to obtain 3D information, a DSLR camera for adding texture to the pointcloud, object avoidance sensors and an IMU for localization. A SLAM algorithm makes use of the laser scan data from the horizontal LiDAR to determine the position and orientation of the mobile robot on the horizontal plane. In order to obtain a pose estimation and a planar map of the environment, the Hector SLAM algorithm was integrated in this study to perform laser scan matching between the most recent LiDAR scan and an incrementally-built map. The translation and rotation parameters of the current scan are obtained for each set of raw scan data from the horizontal LiDAR acquired from a 2D single planar sweep by performing scan matching with the most recent estimated map of the environment 3.3.

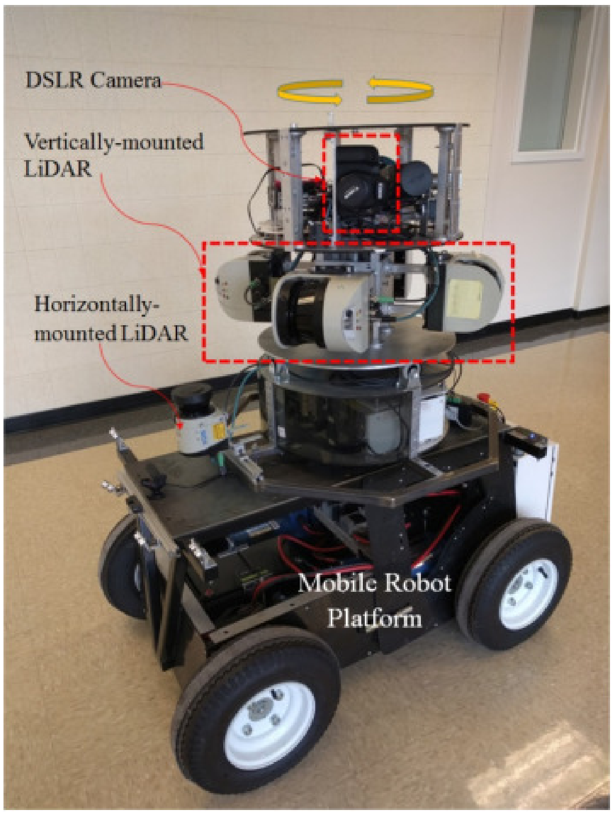
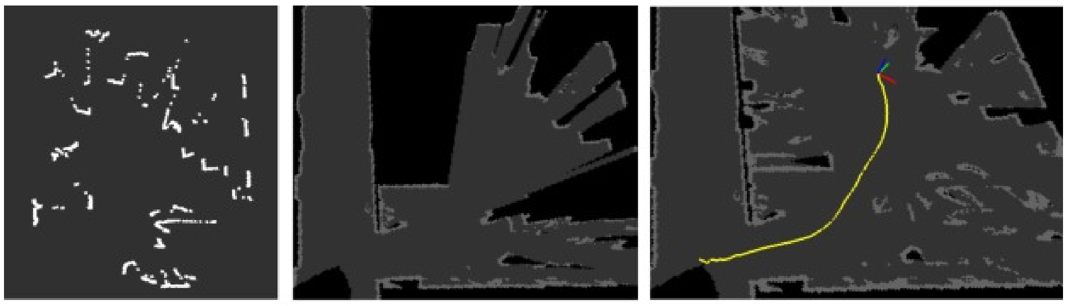


FIGURE 3.2: Architecture

Source: [12]



(a) raw scan data from LiDAR

(b) incrementally-built 2D map of the environment

(c) final recovered trajectory with position and orientation of robot

FIGURE 3.3: Mapping

Source: [12]

3.1.3 vSLAM and RGBD based approaches

In general, vSLAM is more affordable than laser scanners since it uses one or more cameras as the primary perception sensors. It also offers competitive localization performance on a variety of datasets. Another advantage is that, even if extra odometry or IMU input can aid further increase accuracy and robustness, vSLAM can operate with simply frame observations. This suggests that the only necessary hardware setup for utilizing a vSLAM solution is to place one or more common cameras on an existing mobile platform or the tracked objects.

The paper [13] discusses the potential of using drones for indoor farming and livestock management, which is limited by the challenges and constraints of GPS unavailability and limited physical space. The study compares two visual simultaneous localization and mapping algorithms using a monocular camera onboard a small drone to investigate drone positioning in these indoor workspaces. The results show that ORB-SLAM algorithm is the most suitable for these workspaces, allowing for aerial VSLAM and potential benefits in plant and cattle monitoring. However, for the current work that requires a precise 3D representation, a different approach may be required.

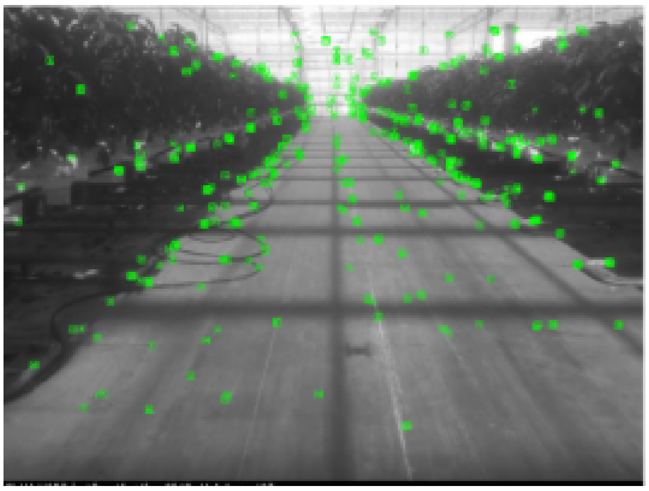


FIGURE 3.4: Monocular SLAM

Source: [13]

Stereo cameras are commonly used to extract sparse distance information from the disparity of the textured regions of each image. In contrast to his SLAM, which is laser-based, Visual SLAM systems typically extract sparse keypoints from camera images. Visual landmarks have the advantage of being more prominent than typical geometric structures and simplify data mapping. Common generic keypoint detectors and descriptors include SIFT, SURF and ORB [14]. In his research, [15] Shinya Kawabata proposes a RGBD based SLAM system for indoor infrastructure inspection deploying it in a custom made UAV. The system includes a flight controller embedded with IMUsensors, four motors to rotate propellers, four drivers so-called ESC to provide power to motors according to control command from flight controller, companion computer, depth camera, and receiver of wireless communication and so on. The depth camera is connected to the companion computer which manipulates 3D depth information and computes vehicle position with SLAM algorithm. For providing stable position information of drone itself,

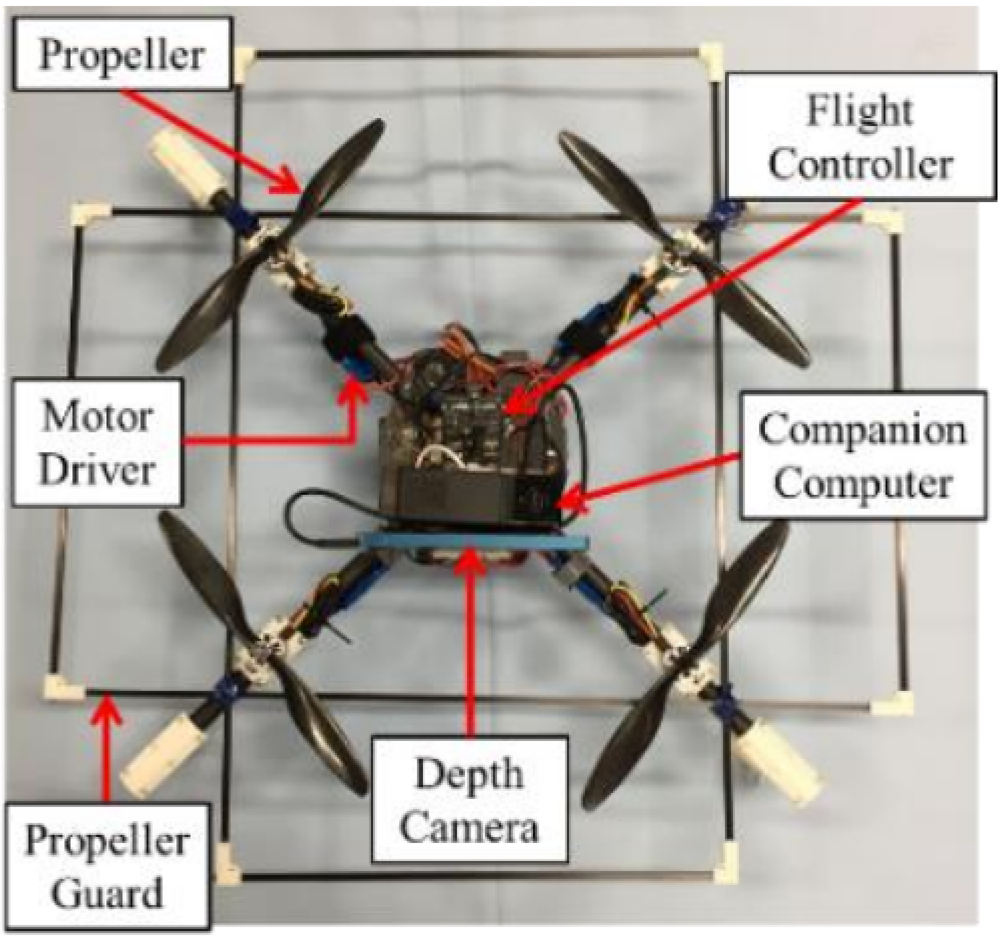


FIGURE 3.5: UAV system

Source: [15]

SLAM algorithm was utilized in this research. In most cases of inspection task, 3D map for autonomous navigation is not provided. So the drone is required to have capability of both making map near the target position and estimating its position. Therefore, SLAM technology is appropriate for this application. SLAM software library for RealSense ZR300 depth camera developed by Intel corporation was employed in this research for map building and localization. It updates position information 30 times per a second with sensor data of depth camera.

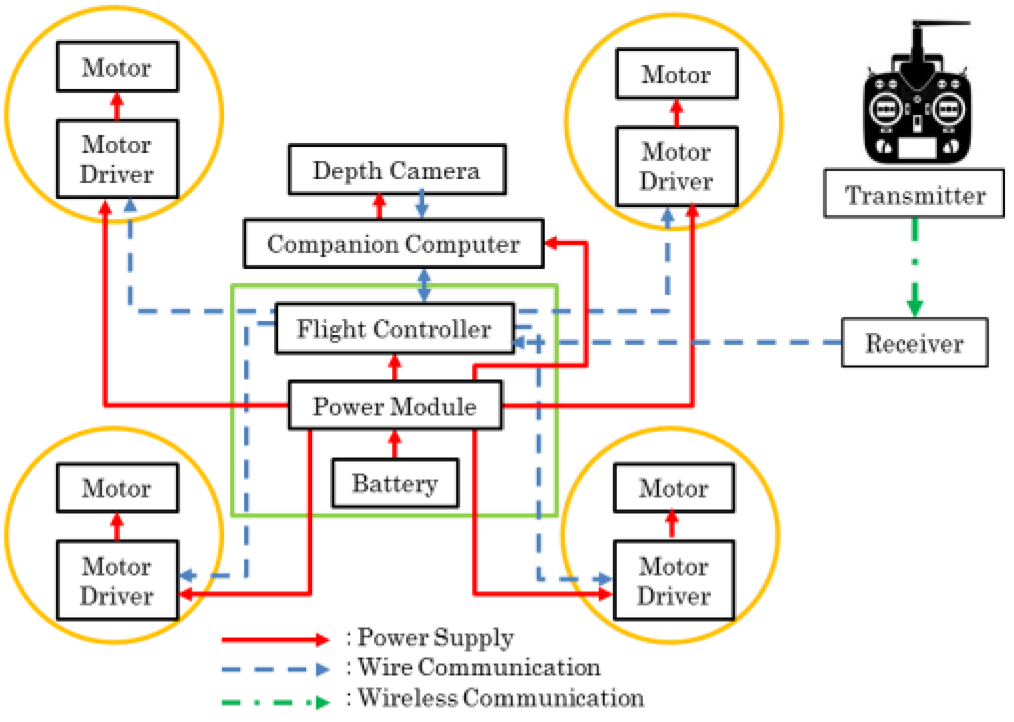


FIGURE 3.6: Control Hardware

Source: [15]

Experimental work to confirm the capabilities of map generation and localization was performed on a development system using a depth camera and a software library of SLAM algorithms. The system has started a linear move from the origin to the midpoint. After reaching the passing point in linear motion from the origin, return to the origin at in a circular orbit. The coordinates estimated by the method developed when the system was at the midpoint were $(x, y, z) = (0.02, -0.02, 1.08)$[m], but the actual coordinates were $(x, y, z). = (0, 0, 1)$ [m], resulting in an error of about 8 [cm] in the z direction. The coordinates of the experimental system after returning to the origin were $(x, y, z) = (-0.04, 0.00, -0.04)$[m]

3.2 Real time Openings Detection

3.2.1 LIDAR approach

C. Fernández-Caramés ,V. Moreno · B. Curto, J. F. Rodríguez-Aragón, F. J. Serrano, in their work [16] present a door detection system that combines data from an end-user camera and a laser rangefinder to enable robots to detect doors for more flexible and complex navigation. The approach uses Haar-like features and the Integral Image to reduce computation time, making it more efficient than other methods in the literature. However, the system requires an additional hardware module, a laser rangefinder, which adds to the overall cost and complexity of the system. In our approach, we aim to minimize the hardware components needed and explore alternative methods for door detection that can be implemented with existing sensors.

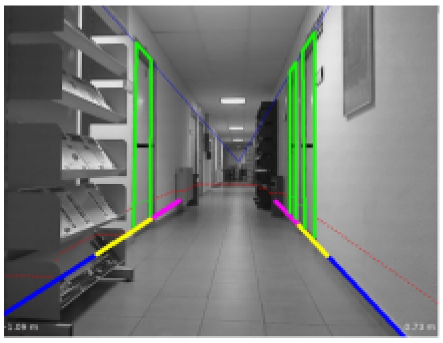


FIGURE 3.7: Lidar Openings detection

Source: [16]

3.2.2 CNN's Approach

Onder Alparslan and Omer Cetin, In their study, [17] considering the UAV's navigation inside the disaster chamber, doors, windows, and stairs are considered possible gates, and they should all be classified as open or closed. To provide this validation, an approach was developed that matches the camera view with other sensor data to ensure that the object pose exists. Additionally, the width of the detected object is calculated and controlled whether it is greater than the width of the UAV. Incidentally, safe traverse spaces are marked as possible gateways to another space or spaces and used for UAV trajectory planning.In this study, a method was developed to detect whether an object is open or closed by fusing lidar data provided by optical sensors with classified object data. For this purpose, LIDAR and optical cameras are used as sensors. CNN indoor object detection algorithm and fusion method developed as part of the research.

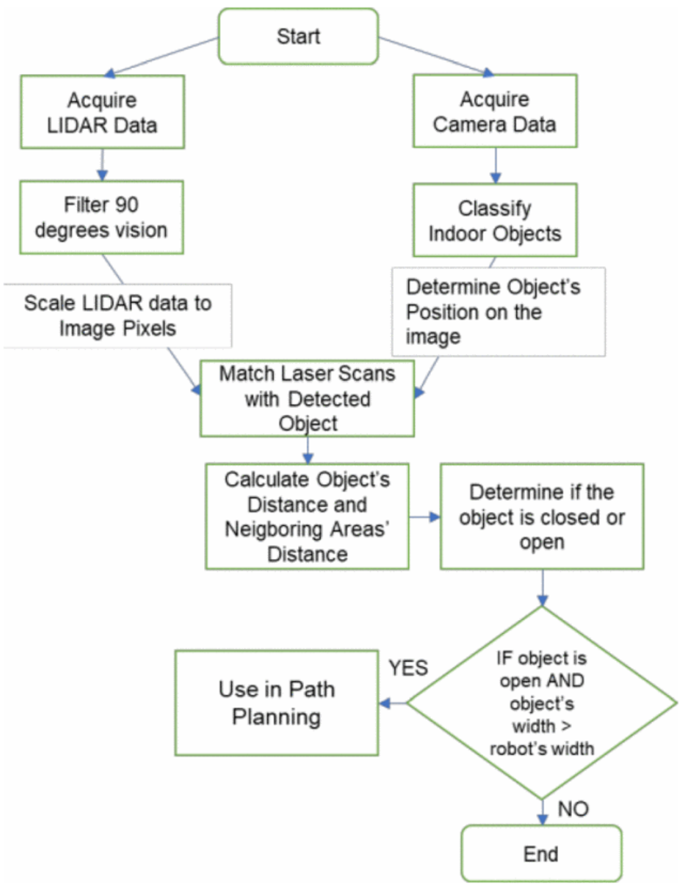

Start
Acquire LIDAR Data
Acquire Camera Data
Filter 90 degrees vision
Classify Indoor Objects
Scale LIDAR data to Image Pixels
Determine Object's Position on the image
Match Laser Scans with Detected Object
Calculate Object's Distance and Neigboring Areas' Distance
Determine if the object is closed or open
IF object is open AND object's width > robot's width
YES
Use in Path Planning
NO
End

FIGURE 3.8: System Modeling

Source: [17]

The main purpose of this research is to collate different kinds of data without delay from different sensors that can perceive different properties of objects and environments in the field of view. To this end, the fusion method should be applicable without knowledge of the physical properties of the sensor and independent of sensor architecture, manufacturer and model. Considering payload, one of the biggest challenges of small platforms, he should prioritize equipment that is as light as possible. Indoor unmanned aerial vehicles must also be miniaturized so that they can easily navigate confined spaces. To focus on this point, a regular optical camera and a 2D LIDAR device were chosen as sensors. The Logitech Brio camera offers a 90-degree close-up view in 4K. It produces at least 30 fps and weighs only 63 grams. A RP Lidar-A2 was used as the 2D lidar sensor. One of the most important features is the

0.25 ms sampling time compatible with cameras producing 30 frames per second. The goal is to demonstrate the efficiency of 2D LIDAR on a UAV that can move vertically in 3D space because it is lightweight and easy to implement.

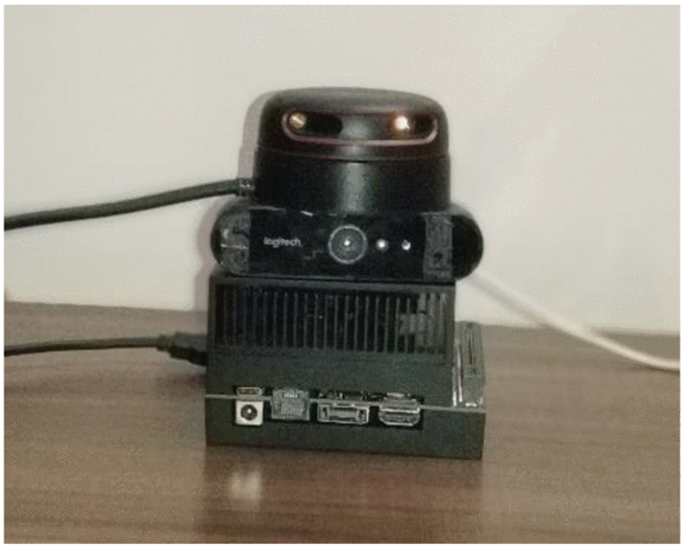


FIGURE 3.9: Hardware

Source: [17]

While hovering, the flying platform detects crossing sites in real-time and inquires as to whether they are open or closed using LIDAR. The returning data is incorporated into the object's labeling algorithm and used to determine whether it is open or closed.The key distinction between the door and window characteristics was their height from the ground, even though it was the goal of this study to identify the doors, windows, and transition zones inside the building. Glass surfaces do not interfere with the system's operation, and the proposed method can be simply modified to detect additional transitions. The technology was tested with doors and windows. In both situations, the crossing point is effectively classified. Additionally, the object's breadth and distance are measured. The information can be used for car navigation with positive results as a result.

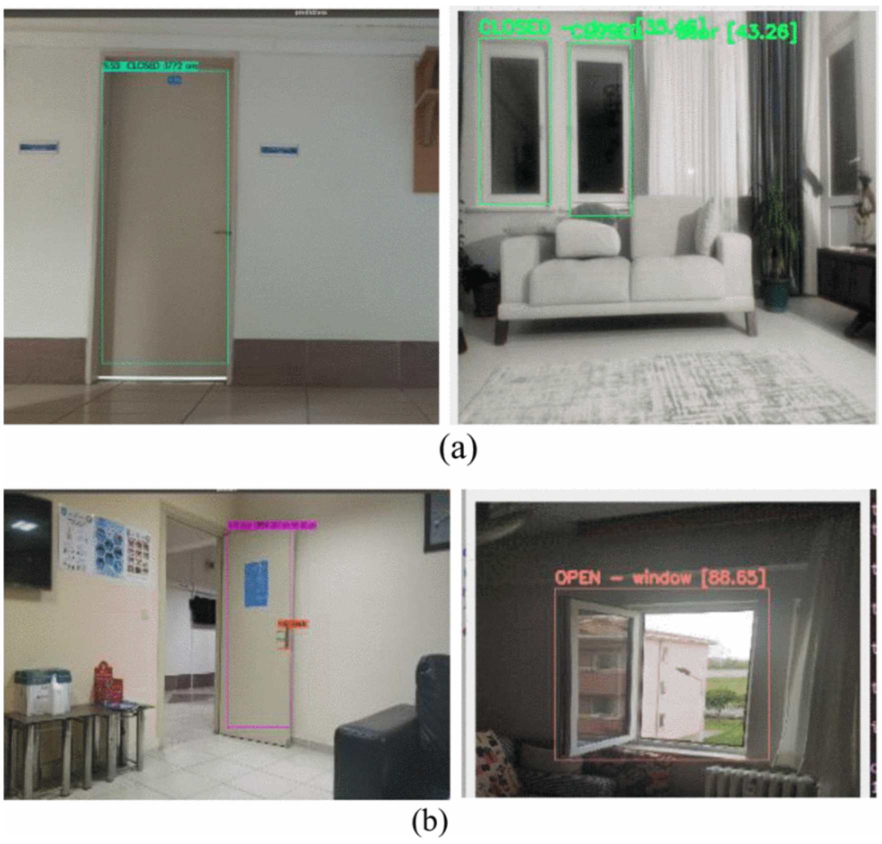


FIGURE 3.10: Results

Source: [17]

3.3 The Hardware Perspective

According to all the previously described ways of approaching each problem specifically we must carefully examine each way seperately to extract the way of this approach depending on the core needs the work, but also how to combine the technologies in order to integrate them together in the system to achieve the right results. The primary objective of the specific work is to achieve as much as accurate possible results, processing a sufficient number of frames per second consuming the less possible resources. Also a really important parameter that needs to taken into account is that the system being made for inspection it needs be economic in a power matter. This need leads us to the result that the system needs to have the minimum hardware modules as possible consuming the less possible energy.

In the above solutions we saw approaches using combinational layouts with Lidars, monocular and depth camera sensors and GPU's. For the current work it is greatly important to keep the power consumption to the least needed, thus the goal is to focus on aquiring all the data for localiyation. mapping and openings detection with one optical unit and make the whole processing using just a CPU.

3.4 Thesis Approach

As mentioned in the above section, this works primary goal is to achieve viable results by minimizing the energy workload and hardware resources. For that reason the solution for each sub problem must take into account the specifications for the other problem as for the hardware, to the computational procedures as well.

Due to the fact that the problem contains feature detection and extraction, using a camera provides the system with more alternatives and methods for achieving sufficient results. Furthermore, the depth extraction is essential for obtaining the space perception the system requires in order to achieve robust indoor navigation. This thought, leads us to the use of a stereo camera. With a stereo camera the system should be able to perform visual odometry extraction, depth extraction and feature detection for the openings detection sub problem. With this design, we can obtain all the information needed by skipping the use of a LIDAR module and by using only the stereo camera. However, the computational requirement for all these procedures could be large enough drop the systems performance significantly. For that reason, as far as the detection module is concerned, the proposed approach contains low cost Image Processing filters based on Canny Edge Detection where by fine tuning their parameters to match the camera attributes, it provides the system sufficient openings extraction.

Chapter 4

Theoretical Modeling

This chapter focuses on the methods and theoretical background used to address the problems in this work. The aim is to provide a thorough explanation of the approach and rationale for the decisions made. Mathematical models and algorithms utilized to solve the problems will be presented, and the steps taken to integrate these models will be described.

4.1 Localization Module

Localization is a very crucial problem regarding autonomous navigation. Various localization methods have been proposed regarding different environmental conditions. The most approach is Global Navigation Satellite System (GNSS) which refer to a constellation of satellites providing signals from space that transmit positioning and timing data to GNSS receivers. In this work the UAV need operates mostly on indoor environments, where there is no access to GNSS signals. The proposed architecture should be able to provide positioning data using sensor fusion.

IMU Data

The main source of information about attitude will be IMU data. IMUs can provide accurate three-dimensional estimates of angular velocity and linear acceleration in addition to steady data on the present attitude. The attitude information that an IMU provides is based on absolute readings that are impervious to cumulative inaccuracies. Finally, considering that the UAV can get to really narrow places, the IMU required for this study should be able to produce data with really high accuracy.

Stereo Camera

The camera can determine the distance to a conspicuous feature that is being utilized as a reference point by the algorithm by comparing the images from the two cameras and computing the differences between them. In particular, a larger distance to the feature correlates to a higher difference in the positions of the feature as seen by each of the two cameras. The accuracy of these estimations is then improved by observing them across consecutive camera frames and adding data from the IMU. Over time, a map of the locations of visual features develops. This is called Visual-inertial odometry. Relocalization refers to the camera's capacity to identify when it has returned to a familiar location by utilizing the features it has previously observed. The algorithm can re-localize, for instance, if a drone performs an aerial loop before returning to its initial location by comparing the visual surroundings to its database of significant elements and their locations in space. The camera can locate its point of origin by comparing its position to that static frame of reference.

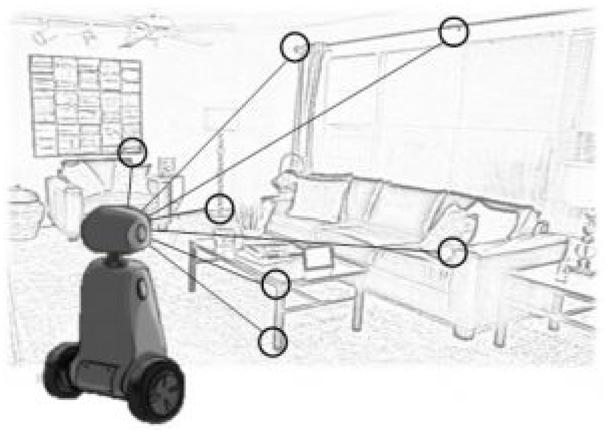


FIGURE 4.1: Visual-inertial re-localization, using visual points of reference to infer position

Source: Intel

The input data will be fused correctly in order to produce a final Localization vector that contains direction, pose and the estimated location given a certain initial point as shown in figure 4.2.

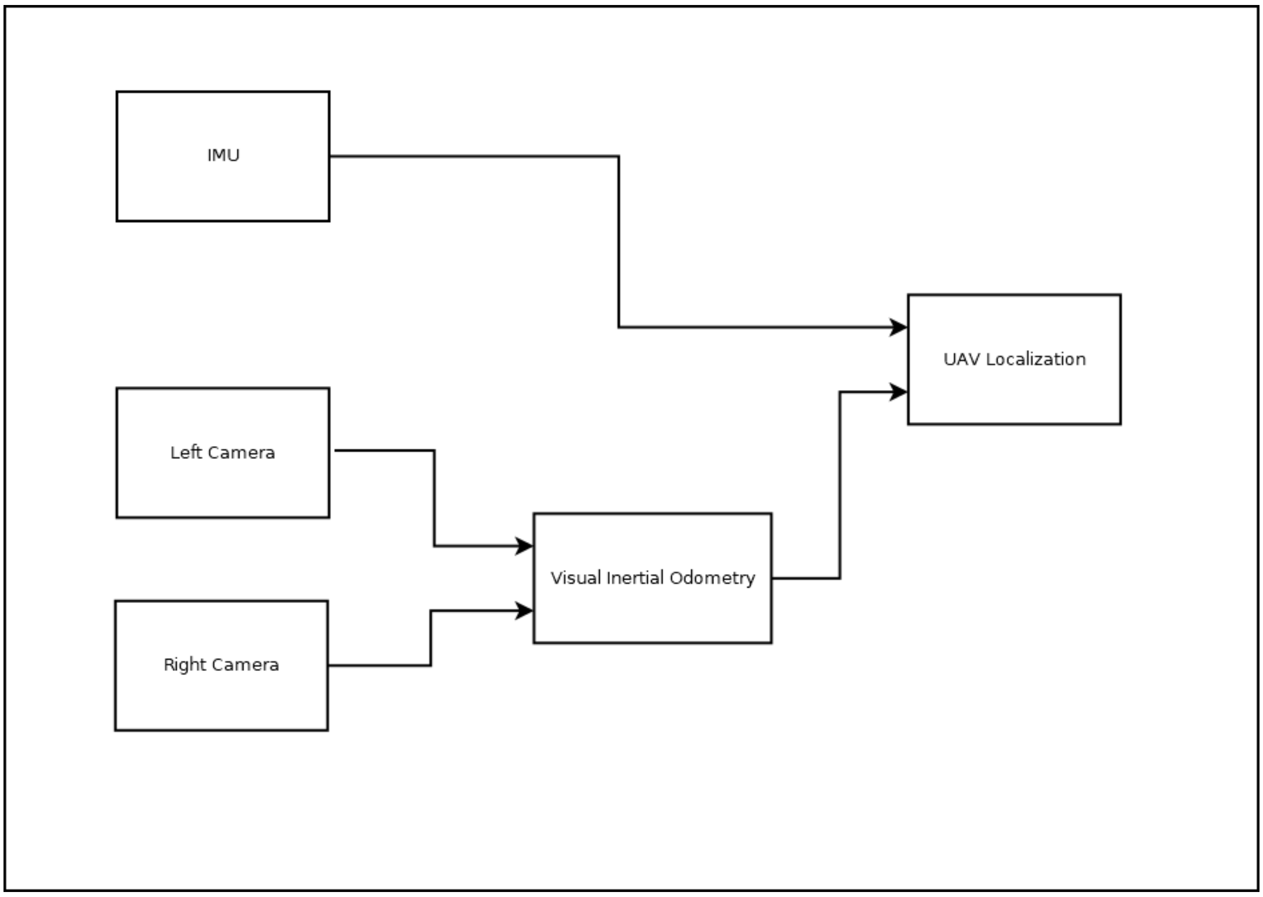


FIGURE 4.2: Localization Diagram

4.2 Mapping Module

Mapping is also a prior challenge in this work. The UAV must be able to perform robust path planning, avoiding obstacles and estimating correctly the attributes of any opening in order to finally navigate through it. A precise three dimensional representation of the space is a prerequisite for that problem. The mapping method this work proposes is the use of binocular vision and depth extraction techniques. The pipeline of the mapping procedure is described in these next steps.

4.2.1 Camera Callibration

A camera is a device that converts the 3D world into a 2D image. A camera plays a very important role in capturing three-dimensional images and storing them in two-dimensional images. To know the mathematics behind it is extremely fascinating. The following equation can represent the camera.

$$x = PX$$

Here x denotes 2-D image point, P denotes camera matrix and X denotes 3-D world point.

$$\underset{\substack{\text{homogeneous}\\ \text{image}\\ 3 \times 1}}{\begin{bmatrix} X \\ Y \\ Z \end{bmatrix}} = \underset{\substack{\text{Camera}\\ \text{matrix}\\ 3 \times 4}}{\begin{bmatrix} p_1 & p_2 & p_3 & p_4 \\ p_5 & p_6 & p_7 & p_8 \\ p_9 & p_{10} & p_{11} & p_{12} \end{bmatrix}} \underset{\substack{\text{homogeneous}\\ \text{world point}\\ 4 \times 1}}{\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}}$$

FIGURE 4.3: Vector representation of $x = PX$

Source: opencv.docs

Camera calibration is frequently used word in image processing or computer vision field. The camera calibration method is intended to identify the geometric characteristics of the image creation process. This is a vital step to perform in many computer vision applications, especially when metric information on the scene is needed. The camera is often categorized on the basis of a set of intrinsic parameters such as skew of the axis, focal length, and main point in these applications, and its orientation is expressed by extrinsic parameters such as rotation and translation. Linear or nonlinear algorithms are used to estimate intrinsic and extrinsic parameters utilizing known points in real-time and their projections in the picture plane

Camera calibration can be defined as the technique of estimating the characteristics of a camera. It means that we have all of the camera's information like parameters or coefficients which are needed to determine an accurate relationship between a 3D point in the real world and its corresponding 2D projection in the image acquired by that calibrated camera.

In most cases, this entails recovering two types of parameters.

Intristic Parameters

It allows mapping between pixel coordinates and camera coordinates in the image frame. E.g. optical center, focal length, and radial distortion coefficients of the lens.

Extrinsic Parameters

It describes the orientation and location of the camera. This refers to the rotation and translation of the camera with respect to some world coordinate system.

The camera matrix is unique to a specific camera, so once calculated, it can be reused on other images taken by the same camera. It is expressed as a 3x3 matrix.

$$camera\ matrix = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

FIGURE 4.4: 3x3 Camera Matrix

Source: opencv.docs

To find these parameters, we must provide some sample images of a well defined pattern (e.g. a chess board). We find some specific points of which we already know the relative positions (e.g. square corners in the chess board). We know the coordinates of these points in real world space and we know the coordinates in the image, so we can solve for the distortion coefficients. For better results, we need at least 10 test patterns

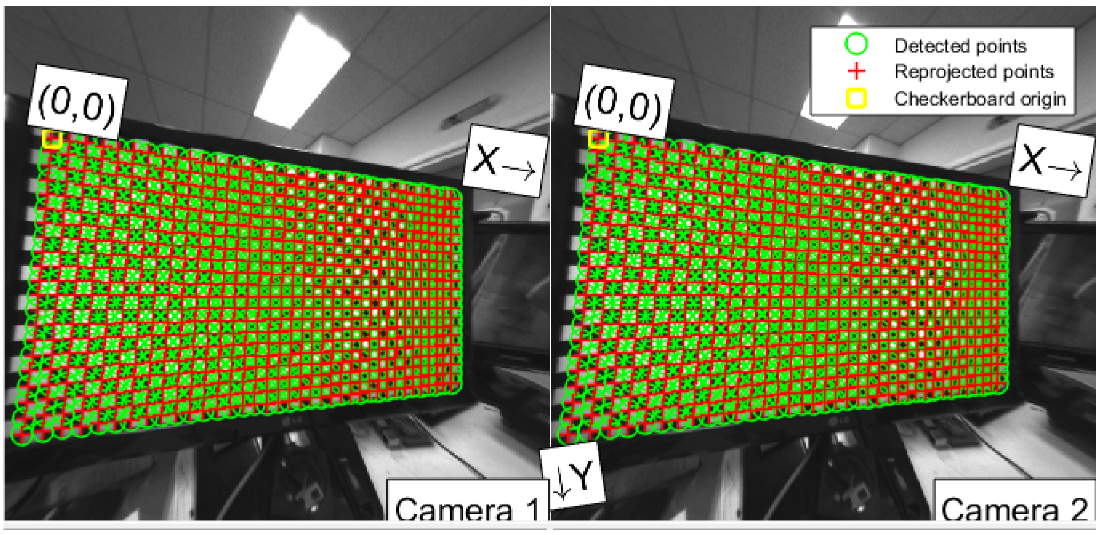


FIGURE 4.5: Chessboard

4.2.2 Image Undistortion

On this step, undistortion and rectification transformation maps are computed by using the camera's intristic and extrinstic parameters camera parameters. This new image is oriented differently in the coordinate space, according to rectification transformation in the object space (3x3 matrix). That, for example, helps to align two heads of a stereo camera so that the epipolar lines on both images become horizontal and have the same y- coordinate (in case of a horizontally aligned stereo camera).

Additionally, the maps needed by the inverse mapping algorithm are constructed. In more detail, the function calculates the matching coordinates in the source image for each pixel (u, v) in the destination (corrected and rectified) image (that is, in the original image from camera). The procedure used is as follows:

$$
\begin{array}{l}
x \leftarrow (u - c'_x)/f'_x \\
y \leftarrow (v - c'_y)/f'_y \\
[X\,Y\,W]^T \leftarrow R^{-1} * [x\,y\,1]^T \\
x' \leftarrow X/W \\
y' \leftarrow Y/W \\
r^2 \leftarrow x'^2 + y'^2 \\
x'' \leftarrow x' \frac{1+k_1 r^2+k_2 r^4+k_3 r^6}{1+k_4 r^2+k_5 r^4+k_6 r^6} + 2p_1 x' y' + p_2(r^2 + 2x'^2) + s_1 r^2 + s_2 r^4 \\
y'' \leftarrow y' \frac{1+k_1 r^2+k_2 r^4+k_3 r^6}{1+k_4 r^2+k_5 r^4+k_6 r^6} + p_1(r^2 + 2y'^2) + 2p_2 x' y' + s_3 r^2 + s_4 r^4 \\
s \begin{bmatrix} x''' \\ y''' \\ 1 \end{bmatrix} = \begin{bmatrix} R_{33}(\tau_x, \tau_y) & 0 & -R_{13}((\tau_x, \tau_y) \\ 0 & R_{33}(\tau_x, \tau_y) & -R_{23}(\tau_x, \tau_y) \\ 0 & 0 & 1 \end{bmatrix} R(\tau_x, \tau_y) \begin{bmatrix} x'' \\ y'' \\ 1 \end{bmatrix} \\
map_x(u, v) \leftarrow x''' f_x + c_x \\
map_y(u, v) \leftarrow y''' f_y + c_y
\end{array}
$$

where $(k_1, k_2, p_1, p_2[, k_3[, k_4, k_5, k_6[, s_1, s_2, s_3, s_4[, \tau_x, \tau_y]]]])$ are the distortion coefficients.

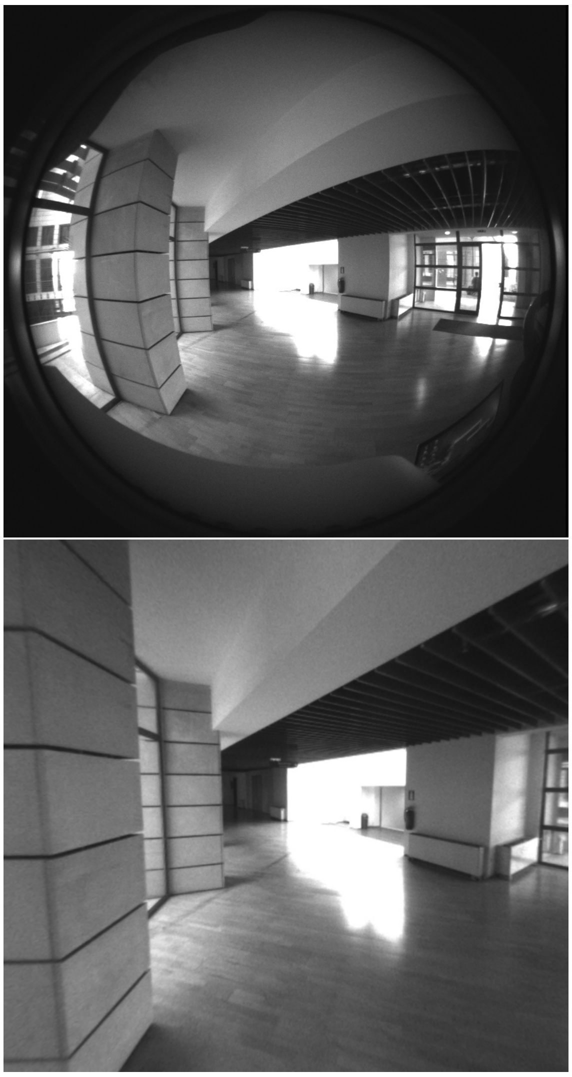


FIGURE 4.6: Image Undistortion

4.2.3 Disparity and filtering

The next step in the Mapping Pipeline is the disparity extraction of the given left and right images. The proposed method for this stage is using Semi-Global Block Matching where the pair of rectified left and right images are passing into the SGBM blocks producing the first version of the disparity map.Stereo matching algorithms, that are intended for real-time processing on CPU, tend to make quite a few errors on challenging sequences. These errors are usually concentrated in uniform texture-less areas, half-occlusions and regions near depth discontinuities. One way of dealing with stereo-matching errors is to use various techniques of detecting potentially inaccurate disparity values and invalidate them, therefore making the disparity map semi-sparse.For the reason that navigation requires extremely high accuracy in the depth-map, this study proposes the use a of filtering procedure to align the disparity map edges with those of the source image and to propagate the disparity values from high- to low-confidence regions like half-occlusions. This procedures pipeline is presented in the next steps.

- We start by loading the source recitified stereo-pair.
- Afterwards we perform downscaling of the views to speed-up the matching stage at the cost of minor quality degradation.
- We are using StereoSGBM to provide best possible quality. The filter instance is created by providing the StereoMatcher instance that we intend to use. Another matcher instance is created for the right Image. These two matcher instances are then used to compute disparity maps both for the left and right views, that are required by the filter
- Disparity maps computed by the respective matcher instances, as well as the source left view are passed to the filter. Note that we are using the original non-downscaled view to guide the filtering process. The disparity map is automatically upscaled in an edge-aware fashion to match the original view resolution.

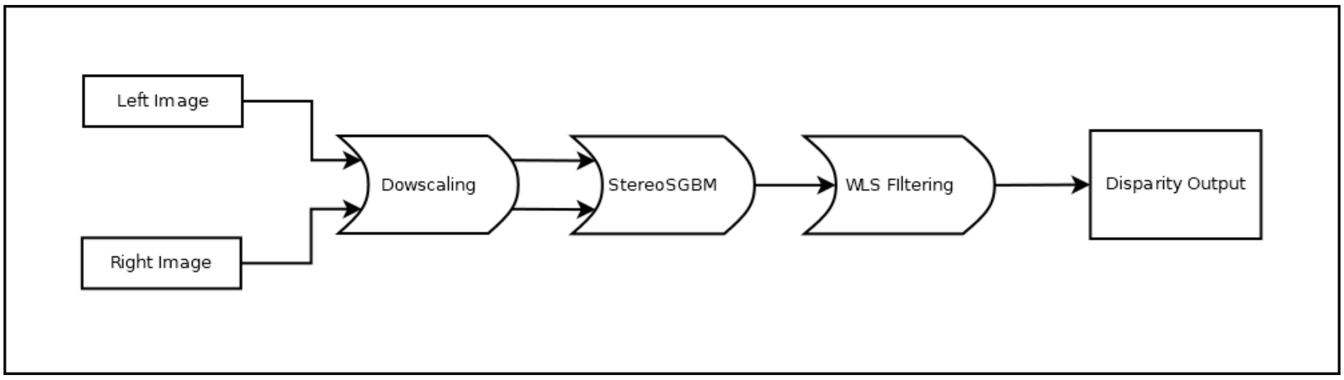


FIGURE 4.7: Disparity extraction pipeline using stereo SGBM matching and WLS post filtering

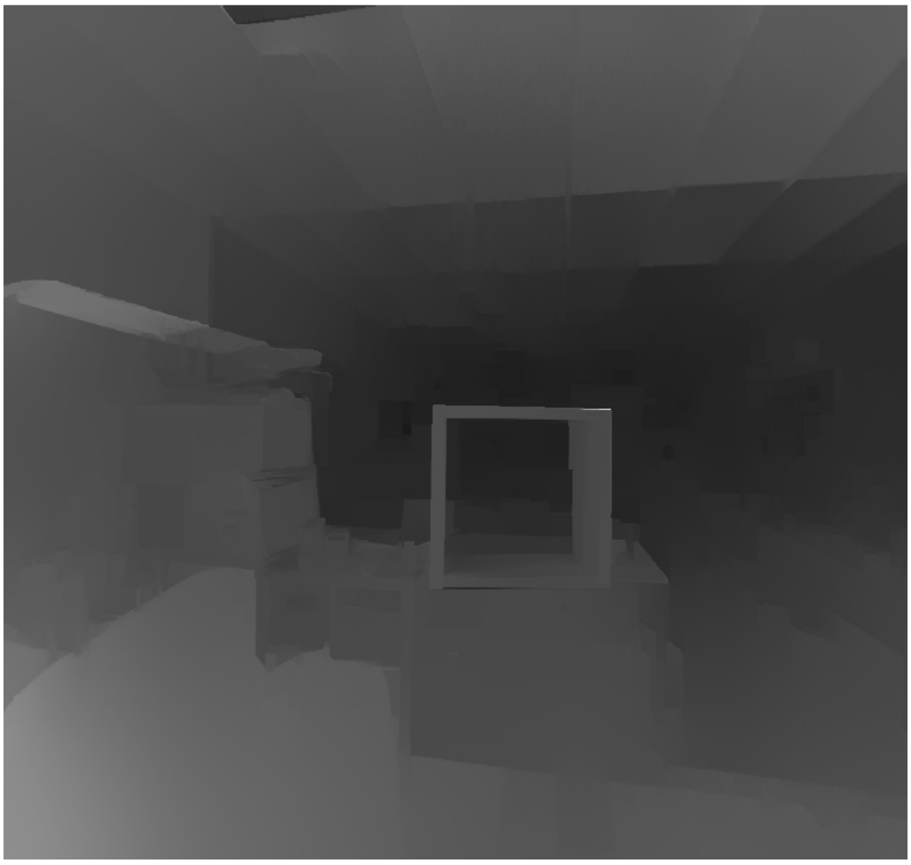


FIGURE 4.8: Filtered Disparity Output

4.2.4 Pointcloud Generation

Depth is inversely proportional to disparity. The more the disparity is, the closer the object is to the baseline of the camera. The less the disparity is, the farther the object is to the baseline. In binocular stereo where the two camera axes are parallel, depth can easily be calculated given the disparity (the shift in position for corresponding points between the images). If the focal length of both cameras is f, the baseline b and disparity d, then the depth z is given by

$$z = f * b/d$$

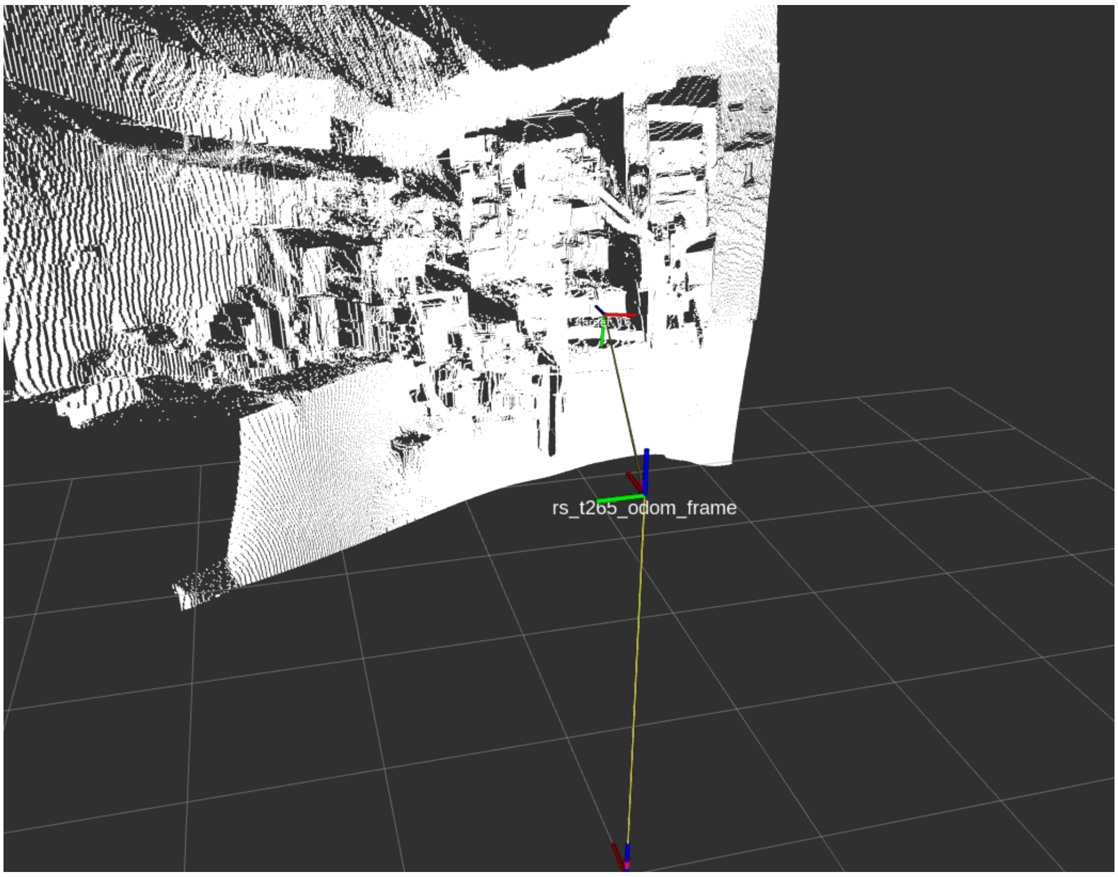


FIGURE 4.9: 3D-Pointcloud Extracted from the disparity output

4.3 Openings Detection Module

The **openings detection** module is the feature of this work taking it a step further to achieve more robust navigation and better structural perception making the system able to recognize more features regarding the space which has been deployed. The real-time factor which is a core requirement for the specific work enables some standard requirements for the specific feature. Furthermore, the fact that the system is hosted in an embedded processor with restricted memory and processing power conclude that the following requirements must be met in order to have successful tests. The requirements are the following:

- The module must be able to autonomously detect an opening.
- The module must be able to process and analyze several camera frames per second in order to constantly have information about the surounding openings.

- Due to the restricted memory and processing power, the module must be implemented in respectful way to the available resources.

Related works such as (..related works..) integrate convolutional neural networks in order to clasify potential openings with really remarkable results.

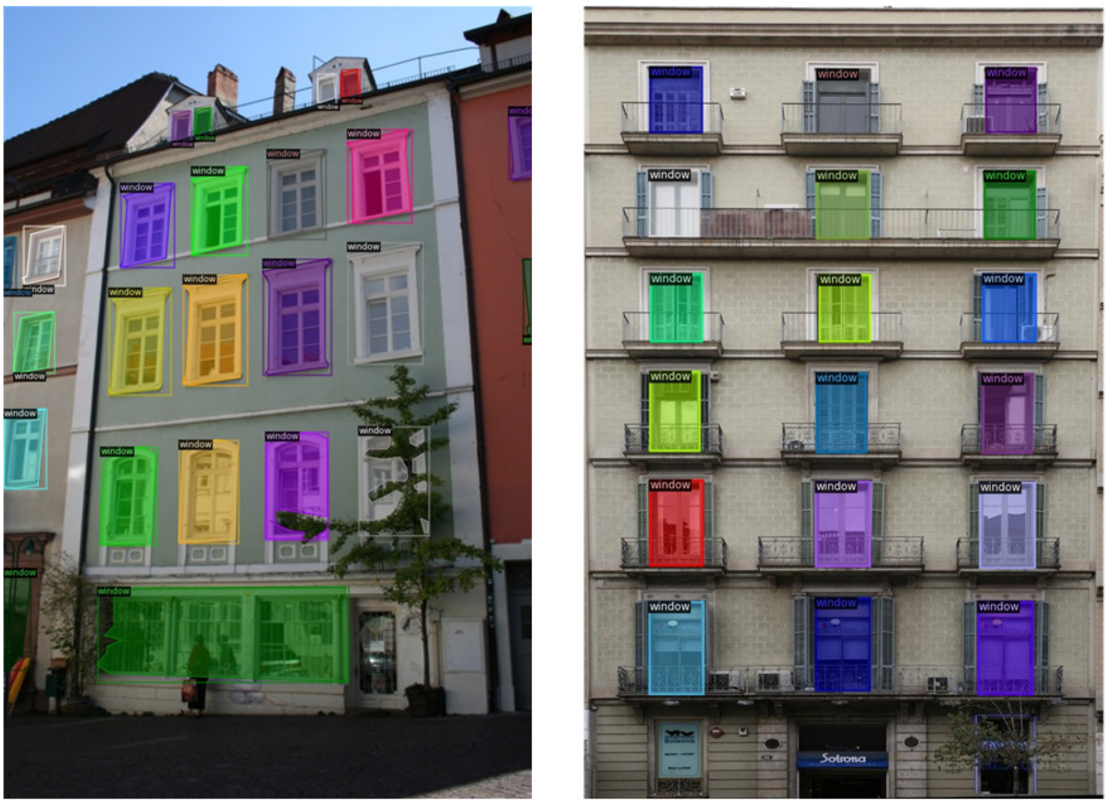


FIGURE 4.10: Window detection using CNN

Source: [17]

However, the restrictions that mentioned earlier, require to explore more "economic" ways to achieve sufficient results. The method that the specific work proposes is using a specific pipeline that consists of passing the left camera frame through a Canny Edge Detector in order to obtain the edges. The reason that we are using the left camera frame as input is because the extracted disparitys pixels from the SGBM are aligned directly to the left frame.The Canny Edge Detectors output enters a Hough Line Transform module in order to extract lines from the detected edges.Finally the lines enter a custom module that is responsible for extracting rectangulars from the detected lines. Both **Canny Edge Detectors** and **Hough Line Transforms** thresholds must be utilized with respect to specific attributes of the operating camera. The proposed pipeline is presented to the following figure.

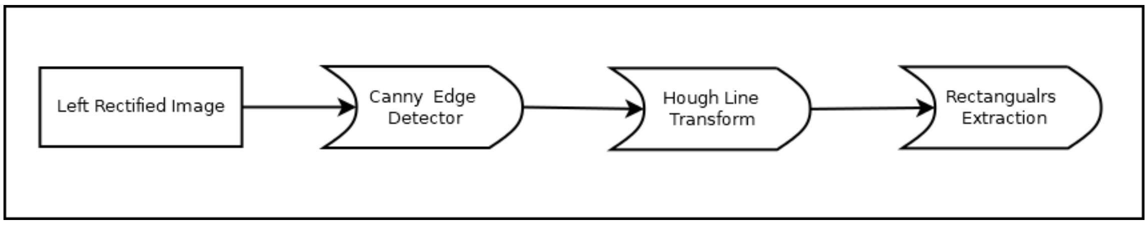


FIGURE 4.11: Openings Detection Diagram

The **Extract Rectangulars** module is responsible for extracting rectangulars from the detected lines. This is achieved by classifying the lines to horizontal and vertical, finding the intersecting points of these lines with respect to each lines angle to the x-axis and finally forming rectangulars from these intersection points.The number of the extracted shapes is relevant to the computational power is available in the current hardware.The most basic version is forming a rectangular from two consecutive points of a raw and a columnn.

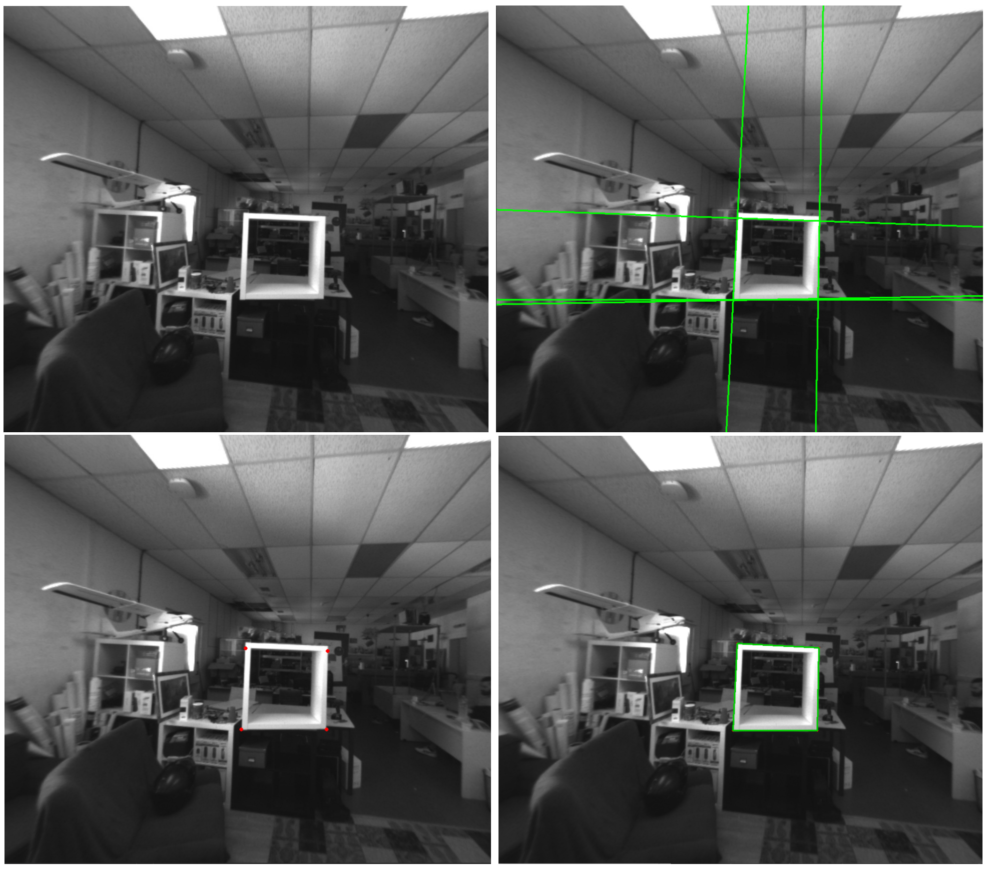


FIGURE 4.12: Openings Detection

Chapter 5

System Level Design

This chapter is dedicated to the design and development of our system, where we extensively analyze the hardware and tools utilized to build the system. We aim to provide a thorough explanation of the various hardware and software components used in our final solution and illustrate how we integrated them to develop a functional system. A detailed description of each hardware and software component, including its specifications, features, and design trade-offs, will be presented. Furthermore, we will discuss the integration process, including the obstacles we encountered and how we tackled them to ensure the system's successful development.

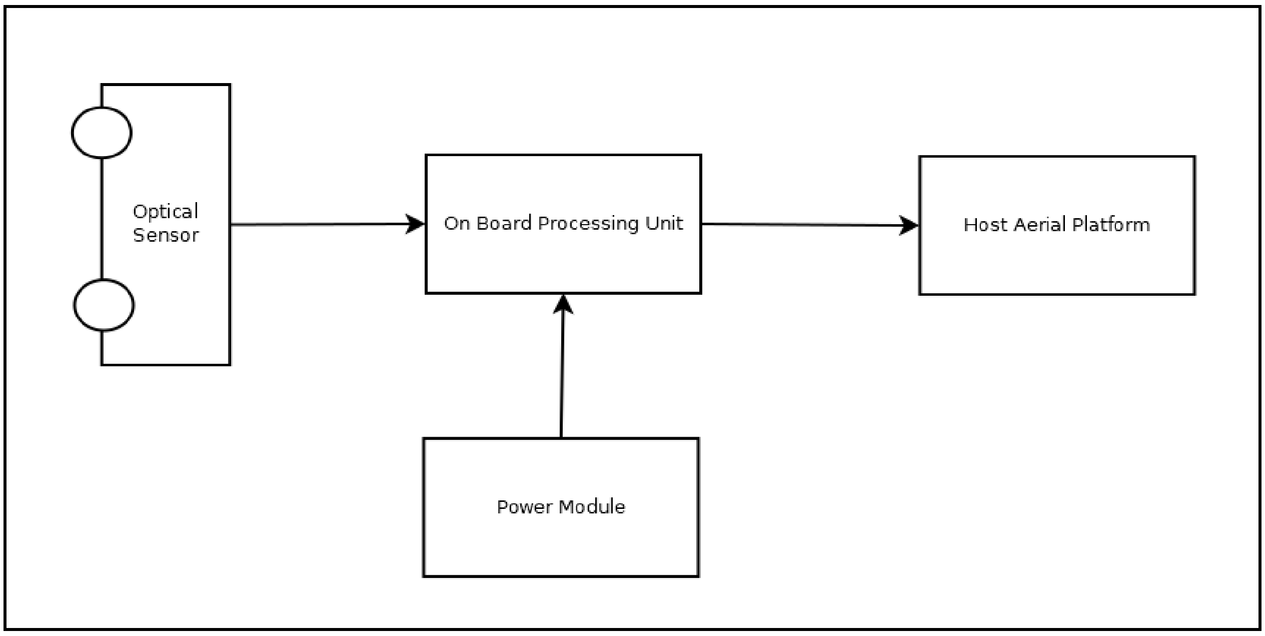


FIGURE 5.1: Top Level of the Hardware Components Layout

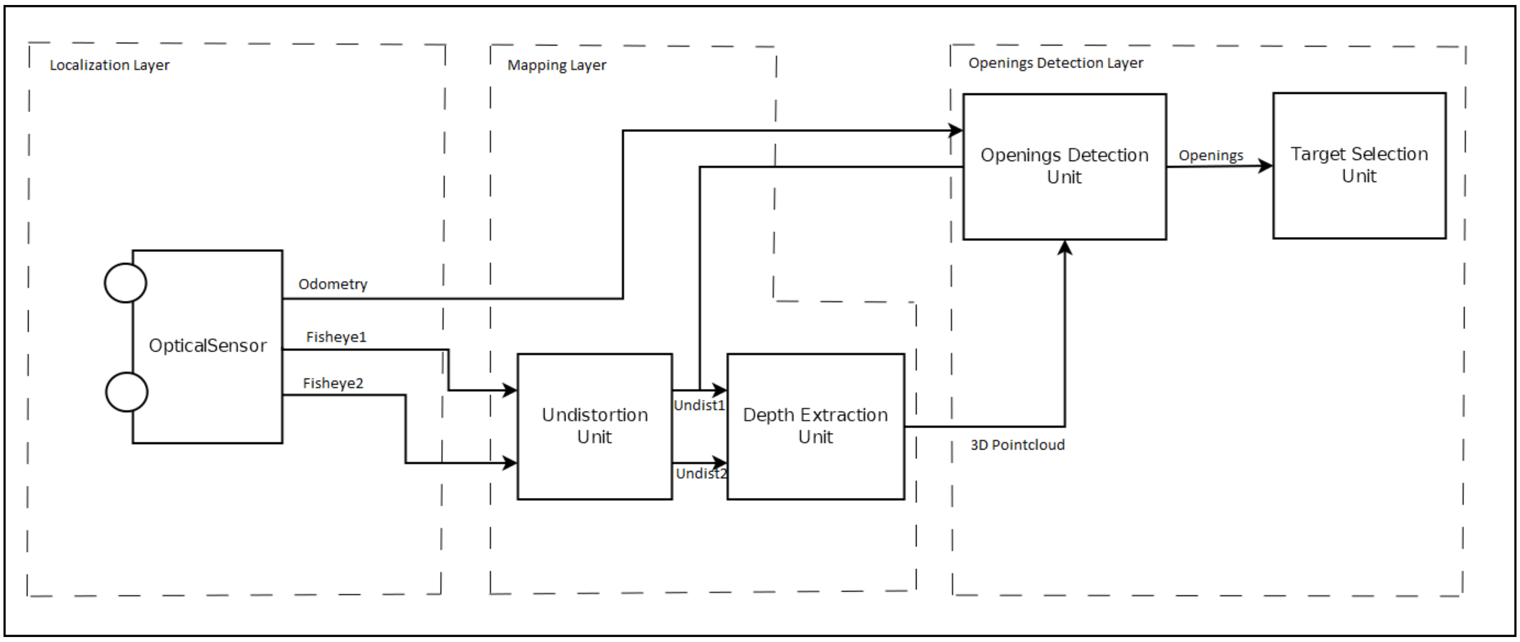


FIGURE 5.2: Top Level of the Software Components Layout

5.1 Overview

The design of the specific system was developed based on certain attributes that was set as principles of the whole idea as discussed in the previous chapter. First of all due to the fact that the system s designed to operate indoors it had to be relatively small in order to move in a flexible way even in narrow spaces. Secondly due to the fact that it is meant for space exploration, it is necessary to consume as less energy as possible and the battery needs to last as much as possible for not having to pause the exploration process halfway through the space. With those two constraints we tried to make the design as efficient as possible with the use of specific hardware modules and software tools.

In the next sections, we are going to present and analyze, the hardware used in the specific design including the Drones frame, motors, battery, flight controller, camera, CPU and also all the software tools and frameworks that were used in order to develop the logic of the system.

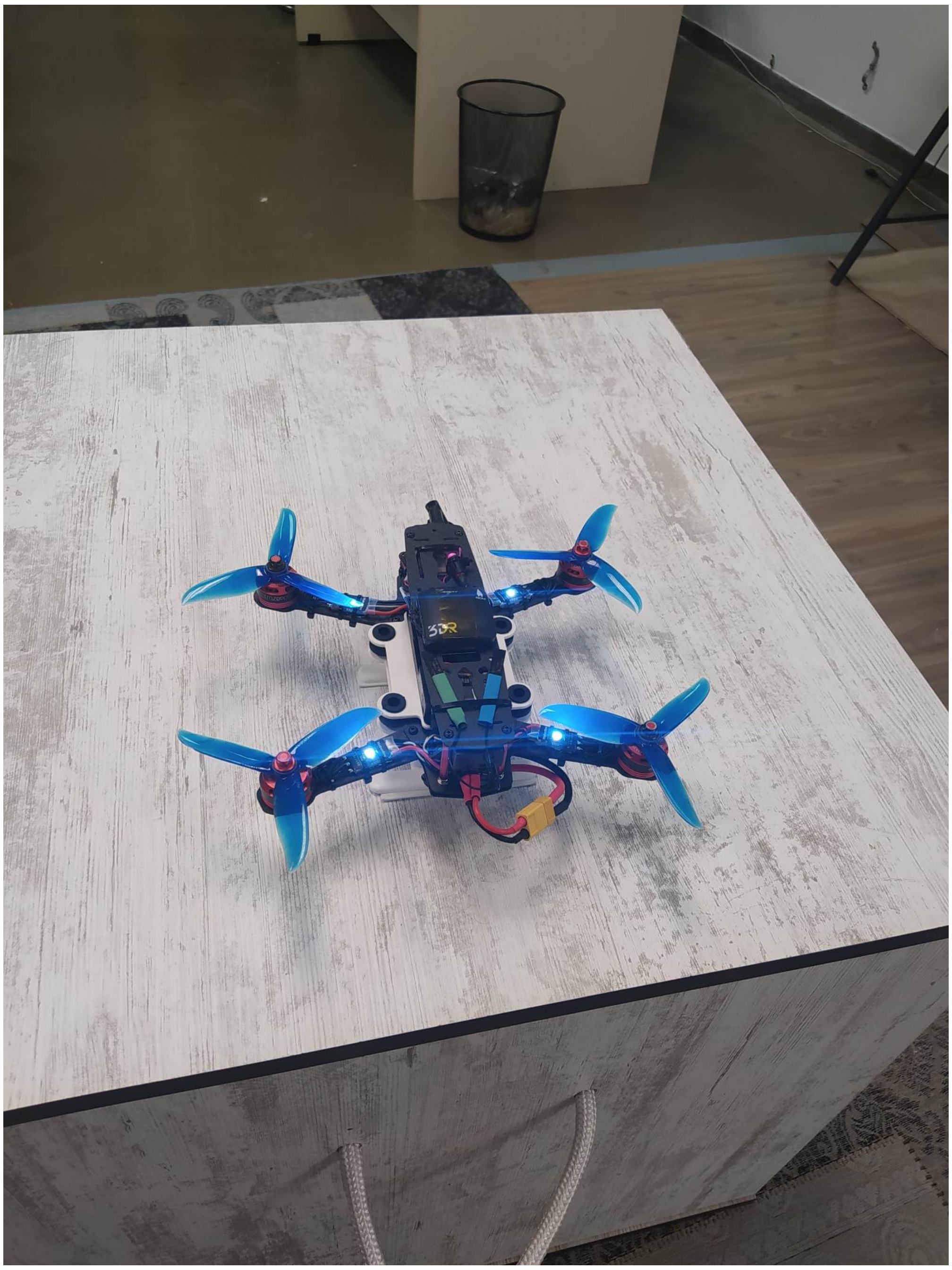


FIGURE 5.3: Primal Version of thesis UAV

5.2 Host Aerial Platform

On this section we are going to analyze the "Host Aerial Platform" component of the Top Level 5.1. The selection process for the host aerial platform of the system was made by taking into account all the hard requirements in order to find the most suitable drone type to integrate in the current work.

First of all, the fact that the drone should be as small as possible in order to be able to maneuver through narrow spaces is an important aspect that leaded us to our choice. Also it should be as less as energy consuming possible, so it's own weight should be minimized as well. Of course the host platform should also offer flight stability, precise control as well as the ability to maintain its position in order for the navigation system to provide the best possible results.

The category that was selected after studying all the possible designs is a multicopter rotorcraft with four number of rotors which is stable and precise enough for the purposes of our work, it's frames come in sufficiently small sizes fulfilling that requirement as well, and also due to the fact that there are four rotors mounted instead of six or eight, it can be less energy consuming than the other types.

FIGURE 5.4: Quadcopter Rotorcraft

Source: DJI Mini 2

5.2.1 Quadcopter's Frame

The frame used is a QAV250 FPV Race Drone frame made of carbon fiber. Carbon fiber is a material composed of a long chain of carbon atoms. The fibers are extremely strong and light. The joining of fibers creates composite components. Due to the nature of the fibers, the strength-to-weight ratio of a carbon fiber component is much higher than that of steel.

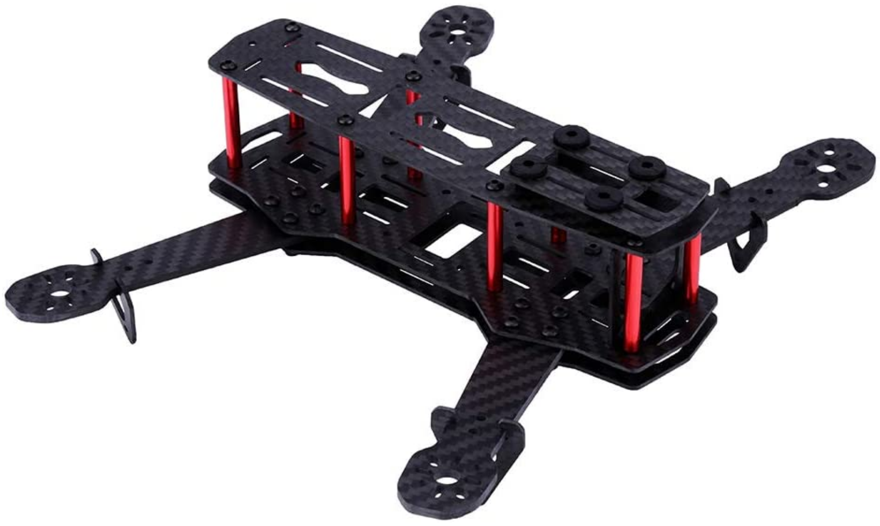


FIGURE 5.5: QAV250 FPV Race Drone frame

Source: Amazon

5.2.2 Propulsion

The Propulsion modules are composed of four 5045, 5-inch, Tri Blade propellers coupled with four 2300KV Brushless DC Motors. The propellers are split into two groups. Two are clockwise and two counter clock wise.This design eliminates gyroscopic effects when there is equal thrust distribution. Each motor weights 28.8g and their total thrust is 2.5 kg, making it more than eligible to support the whole system and capable for future upgrades.

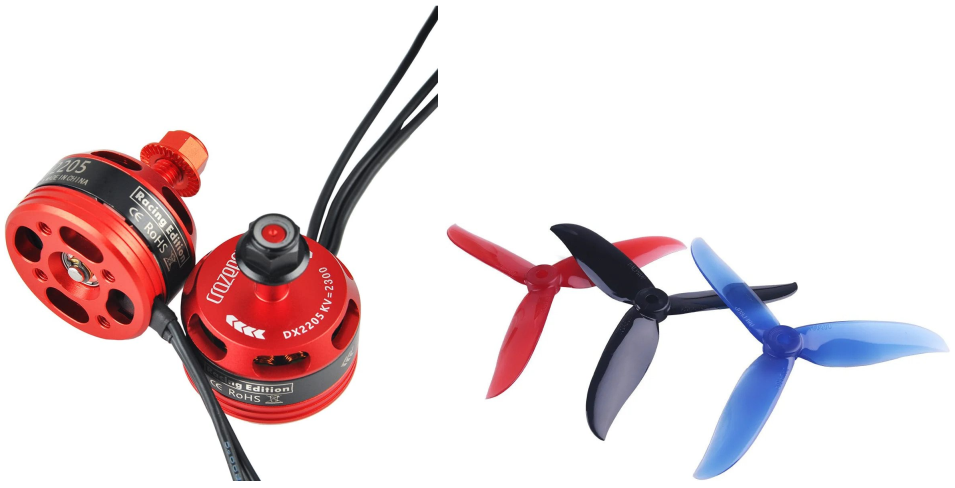


FIGURE 5.6: Propulsion modules

Source: Amazon

5.2.3 Electronic Speed Controllers

Driving a three-phase brushless DC motor is a challenging task as large sums of power and continuous monitoring of the electromagnetic fields generated in the motor. The units responsible for the precise control of the motors are the electronic speed controllers (ESC) of the copter. These units are controlled by the flight control unit (FCU) and float the motors by continuously adjusting the switching interval of each motor phase. As are the engines synchronously, the units must accurately measure the back electromotive force between motor phases to estimate rotor orientation and adjust commutation timing. This aircraft has four Makerstack ESCs that support 35A continuous current treatment at 22.2 volts.

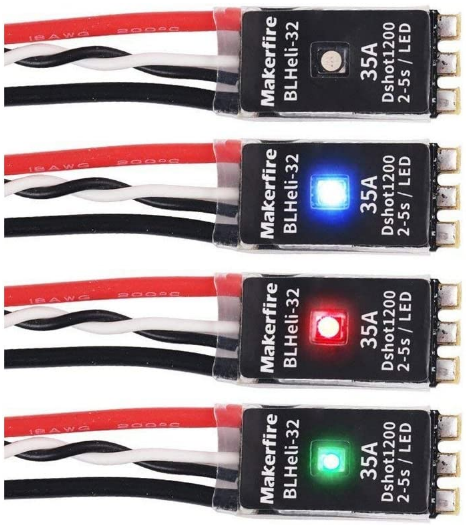


FIGURE 5.7: Electronic Speed Controllers

Source: Amazon

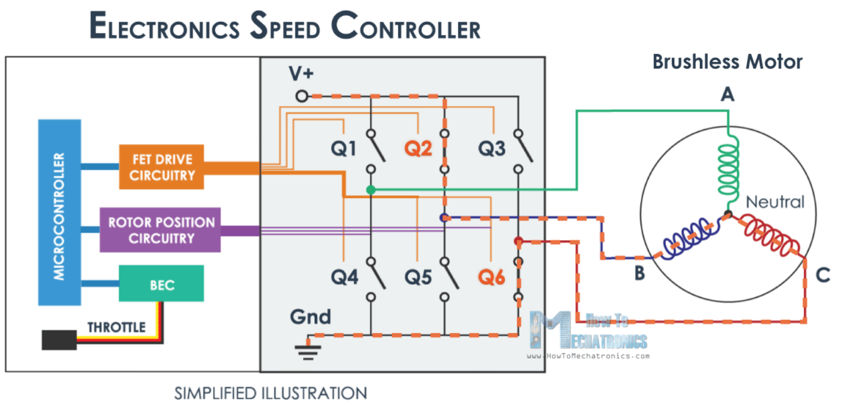


FIGURE 5.8: Electronic Speed Controllers

Source: How to Mechatronics

5.2.4 Battery

A OEM 4S1P 30C 2500mAh 14.8V battery is integrated into the system as a power source. The "4S" in the name refers to the number of cells in the battery, which are connected in series and provide a nominal voltage of 14.8V. The "30C" rating refers to the maximum continuous discharge rate of the battery, or how quickly it can be discharged without damaging the cells. The "2500mAh" rating refers to the capacity of the battery, or how much energy it can store.

Integrating this battery into the drone is important for energy management because it determines how long the drone can fly and how much payload it can carry. It is also important for limiting the weight of the drone, as a larger, heavier battery would add weight and potentially decrease the drone's performance. By carefully selecting a battery with the appropriate capacity and discharge rate, we optimized the energy usage and weight of the drone to meet the specific needs of the specific application.

FIGURE 5.9: Power Source

Source: https://www.gensace.de/

5.2.5 Flight Controller Unit

A flight controller is a device that is used to stabilize and control the flight of a drone or other unmanned aerial vehicle (UAV). It is typically a small computer that is mounted on the UAV and is responsible for receiving input from various sensors and peripherals, such as gyroscopes, accelerometers, and GPS modules, and using this information to calculate the appropriate control signals for the motors and other actuators. The flight controller is the brains of the drone, and it plays a crucial role in ensuring stable and safe flight. For the specific study the chosen flight controller is the Pixhawk 4.

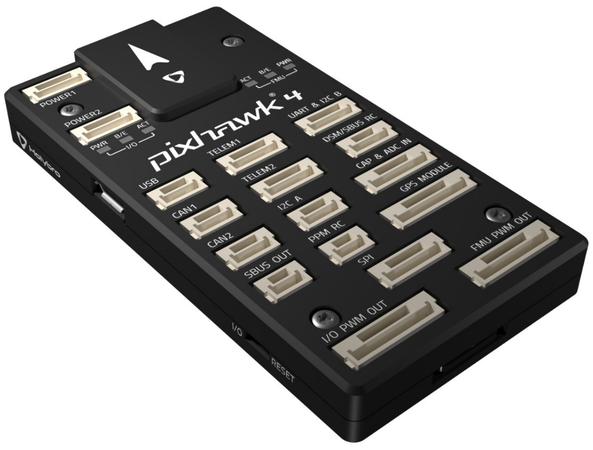


FIGURE 5.10: Pixhawk 4

Source: https://docs.px4.io/

The Pixhawk 4 is a popular flight controller used in unmanned aerial vehicles (UAVs) for a variety of applications, including mapping, inspection, and search and rescue. In this section, we will explore the hardware and features of the Pixhawk 4 and discuss how it can be integrated into a UAV.

The Pixhawk 4 is built on the STM32F765 microcontroller, which has a powerful Cortex-M7 processor running at 216 MHz. It also has 1 MB of flash memory and 512 MB of RAM, providing plenty of storage and processing power for flight control algorithms and data logging. The Pixhawk 4 also has a variety of sensors and peripherals, including a gyroscope, accelerometer, magnetometer, barometer, and GPS module. These sensors are used to measure the orientation, position, and velocity of the UAV and provide feedback to the flight control system.

Firmware

The Pixhawk 4 runs the PX4 flight stack, which is an open-source firmware developed by the PX4 project. The PX4 flight stack is designed for flexible and reliable control of UAVs and other autonomous systems, and it supports a wide range of sensors and peripherals. The PX4 flight stack includes a variety of flight modes, such as manual, stabilize, altitude hold, and waypoint navigation, which can be accessed through the ground station software or a radio control transmitter.

Compatibility

The Pixhawk 4 is compatible with a variety of ground station software, including QGroundControl, Mission Planner, and MAVLink Inspector. These programs allow the user to configure and tune the flight control parameters, upload waypoints, and view real-time telemetry data. The Pixhawk 4 is also compatible with a range of radio control transmitters, including the FrSKY Taranis and the Spektrum DX9, which can be used to override the flight control system and manually control the UAV.

Integration with Robot Operating System (ROS)

The Pixhawk 4 can be integrated with the Robot Operating System (ROS) using the mavros package. ROS is a powerful platform for building autonomous systems, and it provides a variety of tools for perception, localization, and control. The mavros package is a ROS wrapper for the MAVLink communication protocol, which is used by the Pixhawk 4 to communicate with ground station software and other autonomous systems. By integrating the Pixhawk 4 with ROS, it is possible to use a variety of ROS packages and libraries to build advanced autonomous applications for the UAV. In our case it gives the ability to our autonomous system to communicate with the flight controller and to secure back and forth communication with our autonomous system.

RC Overriding through mavROS

The Pixhawk 4 can be overridden using the RC radio control transmitter through the mavros package in ROS. This can be useful in situations where the flight control system is not behaving as expected or if manual control is needed for certain tasks. In the case of the Pixhawk 4 flight controller, RC overriding can be achieved through the mavros package in the Robot Operating System (ROS). The mavros package is a ROS wrapper for the MAVLink communication protocol, which is used by the Pixhawk 4 to communicate with ground station software and other autonomous systems. By using the mavros package, it is possible to control the UAV using code, rather than manually using the radio control transmitter.

5.2.6 Overview

The diagram presents an overview of the "host aerial platform" architecture in our Top Level diagram 5.1, divided into three main sections: the Energy section, the Flight Controller, and the Propulsion section. These three layers work together to provide stable flights for our system. The diagram gives a clear and simple representation of the system's components and their interactions, providing a good starting point for understanding the system's architecture.

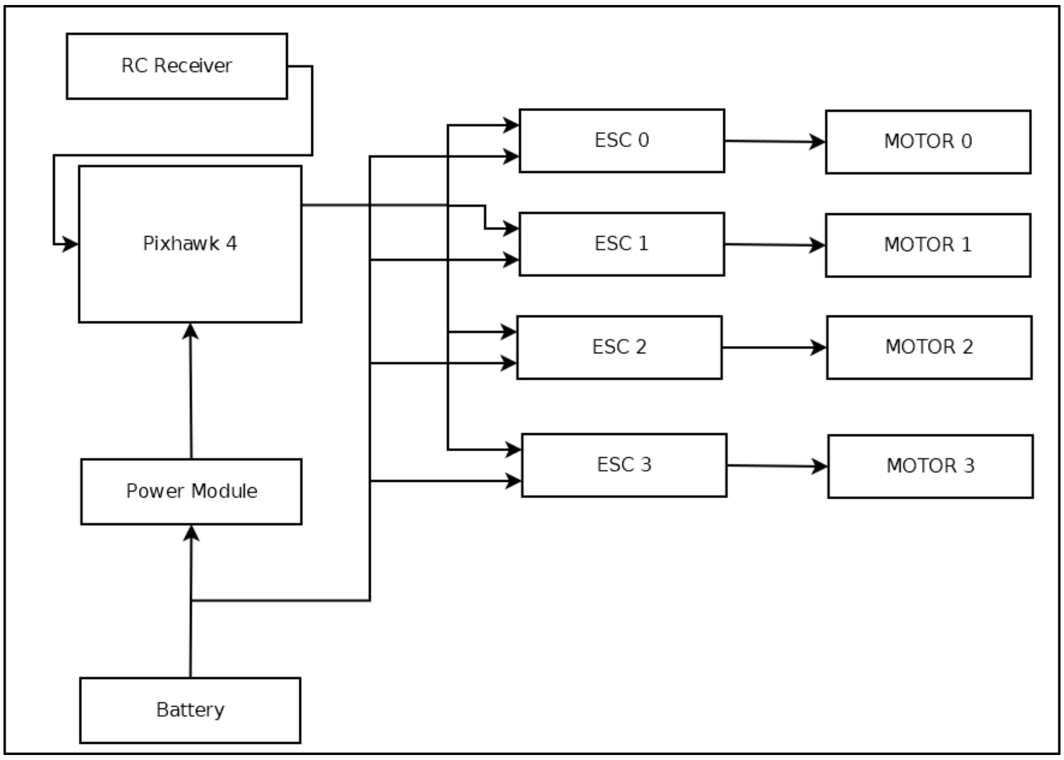


FIGURE 5.11: Host Aerial Platform Overview

5.3 Optical Sensor

This part has to do with the "Optical Sensor" component of the Top Level 5.1 which is a crucial component of our project as it is responsible for gathering vital data necessary to achieve this works goals. Among the various options available, we have chosen to utilize the Intel RealSense T265 for its advanced capabilities and high level of accuracy.The Intel RealSense T265 is an essential component of our project as it provides us with the ability to extract localization data, while maintaining the advantages of an optical sensor.As mentioned in the previous chapters, this work required a sensor that could provide precise real-time location data, as well as the ability to extract information about the environment which is to be deployed. The T265 is a tracking camera that uses proprietary visual inertial odometry (VIO) technology to achieve this, making it an ideal choice for our work. Unlike Lidar sensors, T265 provides the benefits of an optical sensor such as capturing visual information in addition to motion data which make it ideal for fulfilling all the goals of the specific application. In this section, we will delve deeper into the features and capabilities of the T265 and how it contributes to the overall success of our project by providing the best of both localization and mapping without sacrificing any advantages of an optical sensor.

FIGURE 5.12: Intel RealSense T265

Source: intelrealsense.com

The Intel RealSense T265 is equipped with a range of advanced hardware modules that allow it to perform its visual inertial odometry (VIO) technology. The device includes two fisheye cameras that capture stereo images, a 6-axis inertial measurement unit (IMU) to track device rotation and acceleration, and an Intel Myriad 2 VPU that processes the sensor data which is based on the MA2450 chip. The MA2450 contains several specialized vision processors and a general-purpose processor. These include a stereo depth engine, a motion engine, and a general-purpose image processor. These specialized vision processors are designed to perform specific computer vision tasks, such as stereo depth calculation, object recognition, and image processing, in real-time.

The MA2450 chip also contains a general-purpose processor that is used to control the vision processors and manage the overall operation of the Myriad 2 VPU. This general-purpose processor can be programmed to perform custom tasks, such as data management and communication with other devices.

The multicore architecture of the MA2450 enables the Myriad 2 VPU to perform multiple tasks in parallel, which improves the performance and efficiency of the VPU. Additionally, the chip is designed to be low power, which makes it suitable for use in mobile and battery-powered devices.

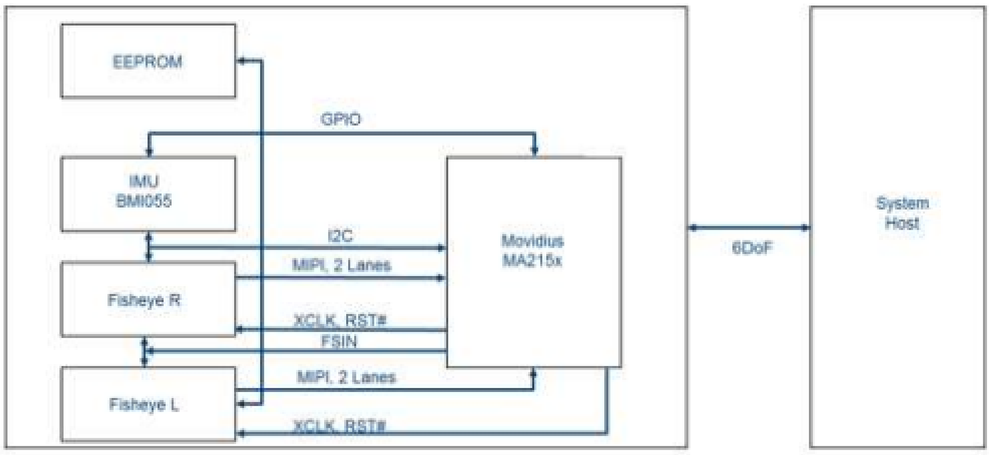


FIGURE 5.13: Intel RealSense T265 Block Diagram

Source: intelrealsense.com

5.4 On board Processing Unit

in this section the "On board Processing Unit" component of the Top Level 5.1 is going to be analyzedThe Nvidia Jetson Nano is a powerful and versatile platform that has been widely adopted in the field of embedded systems and edge computing. It has been used in a variety of applications, including robotics, drones, and smart cameras. In our system, we have chosen to use the Jetson Nano as the base processing unit due to its high performance, low power consumption, and rich set of features. With its powerful CPU and GPU, the Jetson Nano provides the necessary computational power to handle the demanding tasks of our system. Additionally, its small form factor and low power requirements make it an ideal choice for use in portable or embedded systems. With its wide range of interfaces and ports, the Jetson Nano can easily connect to other devices and peripherals, making it a versatile and flexible platform for our system.

In our system, we have chosen to integrate only the CPU of the Jetson Nano in order to keep the power consumption low while still maintaining a high level of performance. This approach allows us to run our system on a smaller and more power-efficient device, making it suitable for use in portable or mobile applications. Additionally, we have not used any other hardware because of the limited resources available in our lab. The Jetson Nano's CPU provides a good balance of performance and power consumption, making it an ideal choice for our system given the available resources in our lab. Using the Jetson Nano's CPU also allows us to take advantage of its rich set of features, such as its support for multiple operating systems and programming languages, which makes it easy for us to develop and deploy our system.

FIGURE 5.14: Nvidia Jetson Nano

Source: developer.nvidia.com

5.4.1 Technical Overview

The Jetson Nano runs on a quad-core ARM Cortex-A57 CPU, which provides high performance and power efficiency. The device comes with 4GB of LPDDR4 memory, which provides ample space for running multiple applications and storing data. Additionally, the Jetson Nano has a variety of interfaces and ports, including Gigabit Ethernet, USB 3.0, HDMI, and 40-pin GPIO, which provide flexibility for connecting to other devices and peripherals. The device also runs on a Linux-based operating system, which provides a familiar programming environment for developers. Without the GPU part, it can still handle tasks that are compute-intensive and real-time, such as image and video processing, machine learning inferencing, and more.

The Cortex-A57 is a 64-bit processor based on ARMv8 architecture, which offers improved energy efficiency, performance and security over its predecessor, the Cortex-A9. The quad-core configuration of the Cortex-A57 allows for efficient multitasking and parallel processing, making it ideal for running multiple applications simultaneously. Additionally, the Cortex-A57 supports a wide range of instruction sets and features such as virtualization, big.LITTLE processing, and NEON advanced SIMD instructions. This makes it a versatile processor that can handle a wide range of tasks, including image and video processing, machine learning inferencing, and real-time data analysis.

5.5 Tools and Frameworks

5.5.1 Robot Operating System (ROS)

For our system's inter-communication between the sensors, the Flight Controller and the central processing module we integrated The Robot Operating System (ROS). ROS is an open-source, meta-operating system for robots. It provides a set of libraries and tools for software development, and is designed to be a flexible framework for building robot applications.

ROS is built on top of the Linux operating system and provides a set of abstractions for common functionality such as communication between nodes, hardware abstraction, and package management.

ROS is composed of several key components, including:

- **Nodes**: These are the individual processes that make up a ROS-based system. They can perform tasks such as sensor processing, control, and decision-making.
- **Topics**: Nodes can publish and subscribe to data on topics, which allows for easy communication between nodes. This allows for a loosely-coupled architecture, where nodes can be added or removed without affecting the rest of the system.
- **Services**: Nodes can also offer and use services, which allow for request-response communication between nodes. Services are typically used for tasks that require a specific response, such as requesting the current robot state.
- **Messages**: ROS uses a standardized message format for communication between nodes. This allows for easy interoperability between different nodes, regardless of the programming language they are implemented in.
- **Parameters**: Nodes can also store and retrieve parameters, which allows for easy configuration of the system.
- **Master**: The master is responsible for registering and tracking nodes, topics, and services. It also provides a naming service that allows nodes to easily find each other.
- **tf**: A transform library that allows for keeping track of the relationship between different reference frames in a robot system.

ROS also provides a set of libraries and tools for common functionality such as navigation, perception, and control. These libraries are designed to be modular and can be easily integrated into a ROS-based system.

ROS is widely used in the robotics community and has a large user base and active development community. It is supported by a number of companies and organizations, and is used in a wide range of applications, including industrial automation, search and rescue, and space exploration.

5.5.2 OpenCV

For this works Image Processing needs we integrated OpenCV (Open Source Computer Vision). OpenCV was founded by Gray Bradsky and it is a library of programming functions mainly aimed at real-time computer vision. It is an open-source library written in C++ and available for C++, Python, and Java. The library has more than 2500 optimized algorithms for image and video analysis. These algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects, extract 3D models of objects, produce 3D point clouds from stereo cameras, etc.

OpenCV is widely used in the field of computer vision, image processing and machine learning. Some of its main features include.

- **Image processing**: OpenCV provides a wide range of image processing functions, including filtering, color space conversion, thresholding, edge detection, and feature extraction..
- **Object detection**: OpenCV provides pre-trained classifiers for object detection, including face detection, upper body detection, full body detection, and more.
- **Video analysis**:It provides functions for video analysis such as motion estimation, background subtraction, and object tracking.
- **Camera calibration**: OpenCV includes functions for camera calibration, which can be used to correct lens distortion and other lens artifacts.
- **3D reconstruction**:It provides functions for 3D reconstruction, including stereo reconstruction and structure from motion.
- **Hardware acceleration**:OpenCV can leverage the power of various hardware platforms such as GPUs and TPUs to accelerate its algorithms.

5.6 Implementation

Beneath, we are presenting a diagram that illustrates the code structure of the system's implementation. The diagram shows the different modules and components that make up the system, and how they interact with each other. It provides a clear and concise overview of the system's architecture, and how the different components are connected and interact with each other. The diagram also shows the relationship between the different layers of the system, such as the Optical layer, the processing layer, and the navigation layer. This diagram is a useful tool for understanding the system's implementation and for identifying any potential areas for improvement or optimization. It will help in understanding the flow of the system, any dependencies and how the system interacts with external devices.

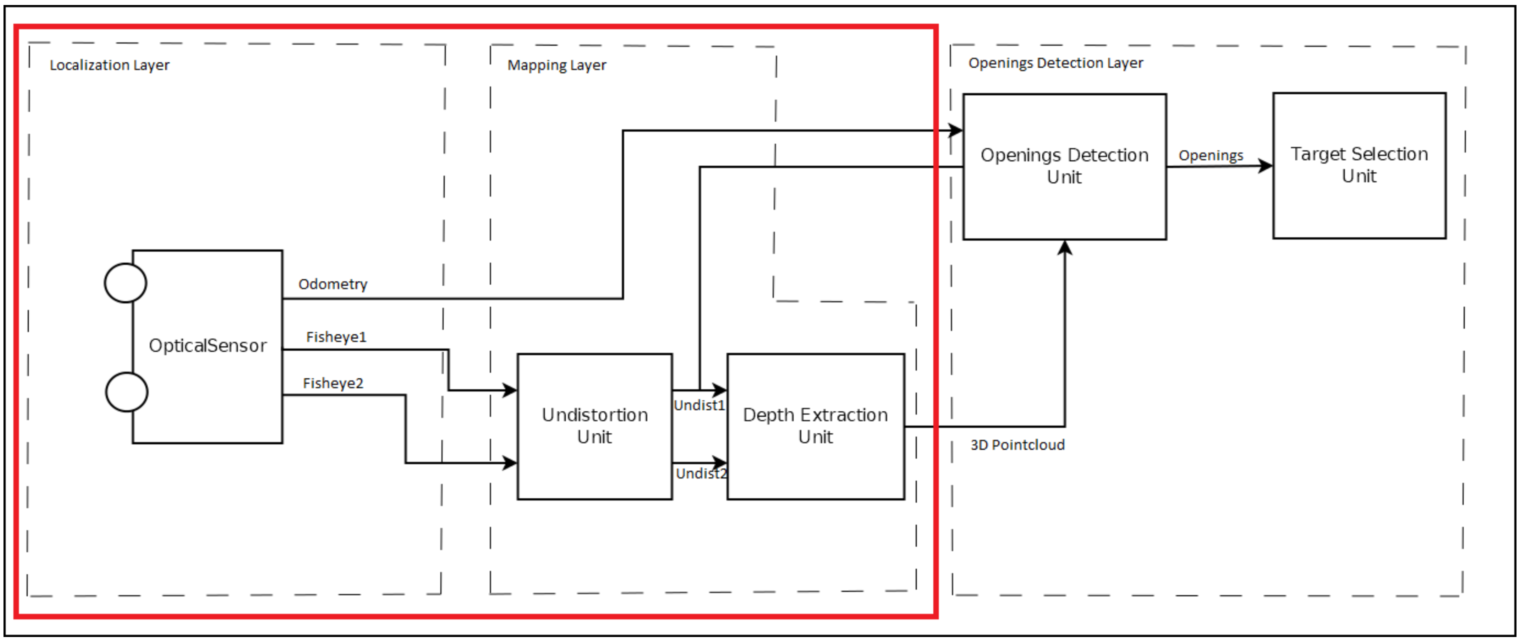


FIGURE 5.15: Implementation Block Diagram

5.6.1 Localization and Mapping

The first layer of our system is the localization layer, where the our systems input is acquired by our optical sensor capturing frames with its fisheye cameras. This Intel Realsense T265 utilizes visual odometry (vSLAM) to produce precise and accurate estimates of our systems position and orientation in real-time. The fisheye cameras provide a wide field of view, which helps to improve the robustness of the vSLAM algorithm by allowing it to extract key features and track them over time, even in challenging conditions. Overall, this localization layer forms the foundation for the rest of the system.

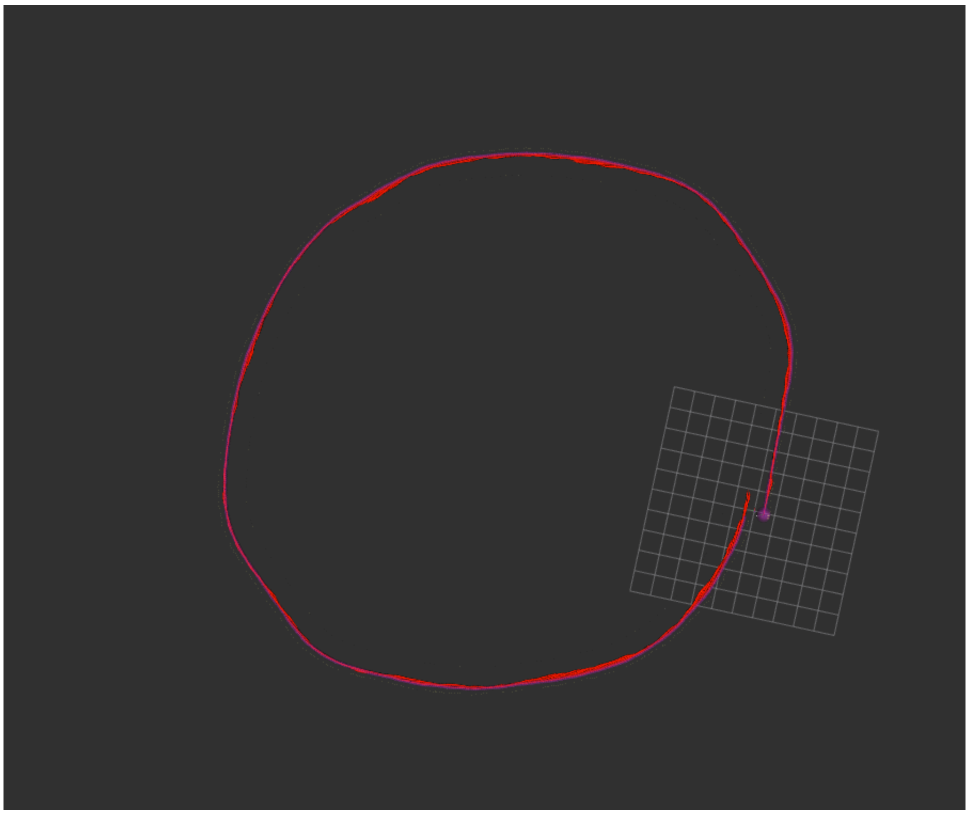


FIGURE 5.16: Odometry Extraction

The Mapping Layer consists of two components. The Undistortion component and the Depth Extraction component.

Undistortion Component

The first component of the mapping layer is the undistortion component. It takes as input the two fisheye streams and it produces as output the undistorted forms of these two streams. Since the two fisheye lenses are already calibrated The system has already loaded the camera intristic and extrinsic parameters in memory in order to perform the undistortion. The undistortion takes place by utilizing openCV function "cv::fisheye::initUndistortRectifyMap". This function is used to generate maps for image rectification and undistortion. These maps can be used with the cv::remap function to transform an image captured by a fisheye camera into a rectified image, which has reduced distortion and improved geometric properties. The function takes in several parameters such as the camera matrix, distortion coefficients, and the size of the output image. Additionally, it also allows the user to specify the rectification method (e.g. "stereographic", "equidistant", "equisolid", etc.) which determine how the undistorted image will be projected. The output of the function is a pair of maps (one for x-coordinates and one for y-coordinates) that can be used with the cv::remap function in order to produce the final undistorted versions of the fisheye images.

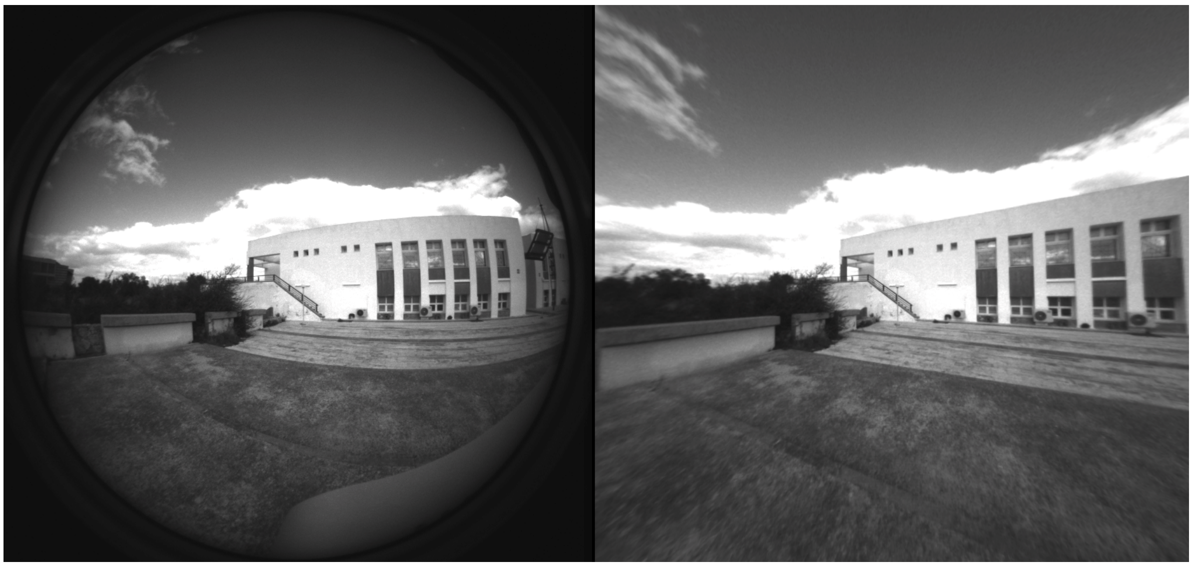


FIGURE 5.17: Undistortion

The second component of the mapping layer is divided into two submodules. The Disparity Estimation module and the Pointcloud Generation module.

Disparity Estimation

The Disparity module is responsible for calculating the disparity map between the two undistorted fisheye images. The disparity represents the relative depth of each pixel in the images, which is used to create a 3D representation of a scene. The method for producing the disparity is by integrating OpenCV's Stereo SGBM matcher function where by passing to it the appropriate parameters that are based on the current camera model it produces the shift of pixels in the left image relative to the corresponding pixels in the right image. At this work in order to achieve sufficient results, the disparity needs to be continuous and detailed enough for it to provide depth values for almost every pixel, thus we are passing the matcher output from a WLS filter smoothing it out significantly.

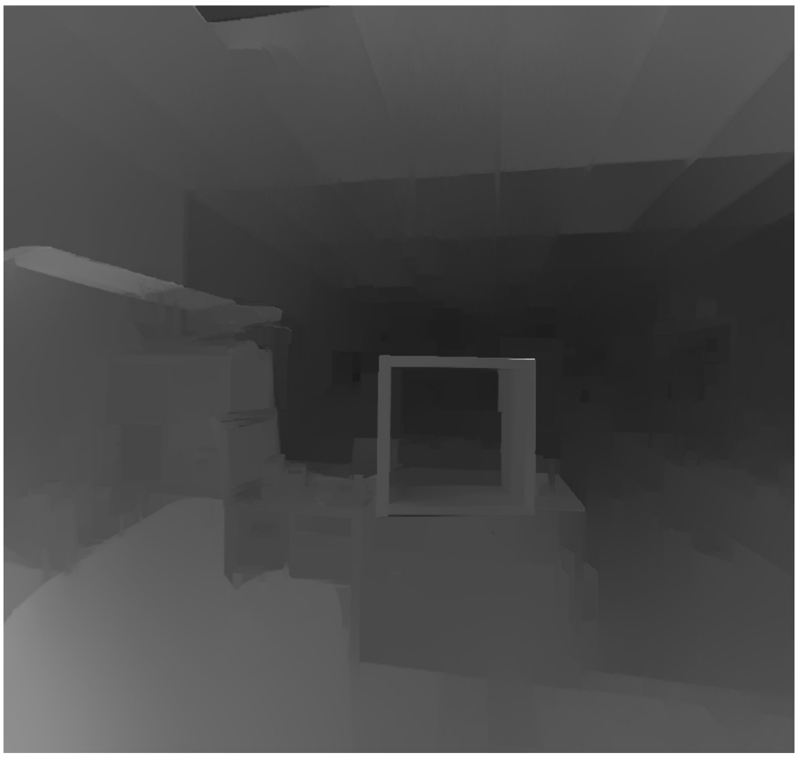


FIGURE 5.18: Disparity

Pointcloud Generation

The last step of the depth extraction is the Pointcloud generation where the final 3D map is constructed from the disparity feed. The procedure we integrate to achieve this is by applying Stereo Vision using the formula.

$$depth = (focal_length * baseline)/disparity \tag{5.1}$$

The focal length is the distance between the image plane and the camera lens, and it determines the field of view of the camera. baseline is the distance between the two cameras in the stereo setup.

So the equation calculates the depth of a pixel as the ratio of the product of focal length, baseline, and 16 to the disparity value. The lower the disparity value, the greater the depth.

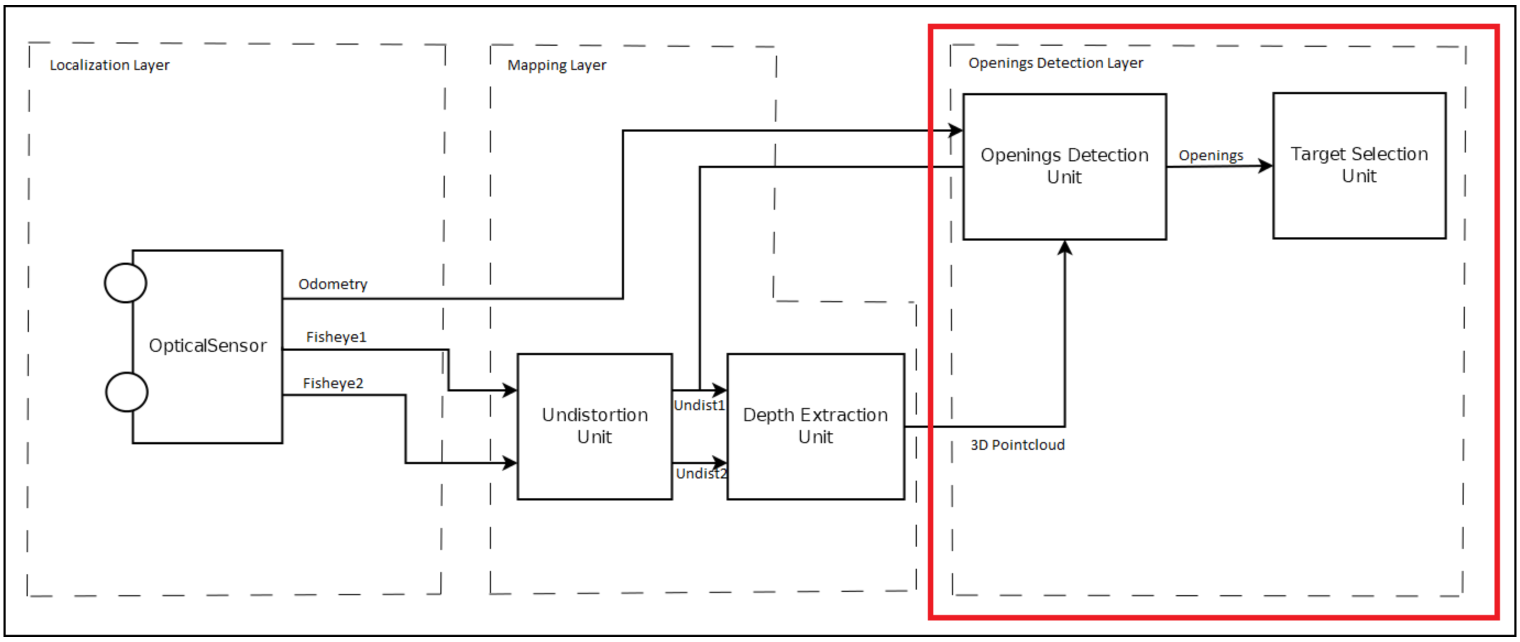


FIGURE 5.19: Implementation Block Diagram

5.7 Openings Detection Layer

The Openings Detection layer is the last stage of our pipeline as it receives the produced Odometry from the Localization layer, the left undistorted fish-eye feed and the Disparity map from the Mapping Layer and it outputs the most suitable target for the drone to come through in the surrounding environment. This layer is splitted in two different modules. The Openings Detection unit and the Target Selection unit. Algorithm 1 represents the high level logic of the Openings Detection feature.

Algorithm 1 Openings Detection Top Level

```
1: $openings \leftarrow []$
2: $frames \leftarrow$ left undistorted image
3: $rectangulars \leftarrow$ DetectRectangulars()
4: for $rectangular$ in $rectangulars$ do
5:     $valid \leftarrow$ compute3D($rectangular$)
6:     if $valid$ then
7:         $openings.append(rectangular)$
8: TargetSelection($openings$)
```

5.7.1 Opening Detection Unit

The Openings Detection is taking as input an undistorted image and it outputs detected rectangulars that have opening properties. The opening properties is firstly to have sufficient depth, and secondly to it's area to be sufficiently large in order for the vehicle to fit through it. The first step of the Openings Detection Unit is the **Rectangulars Detection** 2. On this step the image is passed through OpenCV's Canny Edge Detector, with specific threshold and window size specifically tuned for our camera model, afterwards the edges are passed through OpenCV's Hough Line transform in order to acquire all the potential lines of the image. After the lines are acquired, we are segmenting them regarding their relative angle to the x-axis in horizontal and vertical. The next step is to find intersections between those lines and start combining these intersecting points to detect rectangulars. For dealing with the fact that in some cases the number of the intersecting points is significantly large, we simplify the process by classifying the intersecting points into vertical and horizontal lines forming a grid of potential openings in the picture. Finally for each rectangular inside the grid we examine the disparity values in the corresponding disparity input and we calculate the area of them, in order to determine which of these rectangulars can act as openings.

Algorithm 2 DetectRectangulars

```
1: function DetectRectangulars($undistorted_left$)
2:     $edges \leftarrow$ CannyEdgeDetector($undistorted_left$)    ▷ Apply Canny Edge Detection to extract the edges of the Image
3:     $lines \leftarrow$ HoughLines($edges$)    ▷ Apply Hough Lines Transform to the edges to extract lines of the image
4:     $segmented \leftarrow$ SegmentByAngle($lines, undistorted_left$)    ▷ Segment the lines by angle to estimate if a line is horizontal or vertical
5:     $intersections \leftarrow$ SegmentedIntersections($segmented$)    ▷ Detect the intersections between the horizontal and vertical lines
6:     $verticals \leftarrow$ SplitToVertical($intersections$)    ▷ Classify the intersection based on the vertical line of the point
7:     $rectangulars \leftarrow$ FormRectangulars($verticals$)    ▷ Form the final Rectangulars by combining the produced vertical points
8:     return $rectangulars$
```

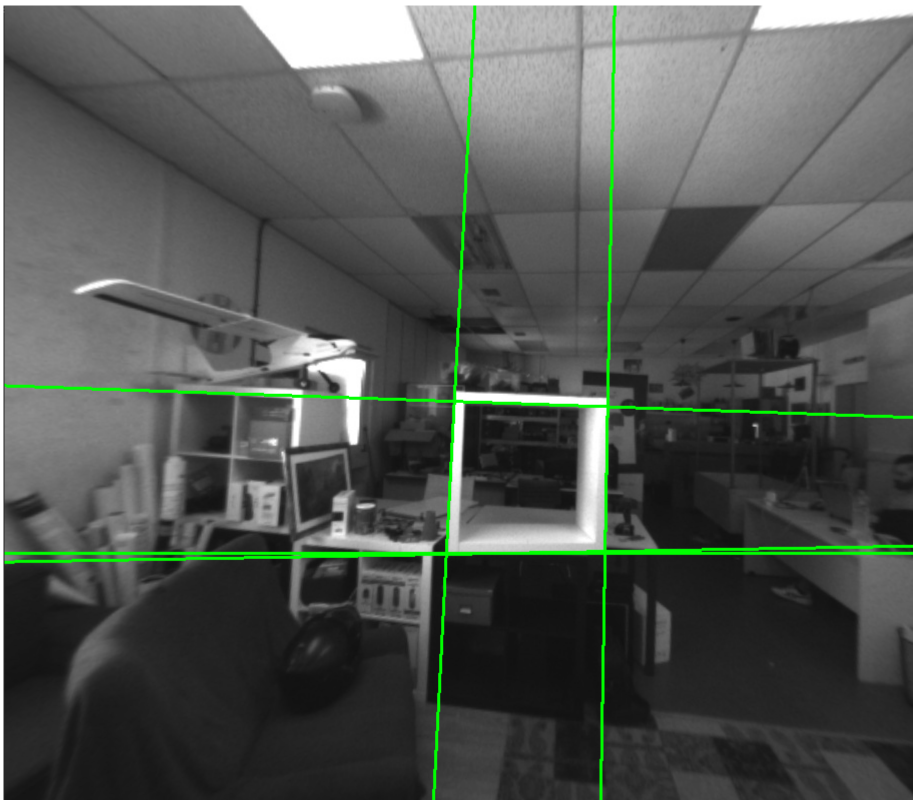


FIGURE 5.20: Rectangular Extraction

5.7.2 Target Selection Unit

The Target Selection is the following and last process of the system. It receives the detected openings of of the previous modules and it extracts a single target. This module provides the system with two features. The first is an automated selection where the system estimates the closest target to the center of the camera and it locks it. The second one is manual selection of target where the operator selects the target by passing input to the system through the RC controller.

Chapter 6

System Verification and Performance Evaluation

In this chapter, we showcase and assess the outcomes of our work. The aim of this chapter is to provide an all-inclusive assessment of the system's performance and exhibit the results of our efforts. We will display the data and metrics that we employed to appraise the system, including experiments and tests that we carried out. Additionally, we will furnish a meticulous evaluation of the results, comprising the advantages and drawbacks of our system, and any valuable knowledge that we gained from the data.

6.1 Specification of Compared Platforms

For evaluating and testing the system we performed field tests on both Jetson Nano and an Ubuntu Desktop environments in order to observe the system behaviour on each platform regarding the power consumption,latency, scalability and accuracy of the results. An important feature of the system is that in the way it is developed, it can be integrated into any Ubuntu platform giving it the flexibility to change processing platform regarding on the need of each potential application.

6.1.1 Nvidia Jetson

The NVIDIA Jetson series are embedded computing boards known for their low weight and power consumption. Nvidia Jetson was selected as a host platform due to the current availability on our lab. It could be hosted on a different ubuntu embedded platform as Raspberry Pi with the same specifications for the same perfomance

Spec	Nvidia Jetson Nano
CPU	Quad-core ARM Cortex-A57 MPCore processor @ 1.47 GHz
RAM	4 GB 64-bit LPDDR4, 1600MHz
Storage	16 GB eMMC 5.1
Size	69.6 mm x 45 mm
Power	5 Watts

Spec	Nvidia Jetson Agx Xavier
CPU	8-core Nvidia Carmel ARM v8.2 64-bit CPU
RAM	16 GB 256-bit LPDDR4x @ 2133 MHz
Storage	32 GB eMMC 5.1
Size	87mm x 16mm
Power	10-30 W

6.1.2 Ubuntu Desktop Computer

Tests were also performed on a Dekstop computer, to observe the system's behaviour on an environment providing much more resources and power.

Spec	Ubuntu Desktop
CPU	six core Intel Core i5-11400H @ 4.50 GHz
RAM	16 GB DDR4, 3200 MHz
Storage	512 GB PCIe 3.0 NVMe M.2 SSD
Size	359 x 256 x 24,7 mm
Power	150 Watts

6.2 Datasets and Performance Metrics

The datasets used in this work were collected by us, covering a variety of environments to provide a comprehensive evaluation of the system. This allows for testing the system under various conditions and helps to demonstrate its performance in different settings, which is crucial for real-world applications. By including a range of environments, the results of the evaluation provide a more robust and accurate picture of the system's capabilities. This helps to ensure that the system's performance is not just dependent on a single type of environment, but instead represents a generalization of its abilities across different scenarios.

The performance of the system is evaluated using two main metrics: frames per second (FPS) and accuracy of the detected openings. FPS measures the speed at which the system can process images based on the pipeline describes in the previous chapter, while the accuracy of the detected landmarks indicates how closely the system's output aligns with the actual position of landmarks in the real world. By evaluating the system using both FPS and accuracy, we can get a comprehensive understanding of its performance and identify areas for improvement. Additionally, testing the system in a variety of environments helps us to ensure that it performs well under different conditions and with different types of input data.

6.3 Evaluation

6.3.1 Localization

This subsection focuses on presenting the localization outputs from two environments to assess the precision of our localization module. Our localization system is crucial for accurately determining the location of an object or device in real-time. In order to verify the performance of our module, we attached the odometry output of our localization module to satellite images from google maps. Also we attach images taken from our system during the process of aquiring the odometry to verify or data.

The Visual Odometry is being produced by the system on a rate of 30FPS in all the platforms, ensuring that the UAV would have very detailed information about it's pose, even on more difficult circumstances.

Dataset from Science Building, Technical University of Crete

This dataset includes a route to the circular core of Ktirio Epistimon building in Technical University Of Crete.

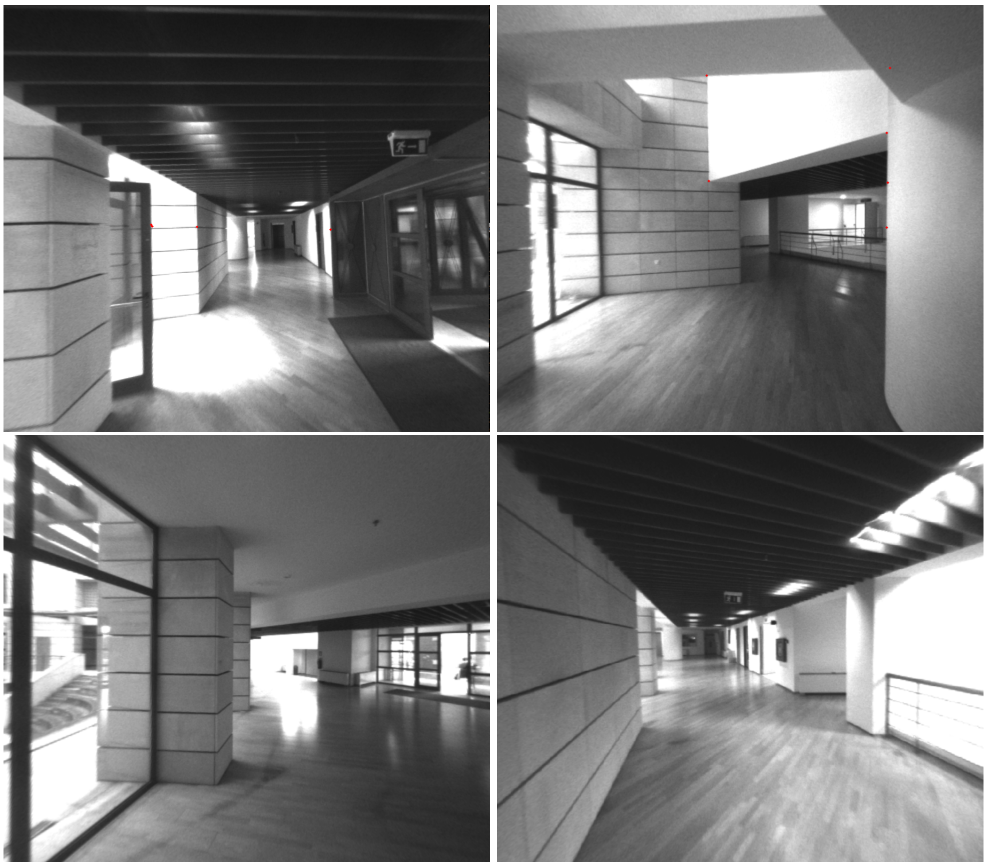


FIGURE 6.1: Sample images from route

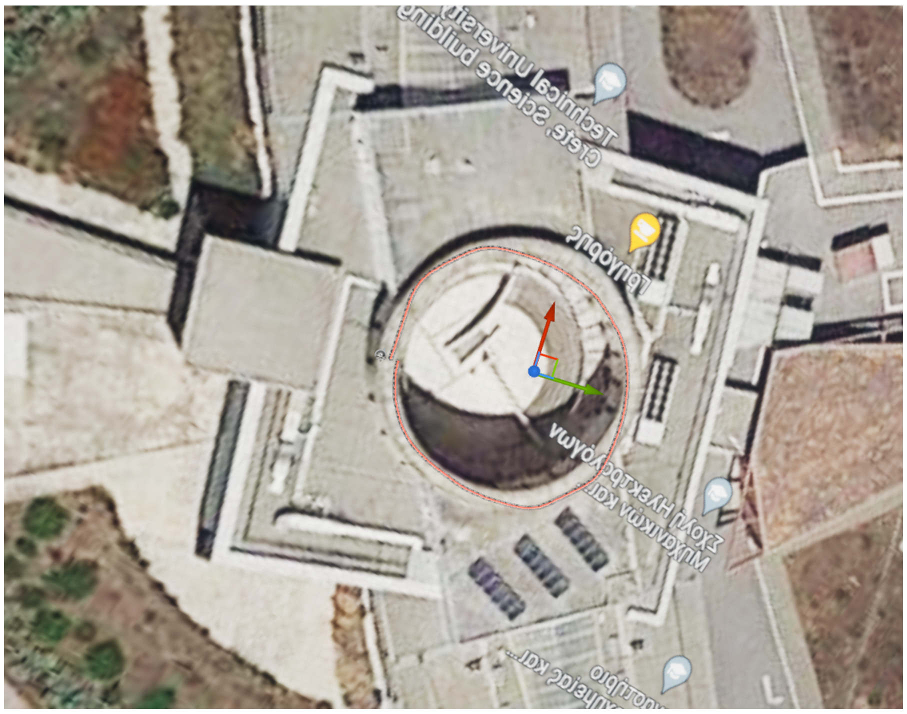


FIGURE 6.2: Odometry Output

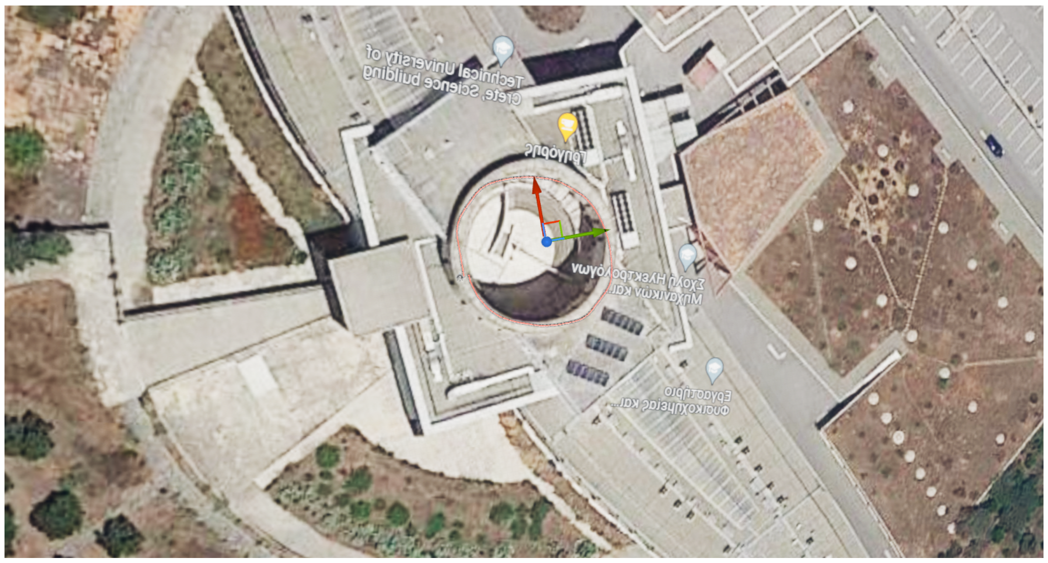


FIGURE 6.3: Odometry Output

Dataset from MHXOP, Technical University of Crete

This dataset includes a route on around the floor outside SenseLab at MHXOP building in Technical University of Crete.

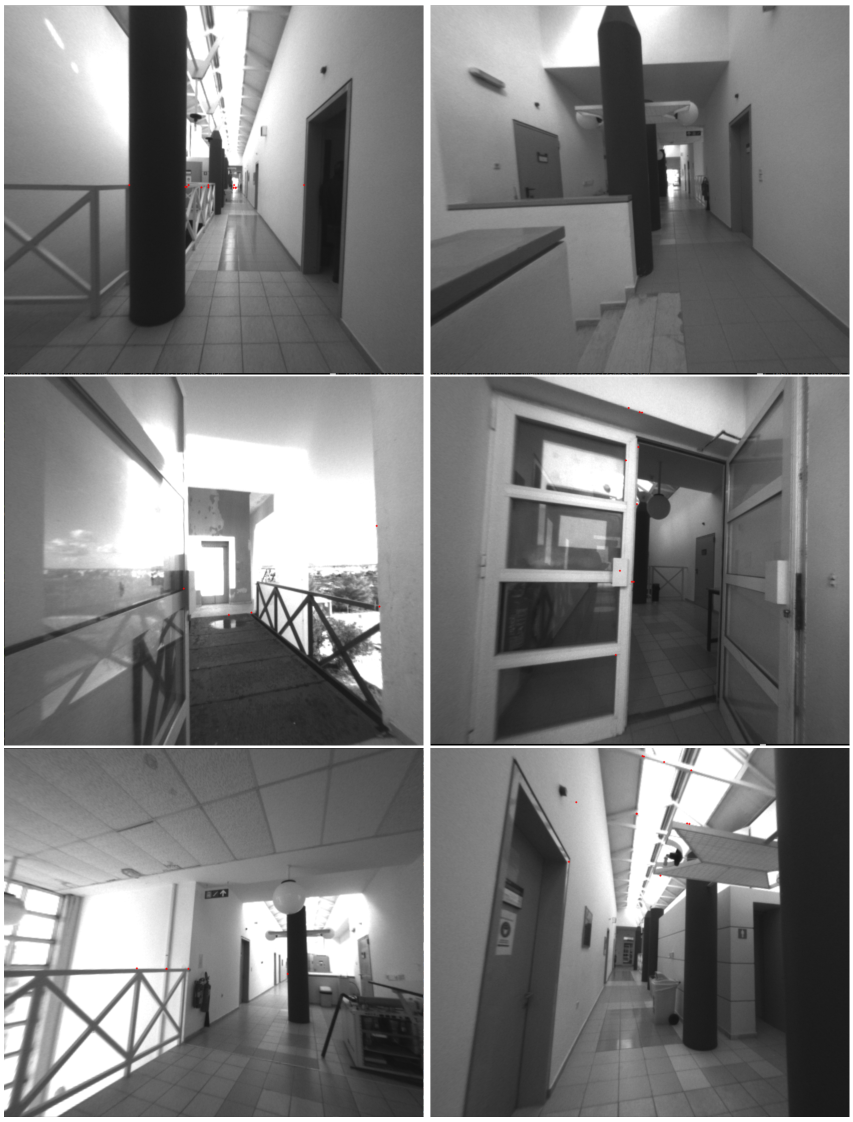


FIGURE 6.4: Sample images from route

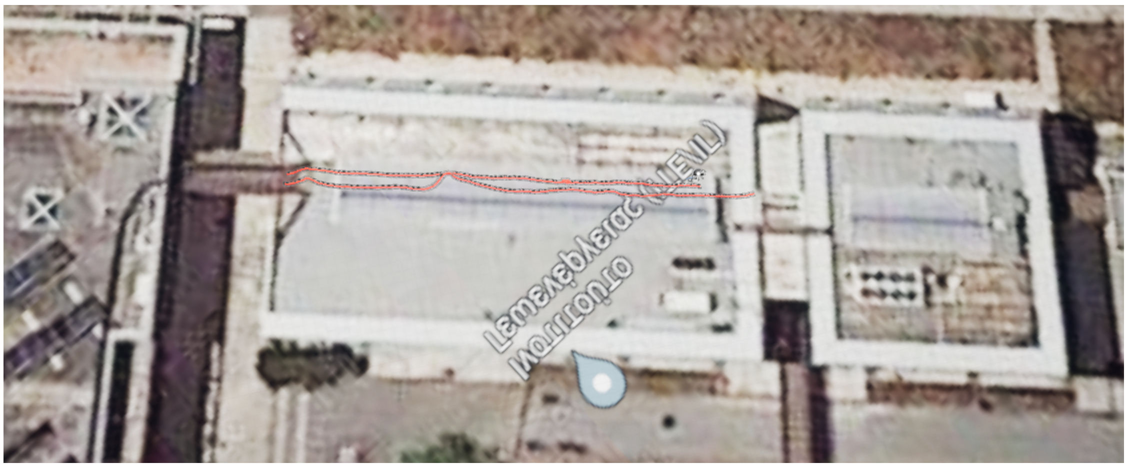


FIGURE 6.5: Odometry Output

6.3.2 Mapping

To evaluate the Mapping feature of the system, we test several snapsots of the undistorted image of the camera and their corresponding pointclouds. The system requires space perception in order to be able to navigate itself through an indoor environment.Therefore there must be information about the basic layout of the space for detecting the presence of obstacles and other kind of blockers.

The system is able to perform the Mapping task with the rate of 15.4 FPS ON Ubuntu Desktop, 4,7 FPS On Jetson Xavier and 1,7 FPS on Jetson Nano. Having 3D mapping data in that frequency, the system is eligible of gaining space perception constantly without information loss if we take as a fact that on inspection missions the speed of the UAV is relatively slow.

The results have shown that the Mapping Module is delivering accurate represantations of the surrounding environment making it capable of "sensing" which directions have open space for it to navigate and which spots contain obstacles. (Figures 6.6, 6.7, 6.8, 6.9).

6.3.3 Openings Detection

The Openings Detection was tested in several environments and it is observed that in it's current state, it is able to detect openings in realtime under specific circumnstances. Firstly, the openings center must be as near as possible to the camera center. Secondly, the opening must be rectangular as the methods integrated for acquiring the openings use detect lines. Lastly

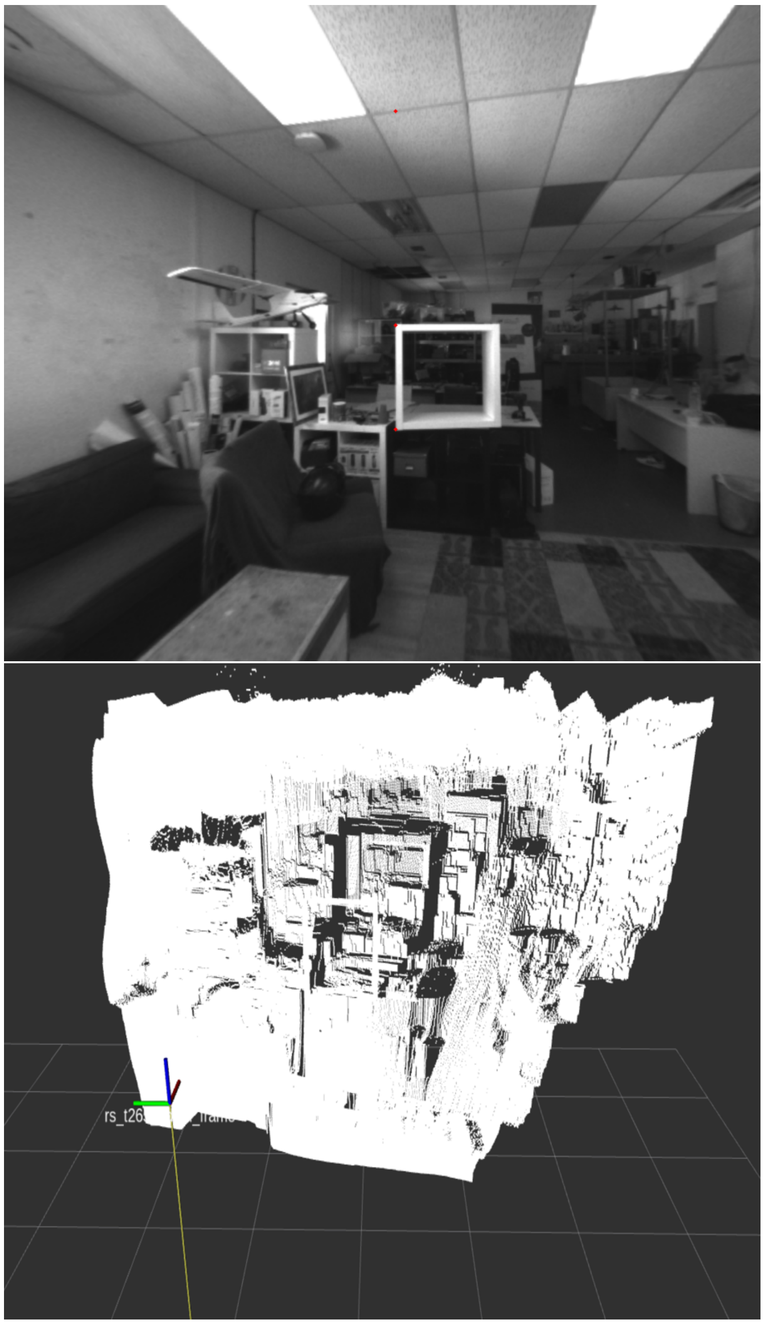


FIGURE 6.6: Snapsot 1

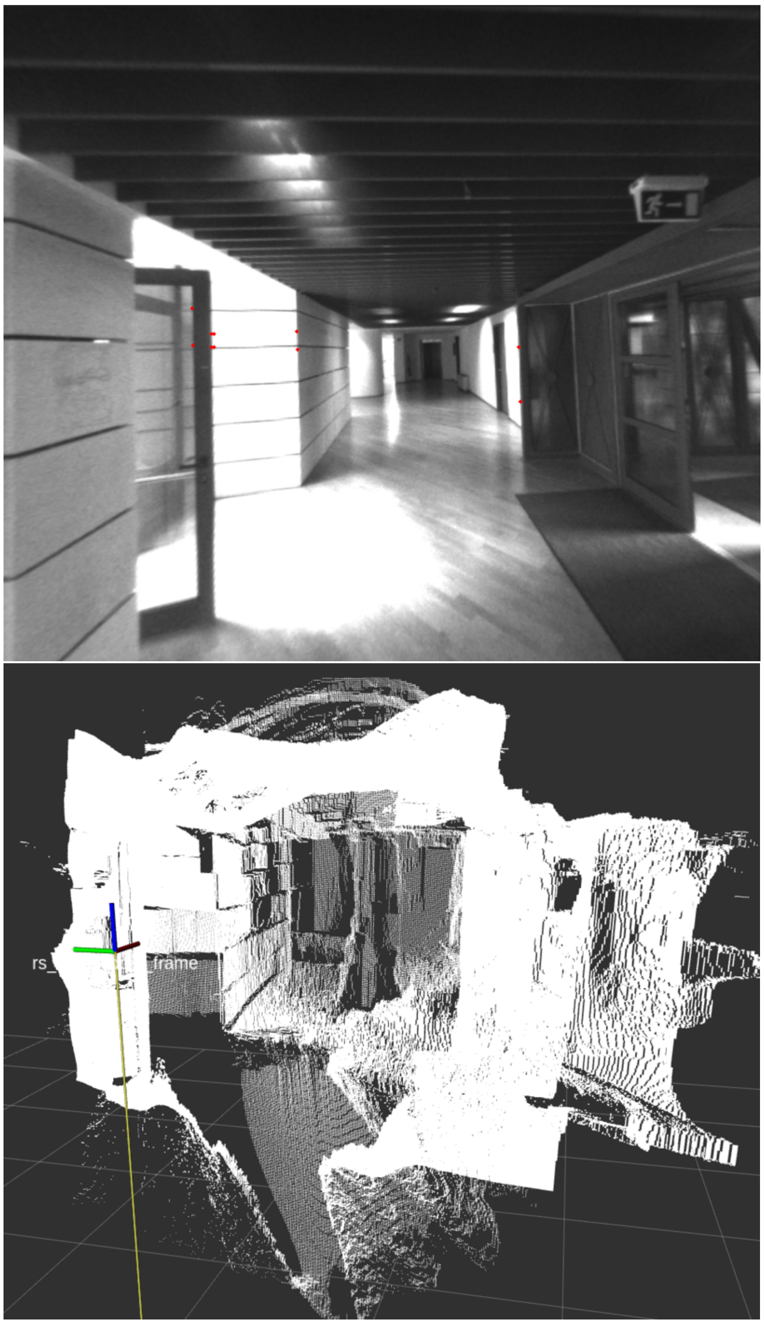


FIGURE 6.7: Snapsot 2

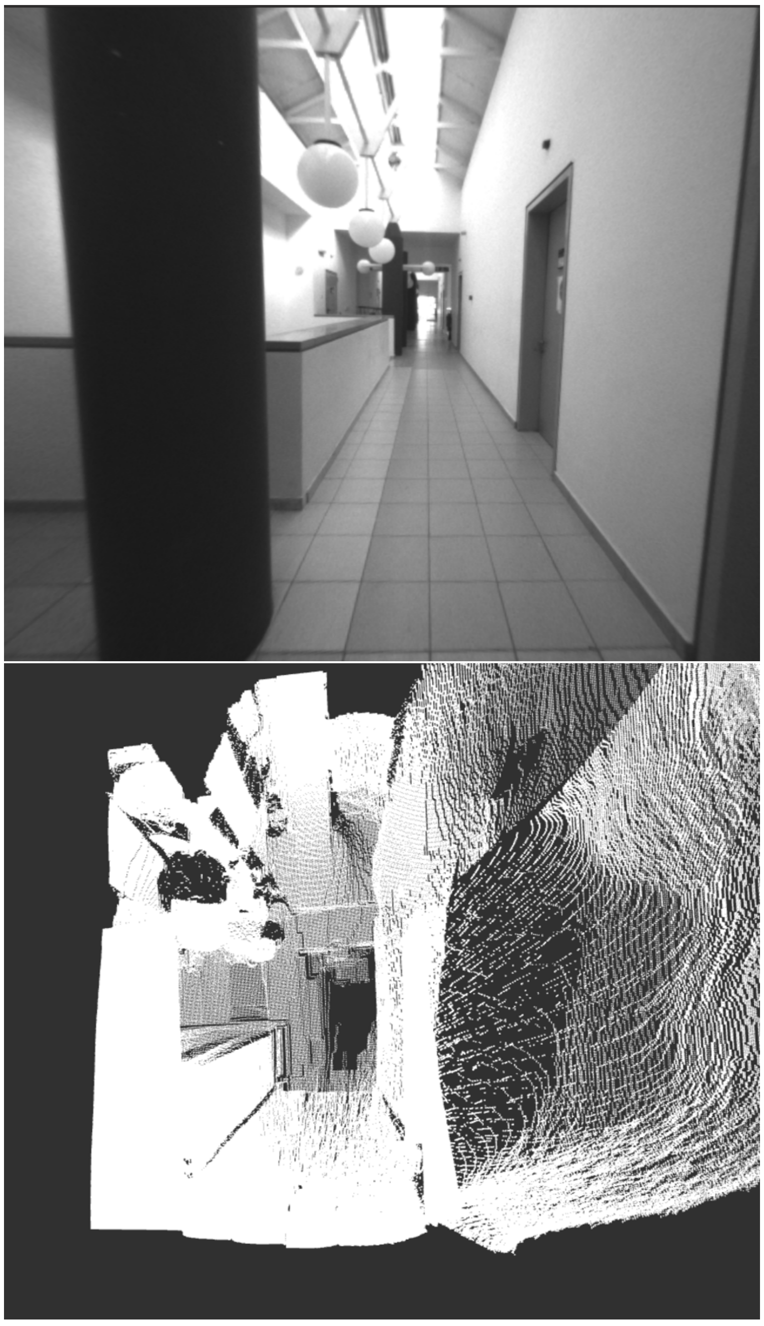


FIGURE 6.8: Snapsot 3

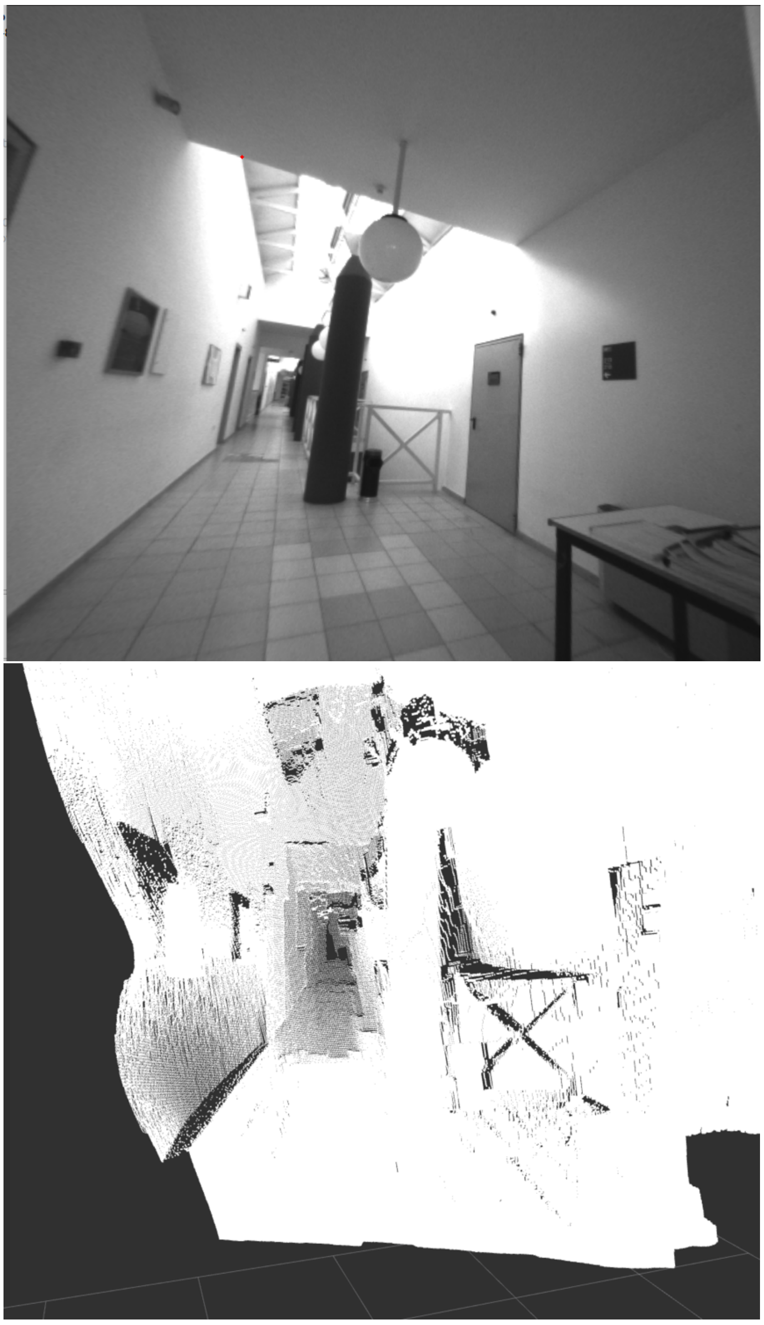


FIGURE 6.9: Snapsot 4

the Openings Detector operates better when there is more light in the scene rather that in dark scenes.

The system is able to perform the openings detection pipeline with the rate of 25 FPS ON Ubuntu Desktop, 15 FPS on Jetson Xavier and 10 FPS on Jetson Nano. This specific rate is sufficient for examining all the potential areas of the environment that might contain an escape opening.

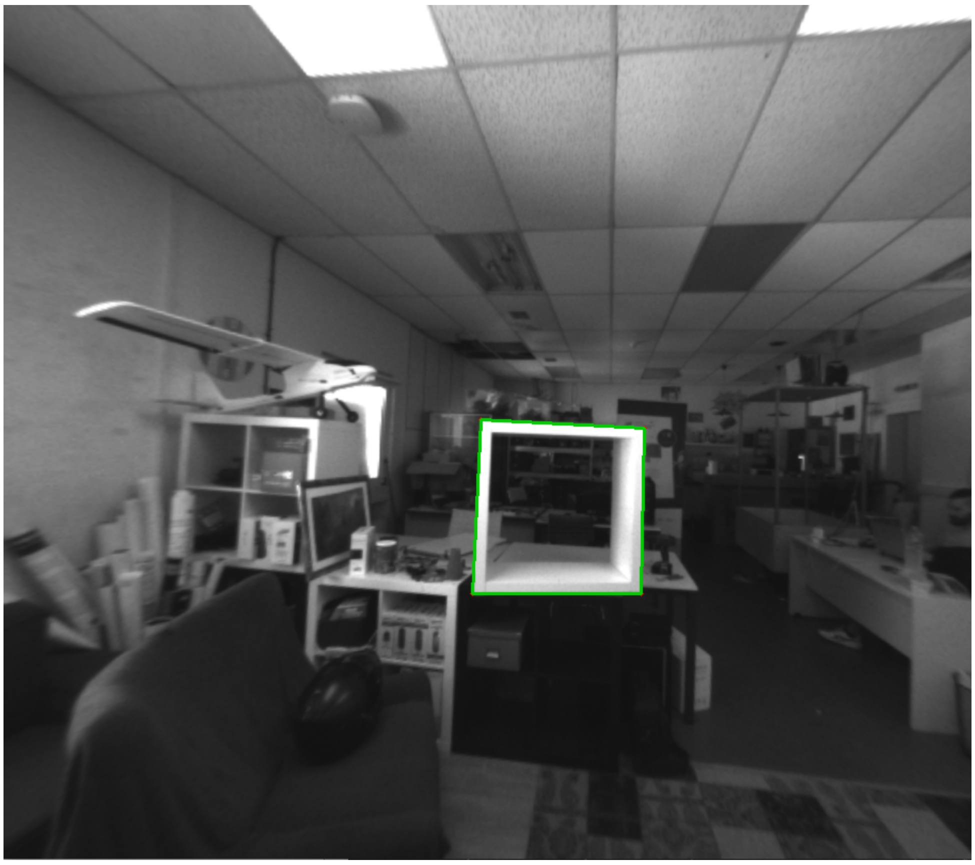


FIGURE 6.10: Openings Detection

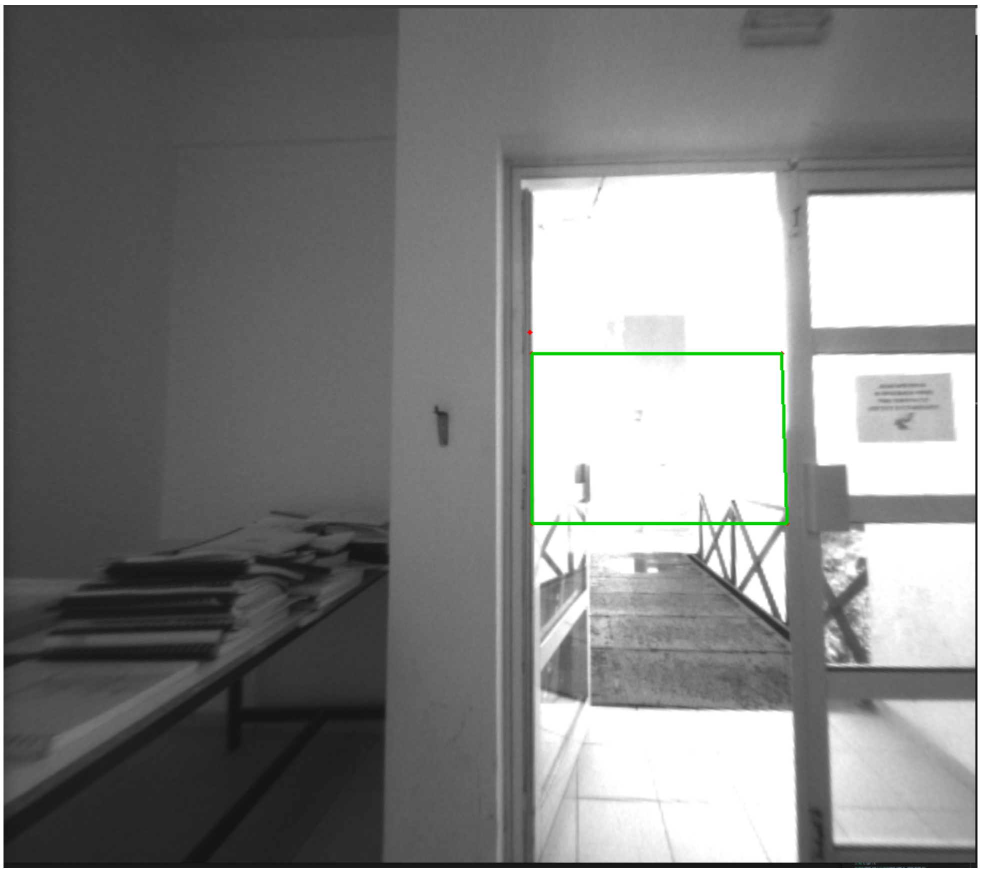


FIGURE 6.11: Openings Detection

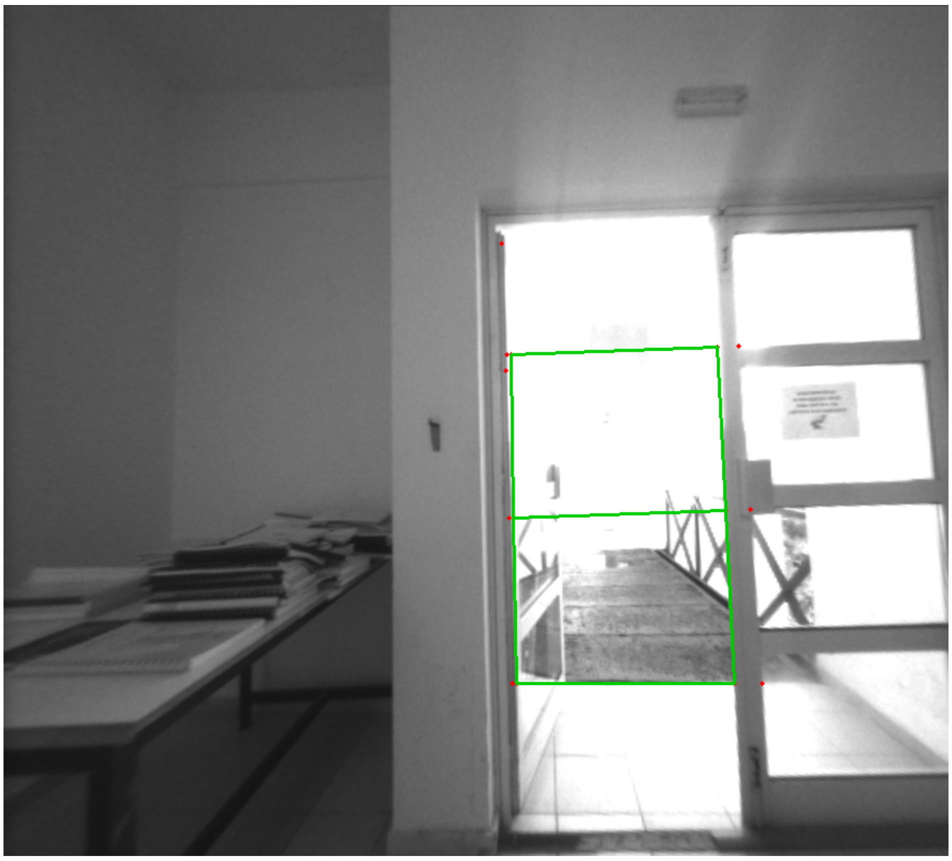


FIGURE 6.12: Openings Detection

Chapter 7

Conclusions and Future Work

7.1 Conclusions

The evaluation of the localization and mapping system has revealed that sufficient results can be obtained with a reduction in hardware modules. Unlike related work that typically employs two hardware modules, our approach utilizes only one, resulting in a significant reduction in both cost and power consumption. This not only provides cost-saving benefits, but also reduces the overall energy footprint of the system, making it more sustainable. The results of this evaluation emphasize the importance of optimizing hardware utilization in the development of localization and mapping systems and demonstrate that high performance can still be achieved with a streamlined design.

The results of the evaluation of the openings detection module, developed using computer image processing techniques, show that while it may not be as accurate as the CNN solutions used in related work, it still delivers satisfactory results. Despite this, there is potential for further improvement, and with the incorporation of more techniques, this module can become as reliable as the other solutions. This study highlights the value of continued research and development in the field of computer image processing and its potential for delivering accurate and efficient solutions for openings detection. We believe that with continued effort and refinement, our approach can become a competitive alternative to existing CNN solutions.

In conclusion, our evaluation of the system has shown promising results in terms of accuracy and reliability. By reducing energy consumption and hardware modules used in the construction of these systems, we are able to achieve similar performance on most factors while also extending the operation time and reducing the overall cost. This demonstrates that effective and efficient design can lead to better outcomes without sacrificing performance. The results of this study highlight the importance of considering energy consumption and hardware utilization in the design process, and we hope to continue exploring these strategies in future developments.

7.2 Future Work

7.2.1 Openings Detection Improvement

As future work, we firstly aim to enhance the performance of our openings detection algorithm to make it even more accurate and robust. This will involve a thorough analysis of current limitations and a systematic approach to address these issues, incorporating advanced image processing techniques and adding more filtering layers to our results. By improving the accuracy and robustness of the algorithm, we hope to increase its reliability and make it capable of handling more difficult detection tasks."

7.2.2 Autonomous Navigation

Another exciting area for future work is the implementation of an autonomous navigation module using the space perception capabilities that we have developed. By utilizing the existing base of our localization and mapping system, we can enable devices to navigate through their environment with great independence. This will require the integration of advanced Artificial Intelligence navigation algorithms into our localization and mapping system. The result will be a comprehensive solution for autonomous navigation that leverages the strengths of our existing system.

References

[4] Niklas Karlsson Enrico Di Bernardo Jim Ostrowski Luis Goncalves Paolo Pirjanian and Mario E. Munich. "The vSLAM Algorithm for Robust Localization and Mapping". In: *10.1109/ROBOT.2005.1570091* (Jan. 2005), pp. 24–29. URL: https://www.researchgate.net/publication/221073651_The_vSLAM_Algorithm_for_Robust_Localization_and_Mapping.

[5] Larry Matthies. "Dynamic Stereo Vision". In: (Oct. 1989). URL: http://reports-archive.adm.cs.cmu.edu/anon/1989/CMU-CS-89-195.pdf.

[10] Man-Woo Park Atefe Makhmalbaf Ioannis Brilakis. "Comparative study of vision tracking methods for tracking of construction site resources". In: (Sept. 2011). URL: https://www.sciencedirect.com/science/article/pii/S0926580511000343?fr=RR-2&ref=pdf_download&rr=79bf6e7408baee83.

[11] Hess Wolfgang et al. "Real-time loop closure in 2D LIDAR SLAM". In: *2016 IEEE International Conference on Robotics and Automation (ICRA)* (2016), pp. 1271–1278. URL: https://ieeexplore.ieee.org/abstract/document/7487258.

[12] Pileun Kim Jingdao Chen Yong K.Cho. "SLAM-driven robotic mapping and registration of 3D point clouds". In: *Science Direct* (Sept. 2018). URL: https://www.sciencedirect.com/science/article/pii/S0926580517303990.

[13] Krul Sander et al. "Visual SLAM for Indoor Livestock and Farming Using a Small Drone with a Monocular Camera: A Feasibility Study". In: *Drones* (2021). URL: https://www.mdpi.com/2504-446X/5/2/41.

[14] Endres Felix et al. "3-D Mapping With an RGB-D Camera". In: *IEEE Transactions on Robotics* 30.1 (Jan. 2014), pp. 177–178. URL: http://link.aip.org/link/?RSI/62/1/1.

[15] Shinya Kawabata Kodai Nohara Jae Hoon Lee Hirotatsu Suzuki Takeaki Takiguchi Oh Seong Park and Shingo Okamoto. "Autonomous Flight Drone with Depth Camera for Inspection Task of Infra Structure". In: *e International MultiConference of Engineers and Computer Scientists* 2 (Mar.

2018). URL: http://www.iaeng.org/publication/IMECS2018/IMECS2018_pp804-808.pdf.

[16] Fernández-Caramés Moreno Curto Rodríguez-Aragón Serrano. "A Real-time Door Detection System for Domestic Robotic Navigation". In: *Journal of Intelligent Robotic Systems* (2014), pp. 119–136. URL: https://doi.org/10.1007/s10846-013-9984-6.

[17] Alparsla Onder and Cetin Omer. "Fast and Effective Identification of Window and Door Openings for UAVs' Indoor Navigation". In: *2021 International Conference on Unmanned Aircraft Systems (ICUAS)* (2021). URL: https://ieeexplore.ieee.org/abstract/document/9476840/metrics#metrics.

External Links

[1] “Simultaneous localization and mapping”. In: (). URL: https://en.wikipedia.org/wiki/Simultaneous_localization_and_mapping.

[2] “Aircrafts”. In: (). URL: https://en.wikipedia.org/wiki/Aircraft.

[3] “Unmaned Aerial Vehicles”. In: (). URL: https://en.wikipedia.org/wiki/Unmanned_aerial_vehicle.

[6] “Stereo Disparity Using Semi-Global Block Matching”. In: (). URL: https://www.mathworks.com/help//visionhdl/ug/stereoscopic-disparity.html.

[7] “Canny Edge Detector”. In: (). URL: https://en.wikipedia.org/wiki/Canny_edge_detector.

[8] “Canny Edge Detector OpenCV”. In: (). URL: https://docs.opencv.org/4.x/da/d22/tutorial_py_canny.html.

[9] “Hough Transform wiki”. In: (). URL: https://en.wikipedia.org/wiki/Hough_transform.