VISION-BASED NAVIGATION OF HETEROGENEOUS ROBOTS

by

Selim Ahmet Iz

Submitted to
the Graduate School of Engineering and Natural Sciences
in partial fulfillment of
the requirements for the degree of
Master of Science

SABANCI UNIVERSITY

December, 2022

Vision-based Navigation of Heterogeneous Robots

Selim Ahmet Iz

ME, Master's Thesis, 2022

Thesis Supervisor: Prof. Dr. Mustafa Unel

Keywords: Path Planning, Terrain Analysis, Stereo Depth Reconstruction, Structure from Motion, Image Stitching, Mobile Robots, System Identification

Abstract

Vision-based navigation is one of the most attractive research subjects on mobile robot applications. An image-based navigation strategy was developed for a collaborative UAV-UGV system in this thesis. The work focuses primarily on terrain depth analysis, terrain image stitching, path planning, and motion control of the robot system. Novel algorithms for path planning and image stitching were developed as part of the thesis. In the terrain depth analysis, both stereo and structure from motion problems were tackled and their comparative advantages were highlighted. In addition, the performance of the onboard vision system used on UAV was evaluated using the system identification techniques to check its suitability for UAV motion control. In developing the new image stitching algorithm, inertial data was also used in addition to image data, and more accurate results were obtained than pure vision-based techniques. The developed path planning algorithm provided faster and more accurate results than the well-known A* and PRM, two widely used path planning algorithms. Simulation and experimental results obtained from the implementation of the algorithms show the effectiveness of the developed techniques for the collaborative navigation of heterogeneous robot systems.

Heterojen Robotların Görüntü Tabanlı Navigasyonu

Selim Ahmet Iz

ME, Master Tezi, 2022

Tez Danışmanı: Prof. Dr. Mustafa Unel

Anahtar kelimeler: Yol Planlaması, Arazi Analizi, Stereo Derinlik Rekonstrüksiyonu, Hareketten Yapı, Görüntü Dikişi, Mobil Robotlar, Sistem Tanımlama

Özet

Vizyona dayalı navigasyon, mobil robot uygulamaları konusunda en ilgi çekici araştırma konularından biridir. Bu tezde işbirliğine dayalı bir İHA-UGV sistemi için görüntü tabanlı bir navigasyon stratejisi geliştirilmiştir. Çalışma öncelikle arazi derinliği analizi, arazi görüntüsü dikişi, yol planlaması ve robot sisteminin hareket kontrolüne odaklanmaktadır. Tez kapsamında yol planlaması ve görüntü dikişi için yeni algoritmalar geliştirilmiştir. Arazi derinliği analizinde hem stereo hem de hareket problemlerinden kaynaklanan yapı ele alınmış ve karşılaştırmalı avantajları vurgulanmıştır. Ek olarak, İHA'da kullanılan yerleşik görüş sisteminin performansı, İHA hareket kontrolüne uygunluğunu kontrol etmek için sistem tanımlama teknikleri kullanılarak değerlendirilmiştir. Yeni görüntü dikişi algoritmasının geliştirilmesinde, görüntü verilerine ek olarak eylemsizlik verileri de kullanılmış ve saf görüşe dayalı tekniklere göre daha doğru sonuçlar elde edilmiştir. Geliştirilen yol planlama algoritması, yaygın olarak kullanılan iki yol planlama algoritması olan iyi bilinen A* ve PRM'den daha hızlı ve daha doğru sonuçlar vermiştir. Algoritmaların uygulanmasından elde edilen simülasyon ve deneysel sonuçlar, heterojen robot sistemlerinin işbirliğine dayalı navigasyonu için geliştirilen tekniklerin etkinliğini göstermektedir.

Acknowledgments

I am deeply grateful to my thesis advisor, Professor Dr. Mustafa Unel, for his unwavering support and guidance throughout my Master's studies at Sabancı University. His unique perspective, dedication, and passion for research have been a constant source of inspiration for me. I couldn't have achieved this without his invaluable support and motivation. I have gained a wealth of knowledge and experience from my advisor not only in the subjects he taught but also in various other areas outside the classroom.

I would like to extend my sincerest appreciation to my thesis jurors, Associate Professor Dr. Kemalettin Erbatur and Associate Professor Dr. Huseyin Uvet, for their invaluable feedback and for taking the time to review my M.Sc. thesis. Also special thanks and regards to Associate Professor Kemalettin Erbatur for his early guidance at the beginning of my Master's journey, particularly for the valuable insights and skills he imparted on me in the Autonomous Mobile Robotics course.

I want to send my special thanks to Assist. Prof. Dr. Melih Turkseven because of his guidance, trust, and patience throughout my assistantship journey. I am grateful for his insights into the challenges and rewards of being an academic. His mentorship has been invaluable.

Special thanks to Ilker Sevgen, Zafer Comlekci, Onur Cirakoglu, Yavuz Toksoz and intern Berksan because of their incredible help, patience and labour during my experiments. Without their hands behind me, I would encounter excessive delays in the completion of my preparations.

I send my sincere thanks to my colleagues in Control, Vision, and Robotics (CVR) group and Lab 1093 members. I never imagined that I would have such a research group during my master's journey. My valuable friends made my journey easier thanks to their friendships. All these names will remain my friends for the rest of my life.

Finally, I would like to thank my extended family but especially my dear parents, and grandpa for all their love and endless support throughout my life.

To my family, who have always believed in me and supported me in my academic pursuits, may this thesis serve as inspiration for the next generation to chase their own dreams and achieve greatness...

Contents

List of Figures

Chapter 1

Introduction

The level of autonomy and cooperation of robots is advancing daily. The communication capabilities of homogeneous and heterogeneous robots make the majority of missions simpler and more precise. To comprehend the relationships between these processes, it is necessary to have prior knowledge of this topic's fundamental steps.

For autonomous missions, the robot must first know where it is, where it needs to go, and how to get there. The responses to these questions are separated based on whether the environment was previously known (i.e., mapped case) or unknown (i.e., unmapped case). After this separation, the robot perceives its environment (perception), localizes itself within this environment (localization), plans its path to avoid dynamic and static obstacles, and then executes the movement. All of these actions are referred to as navigation. Therefore, navigation includes algorithms for perception and motion estimation, as well as path planning and optimization, to connect the start point to the goal point by having a model of the environment, perceiving and analyzing the environment to determine its position/situation within the environment, and planning and executing the movement.

Localization is frequently performed using GPS, but because GPS has an error margin of up to 3 meters, which affects the sensitivity of autonomous vehicles, it is preferable to use optical and sonar sensors in addition to GPS. After localization, the robot's perception of its environment becomes an important issue. In environmental perception, subjects and methods are again subdivided based on whether the environment's objects are static or dynamic. In the case of dynamic obstacles, robots can either wait until the dynamic object leaves the camera's field of view, which is the simpler option or select a different route by analyzing the past and present movements of the dynamic obstacle and estimating its future location, which is more difficult. Especially considering that this second case is significantly more demanding in terms of processing load and that it is assumed that the dynamic objects used in the case are constantly moving at a constant speed and orbit, it produces results that fall far short of the desired performance goals. When it comes to path planning, the results depend directly on whether the environment is known beforehand or not, and if it is known, what type of map is utilized (topographic, 3D, or 2D). A trajectory is planned using probabilistic or deterministic algorithms such as RRT*, and A*, after defining the desired point and detecting obstacles on the surface of maps based on prior knowledge. Especially in 2D maps where the depth is unknown, where segmentation of the map's structures is performed, the path drawn for the robot may not be suitable. Consequently, Simultaneous Localization and Mapping steps, which are frequently used in unmapped cases to reconstruct 3D maps, are increasingly preferred, especially in the past decade. All of these processes are fundamental to the autonomous navigation of mobile robots. For instance, detailed, high-resolution maps that were previously known cannot be used in large-scale outdoor applications because they consume a great deal of memory and may become inoperable as a result of environmental changes such as natural disasters.

Against all of these fundamental steps, since the majority of these conditions are not provided in the majority of real-world applications, the robots' capabilities become excessively constrained or their performance suffers. In addition, robots that utilize GPS for localization may become impractical in GPS-denied environments, such as subterranean obstacles, or in cases where the signal is weak. Recent studies have emphasized the importance of increasing productivity by communicating and cooperating with more than one robot (robot-robot communication) in order to complete tasks and expand capabilities. In autonomous navigation and data collection steps, flying robots (UAV) (commonly referred to as flying eye - bird eye) establish heterogeneous communications with the ground or water vehicles.

In this thesis, it is examined a case intended to deploy an unmanned aerial vehicle (UAV) and ground vehicles. The primary objective is to send the ground vehicle to the desired location in various environments while processing the map images from the air vehicle. The fact that the map is be obtained mere seconds before the mission permits map planning and segmentation on the current map. This step differs from previously known maps in this regard. Also, since only the map of the desired area is stored in memory, processing time and system performance are anticipated to increase. In addition, since the camera that views the map is not fixed, it is utilized image stitching techniques by capturing multiple overlapping images in order to extend the map as needed.

On the other hand, as stated in the preceding paragraph, segmentation and depth information are also crucial for analyzing uncharted terrain. This situation is most problematic on 2D maps. Therefore, 3D topographic maps are preferred in the majority of recent studies. There are numerous ways to create a 3D map of the environment. However, the Lidar sensor is predominantly used in SLAM methods and in situations where it is impossible to take images from above, such as underground. In recent years, various depth recognition techniques and image-based 3D modeling have also gained popularity for reconstructing maps.

The developed system consists of various subsystems, each of which has its own methods for achieving results. When it is examined the current state of this concept, it is observed that there are numerous studies on UAV and UGV collaboration, but these studies do not completely cover the subsystems mentioned above in terms of the ideas and methods breaking apart. However, each subsystem can be viewed as separate and distinct topic. Therefore, it is separated the articles according to the sub-systems, such as self-localization and mapping, SLAM, stitching techniques, and path planning in the state of the art to cover the most recent advancements in the related fields.

Last but not least, this type of hybrid system appears suitable for use in areas such as SaR missions, civilian delivery, and hazard site autonomy projects including vehicle management on construction sites. Nevertheless, if it is considered the DARPA's projects beyond 2020 and the work done in the space industry, it is possible to conclude that such hybrid autonomous systems will be more valuable to the defense and space industries in the next decade. Particularly in the space industry, since it is not possible to determine depth from satellites, nor is it possible to control a robot from the earth due to radio wave delays of 5 to 20 minutes along the entire path (i.e., impulse signal transfer from World to Mars takes approximately 3.5 minutes, an image transfer from Mars to Earth takes at around 20 minutes).

1.1 Motivation

Sending a UGV autonomously from a random location to the desired location in unmapped, rough, and GPS-denied environments by selecting the optimal path without going through a pit and avoiding both static and dynamic obstacles is an attractive and promising research problem. Utilizing a UAV to solve this issue offers numerous benefits, particularly in terms of mapping the environment. With the cameras they carry, UAVs are very useful for analyzing structures that cannot be seen from the perspective of a UGV camera. In addition, the depth reconstruction, image segmentation, and stitching techniques employed in UAV-captured terrain images increase the likelihood of constructing a safer path in expansive working areas. Each of these steps necessitates further research. According to the literature review conducted, no study combining these three research topics to solve the vision-based navigation problem for robot systems could be found. This thesis was motivated primarily by the potential innovations and promising novel approaches that the use of these techniques together will bring to the topic of heterogeneous robot collaboration.

1.2 Problem Formulation

Path planning is one of the most crucial issues in mobile robot applications. Despite the fact that many algorithms have been developed to solve the problem with high efficiency over the past few decades, computer vision, machine learning, and deep learning techniques have allowed the algorithms to reach a higher level. However, the topic is still open to research because even the best-known probabilistic and deterministic algorithms do not produce exact results under all circumstances.

When examining autonomous applications, Dijkstra and A* from deterministic methods, and PRM and RRT* from probabilistic methods, have been the most popular algorithms since the beginning of the 21st century. However, even these algorithms cannot guarantee that they provide the correct route when the terrain's depth is unknown. Since these algorithms operate on the occupancy grid of the map, which is the binary representation of the terrain image, depth is ignored. Due to insufficient terrain analysis, robots cannot eliminate the risk of colliding with hills or falling into a chasm that is not visible on a single terrain image. Therefore, particularly in the last decade, it has developed several image-based path planning algorithms that use topographic maps (2.5D maps) to solve the unknown depth problem, or the depth reconstruction techniques have begun to be used to determine the next surface step of the robot.

The technique of stereo imaginary depth reconstruction, which is one of the primary techniques followed by the developed path planning algorithm, should not be used frequently for global path planning. Because the technique identifies which obstacle in the image is closest to the camera, which is typically used for local path planning to avoid dynamic and static obstacles but does not provide a global path. Furthermore, it is not possible to use it alone because there are more robust algorithms and systems that perform this task more effectively.

On the other hand, since the proposed path planning algorithm identifies the path on an image, image-stitching techniques are required to expand the scanned area. Despite the fact that several feature-based and direct image stitching techniques exist in the literature, the techniques require large overlapping regions and a significant number of pixel similarities between image pairs, and their outputs are subject to certain constraints.

In addition, vision-based UAV control, which is a part of the thesis, is an attractive problem because the UAV has a fairly complex structure to control, and there are numerous linear and nonlinear control approaches in the literature to solve this problem. Managing the position control of a UAV using only an onboard sensing system and no external motion tracker remains a significant challenge.

This thesis aims to drive a UGV over the shortest distance possible in an unexplored rugged terrain with the assistance of a UAV by avoiding topographical undulations and static and dynamic obstacles, as well as developing novel techniques for the aforementioned problems. To achieve this objective, it is necessary to review the literature on UAV-UGV collaborative systems, vision-based navigation in the field, and UAV and UGV control techniques and analysis applicable to the creation of the map and its interpretation. This dissertation develops and presents novel techniques, particularly path planning and image stitching, as a result of the conducted experiments.

1.3 Contributions

The thesis's contributions can be summarized as follows:

• A novel image-based path planning algorithm is proposed in which the path between automatically detected initial and desired points is estimated using surface depth and terrain image features.

• The stereo depth reconstruction is used in the proposed path planning algorithm to compute a global path that avoids negative and positive obstacles.

• Performances of Convolutional Neural Networks and different hand-crafted feature-based approaches for stereo depth reconstruction techniques are compared using machine learning classifiers.

• Comparative analyses of the performance of stereo depth reconstruction and structure from motion methods for terrain depth reconstruction are performed.

• Using an onboard vision system, quadrotor parameters are estimated, and it is demonstrated that the onboard vision system can be used for UAV motion control.

• A novel technique for image stitching utilizing UAV-IMU data has been developed.

1.4 Outline

The structure of the thesis is as follows:

In Section 2, the literature review conducted on UAV-UGV collaborative systems is presented. In addition to UGV classification and modeling, this section discusses linear and nonlinear UAV control strategies. In addition to image stitching techniques, the state of the art is also reported for mapping methods.

In Section 3, titled "Terrain Analysis and Topographic Map Building for Heterogeneous Robot Systems," the effects of various machine learning classifiers and

CNN method on stereo depth reconstruction technique are discussed, and structure from motion technique is explained after giving explicit camera calibration steps. In addition, the proposed algorithm for image stitching is described at the end of this section.

In section 4, the survey and experiments conducted regarding image-based motion and path planning are explained. The proposed algorithm for image-based path planning is explained here. Moreover, system identification experiments of a quadrotor using an onboard vision system are also conducted under this title. The significance and performance of vision-based systems for UAV motion control are then demonstrated.

The experimental setups and actual and simulated experiment environments are described in section 5. The section concludes with a discussion of the obtained image-based path planning results.

The thesis concludes with some concluding remarks and possible future directions in section 6.

Chapter 2

Literature Review

2.1 Heterogeneous Robot Collaboration

Multi-robot systems improve not only the system's capacity to solve the target problem in less time and with less energy waste but also its adaptability and resiliency when performing complex tasks. Multi-robot collaborative systems may consist of robots of the same type, referred to as homogeneous robot collaboration, or robots of different types, such as unmanned aerial vehicles (UAV), unmanned ground vehicles (UGV), and unmanned underwater vehicles (UUV), referred to as heterogeneous robot systems. These systems are classified as centralized or decentralized according to their coordination strategies. In centralized systems, a leader robot or observer controls a group of robots, which act as a unit. In decentralized systems, robots of the same rank share the sub-branches of a fundamental task, and each robot strives to complete its assigned task autonomously. Using onboard sensors or global tracking systems, each robot avoids collisions as it progresses through the process.

In the literature, numerous heterogeneous and homogeneous multi-robot navigation systems are described. When solving a problem, these systems typically follow the steps of task decomposition, formation distribution, perception and control, and navigation. Since the very first day mobile robots were discussed, the navigation issue has remained of the utmost significance. Due to the development of intelligent control and perception methods, unmanned aerial vehicles have been utilized more frequently and successfully in mapping and navigation issues over the past several decades.

2.1.1 UAV-UGV Collaboration

Literature reviews demonstrate that UAV-UGV collaborative systems are capable of performing a variety of missions, including intelligent surveillance and reconnaissance (ISR), autonomous localization/mapping, and navigation missions. During these missions, various UAVs, such as fixed wing, rotary wing, and hybrid UAVs, and UGVs, such as differential drive, holonomic, nonholonomic, steering, or tank drive ground vehicles, may be utilized. The type of UAV has a significant impact on the performance of mapping and aerial surveillance when, depending on the surface structure, the type of UGV affects the success rate and efficiency. For the experimental setups of the thesis, a skid-steering wheeled UGV and a rotary-wing quadrotor are chosen.

Fixed Wing	Rotary Wing	VTOL (Hybrid)
Needs large zones to take-off and landing	High accelerated maneuverability	Optimum energy consumption
Velocity (v)	$v_{fix} > v_{VTOL} > v_{rotary}$	
Flight Distance (d)	$d_{fix} > d_{VTOL} > d_{rotary}$	
Flight Altitude (h)	$h_{fix} > h_{VTOL} > h_{rotary}$	
Payload Capacity (p)	$p_{fix} > p_{VTOL} > p_{rotary}$	

FIGURE 2.1: UAV Types and Specifications

In the paper [1], the UAV used as part of a cooperative system serves as both a flying sensor and a tether attachment device. A tether connects two robots, allowing the UAV to anchor the tether to a structure located on a steep terrain that is inaccessible to UGVs. Thus, improving the UGV's poor traversability by not only providing a broader range of scanning and mapping from the air, but also by allowing the UGV to climb steep terrains via tether winding. Moreover, the authors present a framework for autonomous collaborative navigation and tether attachment in an unknown environment. The UAV uses visual-inertial navigation with 3D voxel mapping and planning for obstacle avoidance where the voxel mapping means the division of the surface into small cubes, with each cube taking on a different color based on its height. Furthermore, this article compares the pros and cons of possible methods for tether anchoring from multiple perspectives. In addition, the entire system consists only of UGVs and UAVs; there is no external central computer. The UGV operates ROS, which is linked to the UAV client. For localization, both robots use dead reckoning based on odometry and IMU data. The UAV's onboard computer receives voxel data from the lidar sensor (or a similar sensor such as Time-of-Flight (ToF) sensor) and converts it into a grid map. Using these grids and depth data, the path planning algorithm then determines the optimal route for detecting obstacles and cliffs by locating the optimal path. As is common knowledge, the aerial vehicle has a mission other than observing the map and locating a suitable obstacle to attach the hook. In order to complete the specific mission, UAV navigation is of utmost importance. Nevertheless, the detection algorithm also employs simple steps. Initially, the UAV takes off and hovers at a height of one meter before using a ToF sensor and camera to detect surfaces and depths. After detecting the environment, it chooses an obstacle that remains on a higher surface and circles around it. After completing the trajectory, the system searches for a suitable landing area using grids generated during image processing. Following these trajectory and detection processes, only the onboard tow is active, and the UGV is climbing the cliff. In order to increase the likelihood of successful anchoring, an experiment is conducted to evaluate the anchoring strategy. An autonomous mission experiment in the field with an obstacle and a cliff that is representatively constructed in indoor and more realistic outdoor environments demonstrates the viability and capability of the proposed system.

The papers that discuss collaborative mapping and navigation efforts employ parallel techniques and comparable hardware. Due to the use of a blimp as an aerial viewing robot, the research presented in this paper [2] is significantly distinct from the majority of projects with a similar objective and more suitable for long-term missions. In addition, since the lidar sensor, stereo camera, and wide-angle camera are all utilized concurrently, the collected data and resulting maps are more precise, and errors caused by incorrect depth information during map planning are minimized. The authors discuss the inefficiencies of indoor mapping and navigation techniques at the outset of the paper. They advocate for the possibility of developing a novel system that can address these issues and improve efficiencies. They propose a system that designs a UAV-UGV team consisting of two custom-built mobile robots. The UAV serves as an external eye for the UGV, observing the scene from an inaccessible vantage point. Continuous estimation of the UAV's relative position allows it to maintain a fixed position relative to the UGV. The development of this autonomous navigation system hinges on the localization of both UAVs and UGVs, the mapping of the surrounding environment, and efficient path planning using multiple sensors. Vision techniques provide the relative positioning of the aerial and ground vehicle to handle the lack of GPS. These robots can move in tandem because of a marker placed beneath the blimp and a wide-angle camera mounted on the ground vehicle and pointing perpendicularly upward. In addition, both robots utilize IMU sensors for individual localization, and UAV and UGV communicates with each other via the wireless network and onboard raspberry computers. The central computer on UGV is primarily used to run SLAM and depth recognition algorithms, and USB ports are used to connect the cameras and lidar sensors to the central computer. Ultimately, the entire system utilizes vision and SLAM algorithms in GPS-denied environments to predict the environment, 3D mapping, and point-by-point travel (via the shortest route). The proposed system is evaluated in an indoor environment resembling a cluttered construction site. The system's performance demonstrates the viability of developing and deploying in the near future a robust and automated data collection system for construction applications.

This study [3] applies mathematical algorithms to a typical rescue scenario. In a manner similar to the previous paper, the UAV camera is utilized as a "flying eye," and the UAV provided global information via its vision sensor. To model the environment, the images are then processed with the SURF algorithm and image binarization. On the basis of the UAV's data, the optimized path planning algorithm A* proposes a route. Then a feasible path for UGV rescue is designed. In the physical configuration, a flying and ground robot and a central computer are present. In contrast to the previous article, the UGV and UAV do not act as a single body in this one. As with the majority of heterogeneous robot systems, the UAV utilizes wide-angle environmental detection. The algorithm that is running on the central computer uses a picture of the map views to segment the ground from the obstacles. After segmentation, a second algorithm, the map planning algorithm, runs on the central computer to find the optimal shortest route while avoiding these detected static obstacles. After completing these calculations, the established route is transmitted to the UGV. During these sub-steps, the authors focused on improving map analysis and obtaining higher recognition accuracies through the application of preprocessing techniques. In addition, based on their theories, the SURF algorithm and image segmentation techniques have high robustness in map planning. Moreover, simulation and physical setup experiments are performed as the final step. When comparing the desired system to this paper, it can be concluded that the ideas in this project are so similar to the desired one. However, significant differences remain, such as types of ground vehicles and mapping techniques.

The general structure and usage purpose of robot collaboration in this paper [4] are so similar to paper [1], titled "UAV-UGV Autonomous Cooperation: UAV Assists UGV to Climb a Cliff by Attaching a Tether," with the exception of hardware and control algorithms. At the beginning of the paper, the authors discuss the significance of collaboration between fully autonomous mobile robots on unknown and difficult planet surfaces, such as Mars. The system consists of three different mobile robots named ARDEA, LRU1, and LR2, a solar panel station named Lander, and a number of payload containers. Since the system is presented in

Italy in 2021 as part of the ARCHES project, the missions are specified by the competition rule. According to these rules, each robot has specific tasks, and in order to preserve relative positioning, communication, and mapping skills during these tasks, it is becoming essential to implement methods that conflict with the candidate system's expectations. There are two general missions that must be completed in order to succeed in the competition. Both missions cover automated task planning, high-level mission control, spectral rock analysis, radio-based localization, and collaborative multi-robot 6D SLAM in Moon-analogue (with another propulsion system different than propellers since there exist no atmosphere at Moon) and Mars-like scenarios. According to the scenario, ARDEA is used for autonomous, rapid mapping and exploration. The ground vehicles are segmented and detected thanks to the onboard dual-core Intel computer, which serves as the main computer. LRU1 and LRU2 ground vehicles have individual control algorithms, and unlike previous swarm robot papers, they are not required to move in unison because they have distinct missions. Using onboard stereo cameras, both of the ground vehicles determine the depth of their surroundings. The depth of information also facilitates the creation of 3D models and makes SLAM's job easier. In addition, after candidate map detection and optimal route determination, when necessary, the flying robot ARDEA lands on LRU1. Finally, it is possible to say that the used multi-robot 6D Simultaneous Localization and Mapping (SLAM) allows robots to localize themselves with respect to each other and build a shared dense 3D model of their environment for autonomous exploration, mission planning, and coordination of joint actions and that the structure of the entire system is dependent on these techniques.

Using observations from a flying robot, this paper [5] presents a solution to the problem of planning a path for an unmapped ground robot through unknown terrain. The system structure is explained using a scenario involving search and rescue operations. To avoid time-consuming active exploration of the environment, the flying robot selects regions to map in a manner that optimizes the overall response time of the system, which is the sum of the air and ground robots' mission execution time. In this method, terrain classes are estimated across the terrain map, and elevation data is added to regions where the active exploration algorithm has chosen to perform the 3D reconstruction. This terrain data is used to estimate feasible and efficient ground robot routes. After separating the surface grid by grid, the algorithm for path planning begins to run. The algorithm used for path planning distinguishes this article from its predecessors. As is common knowledge, there are various types of path-planning algorithms, which are distinguished as deterministic and probabilistic. In the majority of missions, probabilistic algorithms such as RRT and RRT* are utilized because they require less memory and require less time to calculate the route. However, D*, a deterministic algorithm, is executed in this article. Following the calculation of the terrain surface structure and region depths, the map is segmented and divided into grids. Using these grids, or cubes in a 3D environment, the algorithm attempts to calculate the shortest route between the current location of the ground vehicle and the target location. At the end of the paper, the proposed techniques are demonstrated in both simulation and outdoor environments, and their performances and capabilities are discussed.

2.2 Vision-based Navigation of UAV-UGV Collaborative Systems in GNSS-denied Environments

The navigation system and its accompanying subsystems comprise the majority of an autonomous UAV. The navigation system estimates the pose of the UAV in terms of its locations (x, y, and z) and orientations (ϕ, θ, ψ) using data gathered from a variety of sensors. Other supporting systems do pertinent functions, such as obstacle detection and static object detection or dynamic object tracking, or obstacle avoidance.

Advanced navigation systems are one of the touchstones of increasing autonomy levels day by day. Monoculars and stereotypes of cameras and computer vision algorithms are the most important equipment for navigational operations, especially in GNSS-denied environments. At this stage, It is important to divide the navigation systems into three major subsystems which are pose estimation, obstacle avoidance, and visual servoing where the obstacle avoidance system detects and relays the location of impediments in the path of the UAV, visual servoing system manages and transmits movement commands to the controller to maintain the stability and follow the route, and finally, pose estimation is responsible to the assessment of the location and attitude of the robot in 2D and 3D representations. In literature, there exist some important studies regarding the subject.

This paper [6] uses a stereo-vision system based on multiple unmanned aerial vehicles to aid in the global path planning of an unmanned ground vehicle in GPS-denied environments. Similar to previous endeavors, aerial vehicles are expanding their field of vision. The unique aspect of this strategy is the use of multiple UAVs, and the proposed technique provides the opportunity to move these vehicles along a baseline. The authors explain how a multi-UAV system can be used to overcome the limitations of a single-UAV system with a fixed baseline. Thus, it is asserted that the performance of the innovative system is not significantly affected by altitude. In addition, estimations of relative position and altitude, multi-agent formation control, and image processing methods are considered for the implementation of a prototype system. Due to the utilization of multiple UAVs, the size of the detected area grows automatically. The flying robots are using computer vision techniques to detect ground vehicles. Since the relative positions and orientations between the UAVs and UGV are determined using a marker on the UGV and the laboratory's motion capture facility, it is now possible to detect depth using stereo-vision techniques, unless the flying robots are at significantly different altitudes. In contrast, for the majority of the experiment, the motion capture system is only used to create a ground map. Stereo depth recognition is performed by aerial vehicles that recognize their relative positions in relation to one another based on the detected marker position of the UGV on their display screen. With the aid of detected depths, the obstacles along the

UGV's trajectory line can be identified. No probabilistic or deterministic path-planning algorithm is used in the experiment. The ground vehicle only follows paths that are free of obstructions. Therefore, the system is not dependent on GPS. The performance of the proposed system is then demonstrated in a controlled laboratory environment with a motion capture facility. The experimental results illustrate the performance of the implemented system for a variety of UAV baseline conditions.

In this paper[7], the authors present a comprehensive design and implementation for a micro aerial vehicle (MAV) that is capable of 3D autonomous navigation and obstacle avoidance in cluttered and realistic unknown environments without GPS and other external sensors or markers. To accomplish these autonomous missions, modularized components, such as visual-inertial odometry (VIO), 3D occupancy mapping, and motion planning, are developed for the MAV. The proposed system is designed to operate in real time on a small embedded computer. Simulation and actual flight experiments demonstrate its durability. In this method, the aerial vehicle is equipped with a stereo camera, and the UAV utilizes stereo-camera-based state estimation methods. Nonetheless, an inertial sensor is added to this visual method to increase its applicability and robustness, as the state may deviate significantly during aggressive maneuvers if only visual information is used. In addition, the UAV's powerful onboard computer processor, Nvidia Jetson TX2, processes collected data without sending it to a centralized computer on the ground, and the system continues on its route based on the analysis of movable areas. The robot creates an occupancy grid map using a laser and stereo camera before moving on to the following suitable regions by following SLAM procedures. On the other hand, this paper contains a number of significant contributions that distinguish it from other techniques of a similar nature. For instance, the state estimation based on visual-inertial sensor fusion is designed to function with off-board camera-integrated visual odometry data, thereby reducing computational complexity. It is demonstrated that a lightweight velocity-based fault-detection method can withstand difficult environments with sparse features. The paper also discusses the implementation of the proposed method on a real quadrotor capable of navigating complex 3D environments and tracking moving references. The paper concludes with a discussion of a micro aerial vehicle system that can autonomously navigate in GPS-deficient and obstacle-filled environments while avoiding obstacles. In order to achieve this, vision-based localization techniques are employed, and the proposed system is evaluated using both simulation and actual flight data.

This paper [8] presents a method for 6D localization of mobile robots in the absence of GPS using a 3D laser scanner. Commonly, 6D localization using laser scanners involves the extraction and association of features or the comparison of whole scans (frequently offline) utilizing the ICP algorithm or its variants. In some unstructured, non-urbanized, rugged terrain environments, however, feature extraction does not appear to be sufficiently reliable. PSD is a method presented by the authors for mobile robot localization in GPS-denied applications in such environments (Point-to-Surfel Distance). They consider each laser scanner measurement as an observation and use PSD to correct the position and orientation of the robot, as opposed to the laser scanner-based localization methods of previous articles. The robot's localization is based on a specific terrain representation in a 2.5-dimensional surfel map. There is no collaborative work in this system, so the ground vehicle travels independently around the map using data from the lidar and onboard cameras. The proposed system is evaluated in both indoor and outdoor environments, and Gaussian mixture maps are used to compare the extended Kalman filter and single laser scanner measurement techniques. The results are then presented at the conclusion of the paper.

2.2.1 Self Localization and Mapping

Self-localization is the process of determining the location of a robot within its environment based on sensory inputs, and it is essential to compensate for position errors caused by external noises. In addition to reducing the effectiveness of subsequent trajectory and mapping operations, positional errors can put the robot in danger.

Metric and topological approaches are the two main sections that distinguish mapping methods from one another. The occupancy/certainty grid was the first to provide the most prevalent metric method. Each cell on the occupancy grid represents an area of the immediate surroundings. Each cell in the grid is assigned a certainty value that indicates the probability that it contains an obstacle. Topological networks represent the environment in the topological approach. This is achieved by identifying and connecting the environment's distinct locations and pathways.

These processes are the most fundamental for a robot to navigate itself autonomously. In studies conducted over the past few years, it has been demonstrated that these processes can be carried out concurrently using advanced sensing systems and algorithms. The following published studies are excellent examples of the processes:

This study's [9] objectives are to operate aerial and ground robots cooperatively and to fuse complementary point cloud data. In addition, this paper presents a framework for autonomous cooperation between UGVs and UAVs for the collection of 3D geometric data in a dynamic, cluttered environment. First, an unmanned aerial vehicle (UAV) is deployed and 3D terrain data is collected using images to gain a general understanding of the target location. Using the gradient-based 3D model generated by the UAV, the UGV's path planning and stationary scanning locations are then estimated. The working structure of the entire system is comprised of various steps. Initially, the UAV is used to generate a rough 3D map of the construction site. Using the captured image, the mapping algorithm generates a point cloud that divides the job site into cells (1m by 1m) and computes the gradient between each cell's neighbors. Then the 2D gradient map and occupancy map are generated to determine the site's optimal movable area. Then the entire candidate scan location cells are analyzed by simulating the line of sight and counting points in areas to determine their coverage size. Sending these locations to a ground vehicle is the subsequent step in simulating optimal areas. The ground vehicle has four 2D line laser scanners and a standard camera for collecting 3D map data. One of the horizontally mounted 2D laser scanners estimates the robot's location and pose in a 2D plane, while a regular digital camera captures the RGB data of the scene. The ground vehicle performs the 3D mapping operation while collecting point cloud data. The UGV then continues to move in accordance with the gradient of the potential field and converges on the subsequent scan location. When the UGV reaches the final location to be scanned, it begins to register each point cloud to a single coordinate system using localization data. The horizontal lidar is used to calculate the localization data. The SLAM algorithm uses the laser-scan data from the horizontal lidar to estimate the position and orientation of the ground vehicle GRoMI in the horizontal plane. This study employs the Hector SLAM algorithm to perform laser-scan matching between current lidar scans and progressively constructed maps in order to estimate robot postures and planar maps of the environment. Finally, the project's physical experiment is conducted on the Georgia Tech campus, and encouraging results are obtained at the conclusion of the construction site experiment.

This article [10] demonstrates a direct application of point clouds for UGV path planning. It enables skipping the usual discretization of space and utilizing all available data. This paper presents algorithms for point cloud-based terrain evaluation and global path planning using rapidly exploring random trees (RRT). Experiments utilize publicly available datasets of UAV-generated point cloud maps. The article provides a quantitative evaluation of the performance of planning on a point cloud map on a central processing unit, as well as practical implementation details. Similar to the previous paper, the movable areas are analyzed by first creating a map that resembles a 2.5D map (elevation map that defines height using different colors). After creating a point cloud map of the terrain with a UAV, the algorithm calculates the optimal route using RRT. The ground vehicle then follows the proposed route. The methods presented in the paper are also implemented in a physical environment. Except for the path planning algorithm and a few hardware components, the concept and working principles are identical to the previous article.

2.2.2 SLAM in UAV-UGV Collaboration

SLAM is an important process for providing a trajectory in an unknown environment by utilizing various types of depth recognition and localization sensors, as well as mapping the environment. The majority of studies are incorporating SLAM algorithms into the controller of ground vehicles in order to construct 3D maps and better analyze the environment. The proposed method, it is first extracted the map of an unknown environment and then sends ground vehicles to the desired location using the optimal path planning algorithm. Therefore, the SLAM may not appear to be so important; however, being aware of recent advancements in this area may help to generate new ideas and enhance the system. The abstracts of the following articles describe the most recent developments in the study.

Cooperative robots must perform collaborative semantic mapping to maintain a comprehensive contextual understanding of their surroundings. The majority of current research focuses on either single-robot semantic mapping or collaborative geometry mapping. In this paper, [11], a novel hierarchical collaborative probabilistic semantic mapping framework with a distributed formulation of the problem is proposed. The primary contribution of this work is the mathematical modeling and derivation of the probability decomposition of the overall collaborative semantic mapping problem. At the level of a single robot, a semantic point cloud is generated using a heterogeneous sensor fusion model, and a semantic point cloud is obtained at the level of a single robot. Since the voxel correspondence is unknown at the level of collaborative robots, an Expectation-Maximization approach is proposed to estimate the hidden data association, with the Bayesian rule applied to update semantic and occupancy probability. The experimental results demonstrate the high quality of the global semantic map, demonstrating the precision and utility of the 3D semantic map fusion algorithm for actual missions.

The problem of real-time robot exploration and map building (active SLAM) is examined in this article [12]. A fully autonomous robot uses a single stereo-vision camera to navigate, localize itself, define its surroundings, and avoid any potential obstacles in order to maximize the mapped region by following the optimal route. A modified version of the so-called cognitive-based adaptive optimization algorithm is presented so that the robot can successfully complete its tasks in real-time and avoid becoming trapped in a local minimum. Using properly equipped robots, the method's efficacy and performance were evaluated in a variety of simulation environments as well as in real, unexplored areas.

This article [13] is one of the most representative studies for fully autonomous long-range exploration robots. In addition, one of the earliest efforts to enable large-scale and long-term autonomy with the Boston Dynamics Spot robot is provided. Motivated by the exploration of extreme environments, specifically those involved in the DARPA Subterranean Challenge, this paper pushes the state of the art in enabling legged robotic systems to complete complex real-world missions in relevant scenarios. The authors discuss the behaviors and capabilities that result from the integration of NeBula (Networked Belief-aware Perceptual Autonomy) with next-generation mobility systems, as well as the hardware and software challenges and solutions in mobility, perception, autonomy, and, to a lesser extent, wireless networking, along with lessons learned and future directions. Then they demonstrate the effectiveness of the proposed solutions on actual physical systems. This research endeavor is revolutionary. Even though the used equipment is similar to that of previous SLAM projects, when the system is combined with the capabilities of Boston Dynamics' Spot robot, one of the most accurate results is achieved. In the study, the user can remotely interfere with the system if desired. Using visual data, the localization and orientation processes are carried out. Due to the lidar sensor, a 3D model of the environment is gathered in real-time, and the system detects the environment and applies segmentation. The robot then moves to the next available location. And the entirety of these processes are performed as sub-steps of the SLAM algorithm. Moreover, the proposed solution contributed to the first-place finish in the 2020 DARPA Subterranean Challenge, Urban Circuit.

2.2.3 Path Planning

Path planning is one of the most crucial challenges for autonomous mobile robots. Following perception, robots locate themselves based on the obstacles they can see in their environment. During these steps, local path planning is performed to prevent collisions with dynamic and static obstacles. After locating themselves based on environmental objects, the global path planning algorithm is activated and the robot attempts to reach the desired location. This cycle can be observed in the majority of mobile robot applications, but especially in those that use map-based localization techniques such as Markov Localization, where the map is known beforehand, or SLAM, where the robot has no prior knowledge of its environment and global map.

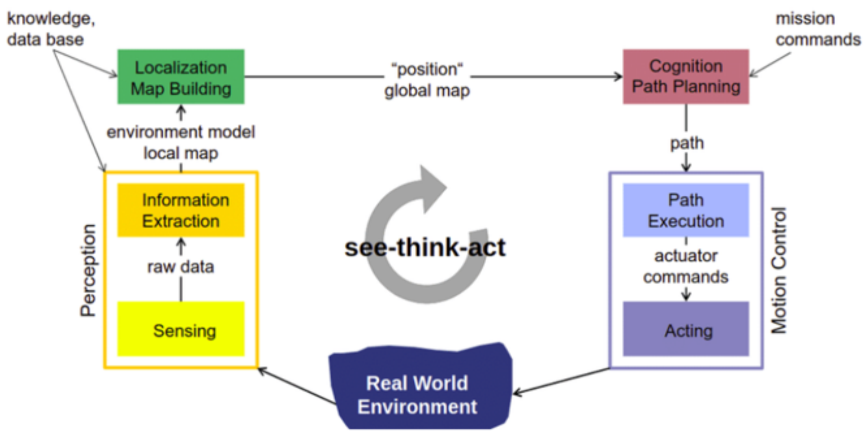


Figure 2.2: See-Think-Act Cycle

On the other hand, since the path planning problem is based on optimization problems, which have been a popular area of research since the middle of the 1960s, many deterministic and probabilistic algorithms have been developed, but only a few have reached an efficiency level sufficient for application in robotic systems. Deterministic algorithms (Hart, Nilsson, Rafael, 1968; Nilsson, 1980) and probabilistic algorithms (Kavraki et al., 1996; LaValle, 1998; LaValle & Kuffner, 1999; 2001) are the two most common techniques [14]. However, prior to 2001, all research focused solely on locating the shortest path between two points on a 2-dimensional image without elaborating on how the map is generated. In 2001, the term ”image-based path planning algorithm” was added to the lexicon with the publication of a paper in Japan [15]. The paper focuses on decreasing the sum of moving distances of a manipulator’s degrees-of-freedom, and the most efficient path planning algorithm of the time, A*, was used to find the shortest path between two points detected by a camera located on the head of the end-effector. Although the application was based on a fairly simple concept, namely dividing the working terrain into grids and locating the shortest path using the grid lines, it is regarded as a cornerstone of terrain analysis-based path planning algorithms that are still in use today.

Numerous path-planning algorithms exist in the literature, and these algorithms can be divided into probabilistic and deterministic techniques, as described in earlier sections. Although they are completely different in terms of computational power, processing time, and fundamental logic, they use the same or similar image characteristics as references. Literature classifies algorithms as grid-based search, interval-based search, artificial potential field-based, and sampling-based. However, the majority of image-based algorithms are founded on geometric principles. Visibility Roadmap, for instance, is a graph of intervisible locations, typically for a set of points and obstacles in the Euclidean plane. Each image corner represents a point location, and each image edge represents a visible connection between them.

The Voronoi diagram is another node merging technique, but it differs from the Visibility Roadmap technique in that it uses the detected feature point offsets to establish the connection between nodes. It is a mathematical partition of a plane into regions close to each object in a given set.

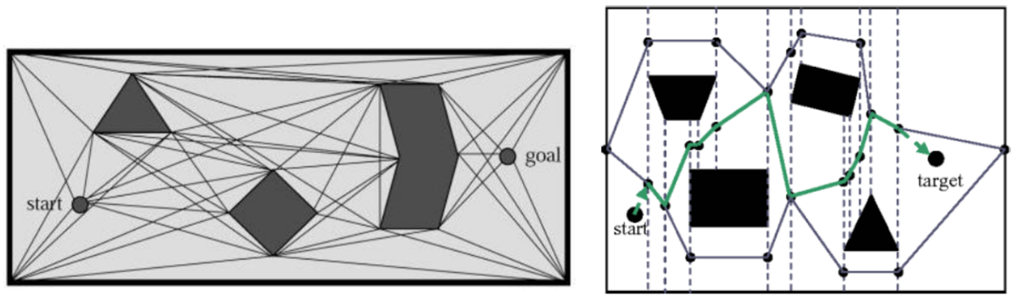


Figure 2.3: Visibility Roadmap (Left) and Voronoi Diagram (Right)

Trapezoidal Decomposition is yet another path-finding technique. Due to the inability to conduct a binary search, it is difficult to discern how to use a trapezoidal decomposition for point location in this method. It can be obtained, however, by firing vertical bullets upwards and downwards from each vertex of the original subdivision. When bullets hit a boundary, they stop and create a new boundary in the subdivision.

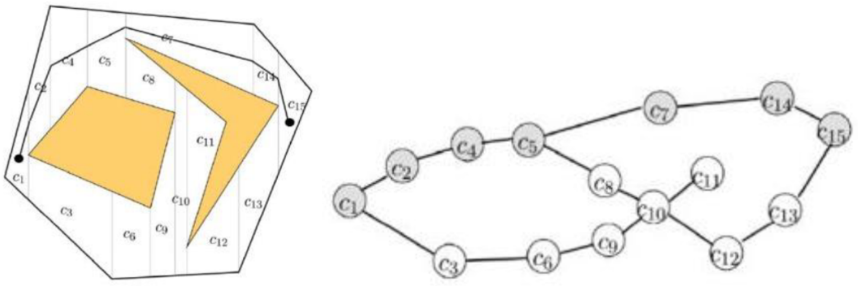


FIGURE 2.4: Trapezoidal Decomposition

There are numerous additional techniques, including triangulation refinement, monotone subdivisions, slab decompositions, and cell decompositions, among others. However, these three methods form the basis of the vast majority of probabilistic and deterministic algorithms.

In the experimental setups established for the research, deterministic and probabilistic algorithms are used to find paths, and the comparative analyses of these algorithms and the proposed path planning algorithm are presented.

In the experimental setups established for conducted research, the deterministic and probabilistic algorithms are employed to find paths and it is presented the results of comparative analyses of these algorithms and the proposed path planning algorithm.

2.2.3.1 Deterministic path planning Algorithms

Deterministic path planning algorithms are those that evaluate nearly all possible routes to determine the shortest route between two nodes. Using the nodes or grids on the occupancy grid, provide optimal paths, but their computations are too time-consuming. Numerous deterministic path-planning algorithm examples exist. A* Dijkstra and the Visibility graph are the most popular. In the experiments conducted within the scope of the thesis, the A* algorithm is chosen as a representative of deterministic algorithms to compute the path in test environments because it produces the most optimal results compared to others.

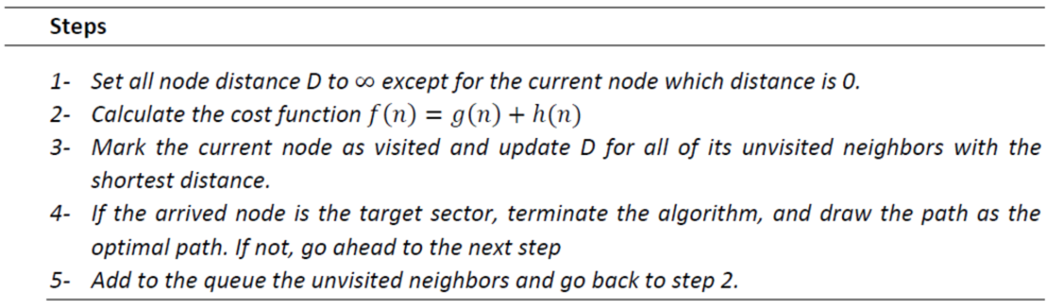
a) A*

The A-star path-finding algorithm is arguably the best algorithm for finding the shortest path between two nodes. In addition, it is a heuristic function-based deterministic algorithm for proper path planning that calculates the value of the heuristic function at each node in the work area. Checking too many adjacent nodes is required to find the optimal solution with zero collision probability. Thus, it requires a great deal of processing time and slows down the rate of work.

In addition, it is essential to note that A* chooses the path that minimizes the defined cost function, such as;

$$f(n) = g(n) + h(n)$$

where the $g(n)$ is the cost of the path from the initial point to node n), and $h(n)$ is the estimated cost of the path from node n to the destination node.



Steps

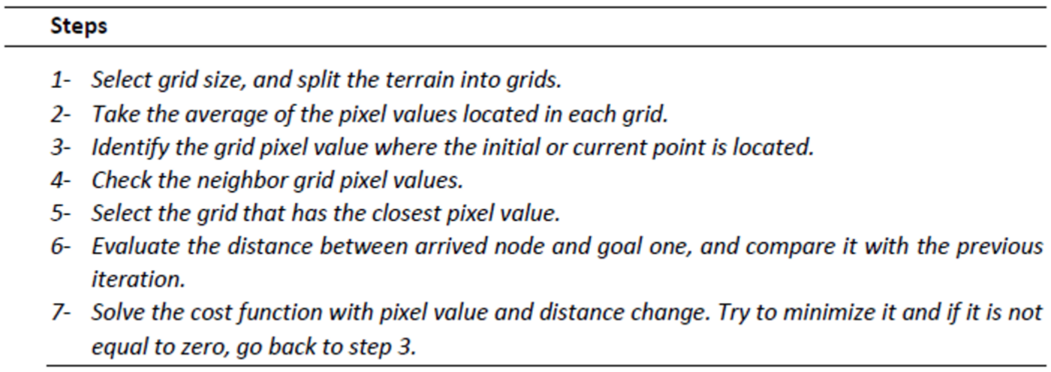
1- Set all node distance D to ∞ except for the current node which distance is 0.
2- Calculate the cost function $f(n) = g(n) + h(n)$
3- Mark the current node as visited and update D for all of its unvisited neighbors with the shortest distance.
4- If the arrived node is the target sector, terminate the algorithm, and draw the path as the optimal path. If not, go ahead to the next step
5- Add to the queue the unvisited neighbors and go back to step 2.

FIGURE 2.5: A* Steps

So, the A* star algorithm is one of the most popular deterministic algorithms developed by Dijkstra, and it operates on the occupancy grid form of the map without utilizing the image's object characteristics. Contrary to Dijkstra, A* does not scan the entire image. Instead, it assigns nodes and the distance between them to the desired point direction based on the selected cost function, and it attempts to find the shortest path between the initial and desired locations by minimizing the cost function.

b) A Dynamic Programming Method on Topographic Map

Path planning based on topographic maps is also known as Traversability analysis and has been an active research field since the turn of the 21st century. In this method, the terrain is divided into grids, and the algorithm connects grids with close pixel values along the path between the initial and desired locations.



Steps

1- *Select grid size, and split the terrain into grids.*
2- *Take the average of the pixel values located in each grid.*
3- *Identify the grid pixel value where the initial or current point is located.*
4- *Check the neighbor grid pixel values.*
5- *Select the grid that has the closest pixel value.*
6- *Evaluate the distance between arrived node and goal one, and compare it with the previous iteration.*
7- *Solve the cost function with pixel value and distance change. Try to minimize it and if it is not equal to zero, go back to step 3.*

FIGURE 2.6: Dynamic Programming Method Steps

Instead of assigned nodes, the dynamic programming method operates on the basis of the average pixel values in each grid. It attempts to choose the grids along the path to the destination that has the closest value to the current grid value. In addition, this algorithm is designed to find the optimal route in a rural setting by selecting the slope with the least gradient.

2.2.3.2 Probabilistic Path Planning Algorithms

The working principles of probabilistic path planning algorithms rely on the random placement of waypoints on a map. This method's most important hyper-parameter is the number of thrown nodes. The path's shape and computation time vary based on the hyperparameter that the user specifies. Except for RRT*, these algorithms do not guarantee to find the optimal path. Due to its structural distinction, only RRT* can locate the shortest path between two nodes. Despite this, RRT* performance may not be adequate for determining the optimal path when the number of candidate waypoints is too small. Probabilistic path planning algorithms include Probabilistic Roadmap (PRM), Rapid-exploring Random Tree (RRT), and RRT*, among others. Due to its quick response and simpler structure, the probabilistic roadmap algorithm is selected for use in the conducted experiments.

a) Probabilistic Roadmap Algorithm (PRM)

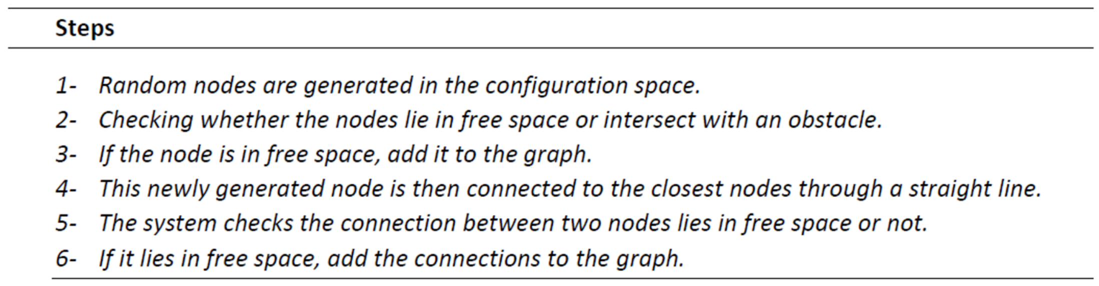
The probabilistic roadmap is an algorithm that constructs a graph from the initial position to the target position in order to place a randomly generated point within a specified map region. It generated a limited number of random points within a specified area, and the path is drawn by connecting the points. There are also numerous techniques for generating random points and connecting them. PRM can be characterized as:

$$R_{PRM} = \gamma_{PRM} \frac{log(n)}{n}^{1/d} \tag{2.1}$$

$$\gamma_{PRM} > \frac{\mu(x_{free})}{\zeta_d}^{1/d} 2(1+\frac{1}{d})^{1/d} \tag{2.2}$$

where R is the radius; n is the number of nodes; d is the number of dimensions; γ is a constant according to the selected cost function; μ is the total area; and finally, the ζ is the volume of a unit sphere in the dimension being used.

The PRM is therefore one of the most prevalent probabilistic path-planning algorithms with RRT. In both of these algorithms, the nodes that are used to create trajectory are thrown randomly onto the map, and the algorithms attempt to find the shortest path between the starting point and the destination point by avoiding obstacles. As it is similar to A*, it was not necessary to apply computer vision



Steps
1- Random nodes are generated in the configuration space.
2- Checking whether the nodes lie in free space or intersect with an obstacle.
3- If the node is in free space, add it to the graph.
4- This newly generated node is then connected to the closest nodes through a straight line.
5- The system checks the connection between two nodes lies in free space or not.
6- If it lies in free space, add the connections to the graph.

FIGURE 2.7: PRM Steps

algorithms to the designed map in order to implement it, as these algorithms work with the grid-based form of the map. In this form of the map, the detected black regions are treated as obstacles, and the path-finding algorithms do not place any nodes on top of these black regions. Consequently, the drawn path has no collisions with obstacles.

2.3 Modeling and Control Approaches for UAV-UGV Collaborative Systems

In UAV-UGV collaborative systems, air and ground vehicles continuously execute their own autonomous control algorithms as they collaborate to solve a problem. Because of this, separate controllers have been developed and implemented for both ground and air vehicles.

In the study conducted for this thesis, an aircraft flies while avoiding dynamic and static obstacles in an unknown environment, and after capturing surface images with a stereo camera, a topographic map of the area containing the starting and ending points is generated. On this map, the global path for the ground vehicle is then created. While the ground vehicle follows this global path created by a system with a broader field of view, the stereo camera constantly performs depth analysis and dynamic obstacle avoidance operations. As shown in the following block diagram, both ground and air vehicles require separate controllers, as the tests are conducted in environments without a global motion tracking system.

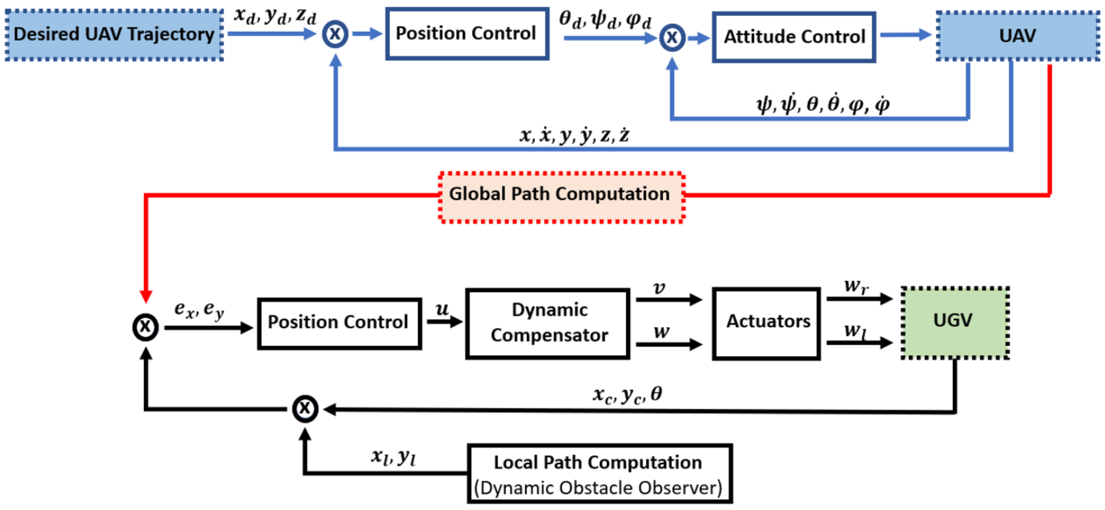


FIGURE 2.8: Block Diagram for UAV-UGV System

2.3.1 Control Approaches for UAV

UAVs can be divided into three primary categories: fixed wing, rotary wing, and hybrid wing as mentioned in Figure 2.1. The rotary wing quadrotors can also be distinguished by the number of their rotors. Within the scope of the experiments conducted for this thesis, quadrotor UAVs were chosen for use. Due to their design and four engines, quadrotors have greater capabilities than fixed-wing aircraft in confined spaces. These four motors allow the robot to move with six degrees of freedom (DoF). These six degrees of freedom include x, y, z, pitch, yaw, and roll motions. The controllers that ensure the stability of these quadrotor movements are referred to as low-level controllers. The modeling of a quadrotor which is explained in section 4.2.1 is the first step of designing a controller.

When designing a low-level controller for a quadrotor, linear and nonlinear methods can be used. It has been demonstrated in the literature that linear methods such as PID [16], H_∞ [17], and LQR controller [18] enable the quadrotor to perform fundamental movements successfully. Nonlinear controllers, such as feedback linearization [19], backstepping [20], sliding mode [21], adaptive [22], and model predictive control [23], have been shown to produce superior results when the environmental noise is increased or when the desired movement is made more complex.

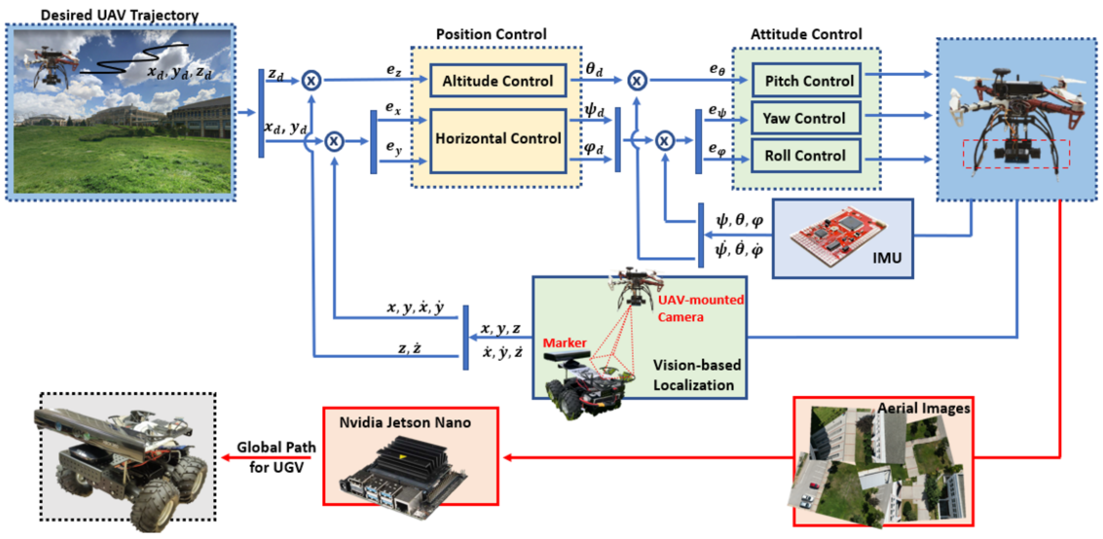
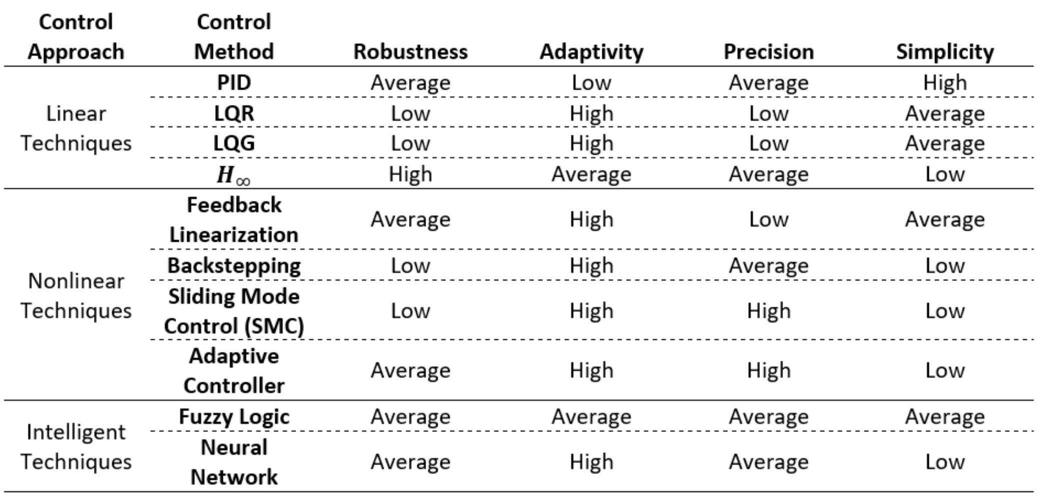


FIGURE 2.9: Block Diagram for UAV-UGV System

There exist some studies in the literature that compare the performances of the linear [24] and nonlinear [25] control strategies on quadrotor control. The table below shows the comparative performance analysis of the mentioned controller design methods:



Control Approach	Control Method	Robustness	Adaptivity	Precision	Simplicity
Linear Techniques	PID	Average	Low	Average	High
	LQR	Low	High	Low	Average
	LQG	Low	High	Low	Average
	H_∞	High	Average	Average	Low
Nonlinear Techniques	Feedback Linearization	Average	High	Low	Average
	Backstepping	Low	High	Average	Low
	Sliding Mode Control (SMC)	Low	High	High	Low
	Adaptive Controller	Average	High	High	Low
Intelligent Techniques	Fuzzy Logic	Average	Average	Average	Average
	Neural Network	Average	High	Average	Low

FIGURE 2.10: Controller Types and Performance Comparison

The given control techniques in the table above can be utilized alone or combined with another technique to control the quadrotor. Especially, intelligent control methods are preferred frequently to combine with linear and nonlinear techniques to increase controller performance. Model Predictive Control is another nonlinear control approach whose effectiveness directly depends on the reference model performance.

Nonlinear controllers control systems without subjecting them to a linearization process, as opposed to linear controllers, also produce positive results in situations that fall outside the linearization process's scope. Therefore, nonlinear controllers are frequently preferred in industrial projects or problems involving human-machine interaction because of their more conditional structure. They are more complex than linear controllers, but they function more reliably.

During the experiments, the linear control techniques PID and LQR are preferred to utilize because of their simple structure, and the suitable conditions of the indoor test environment. Linear control methods are mathematical models that control nonlinear systems, such as quadrotors, by linearizing them at specific equilibrium points. Therefore, they can give satisfactory outputs for low-level control processes.

a) PID Control

Proportional-Integral-Derivative (PID) is a linear control method that has the widest place in the control literature. Decouples the error, which is the difference between the desired signal and the feedback received, into 3 branches and multiplies it by the coefficients Kp, Ki, and Kd. It takes the integral of the error from the multiplier with the Ki coefficient and the derivative of the error from the multiplier with Kd. Thus, it makes clear how the system will react at different time intervals and tries to control it. Moreover, these coefficients can also be used in different combinations such as PI, PD, only P, and only I according to the needs while developing the controller.

$$u(t) = K_P e(t) + K_I \int_0^t e(t)dt + K_D \frac{de(t)}{dt} \tag{2.3}$$

Since the use of PID control is a simpler and faster method than other linear control methods, it has been frequently used in stabilizing and hovering motion in UAV control.

In addition, in some recent studies PID controllers have become better by using them together with intelligent control methods such as Fuzzy Logic [26], Neural Networks [27], Partial Swarm Optimization (PSO) [28], and Reinforcement Learning [29]. Moreover, there exist several studies in the literature which use adaptive techniques [30] [31] to find optimum PID parameters for the flight smoother and more precise. These intelligent techniques help to automatically tune the PID parameters.

b) LQR Control

Linear-Quadratic Regulator (LQR) is an optimization technique that is widely used in dynamic systems to operate them at around minimum cost by stitching the following equation [32]. LQR controller is mostly utilized with Kalman Filter to evaluate better the effect of environmental noises and missing parameters on flight performance.

$$J_{LQR} = \int_0^\infty (y_i)^T(t)Qy_i(t) + (u_i)^T(t)Ru_i dt \tag{2.4}$$

In the literature, the LQR controller is used to optimize trajectory, and develop motor and propeller models for better performance in these papers [33][34] respectively. Furthermore, in this study [35] LQR method is utilized with Lagrange's equation to regulate flight parameters and increase the reaction force of a quadrotor where the Lagrange equation can be expressed as the difference between kinetic (T) and potential energies (V) like $L = T - V$.

2.3.2 UGV Modeling and Control

Although wheeled robots typically come to mind when the term UGV is mentioned, UGV actually refers to all land-traveling unmanned robots. In the experiments conducted for this thesis, a healthy path for wheeled UGVs is sought. Wheeled ground vehicles can be classified as holonomic, nonholonomic, steering, differential drive, articulated drive, and independent drive based on the type of wheels that determine the heading angle of the vehicle, change direction, the number and position of wheels connected to the engines, and the degree of wheel freedom. Different types of holonomic UGVs are created based on the number and location of their users. In the table below, you can see some driving types and important features:

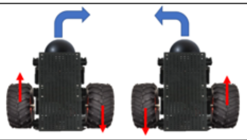
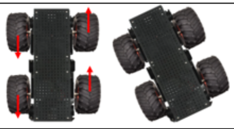
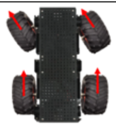
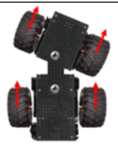
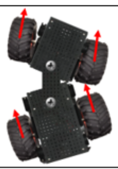
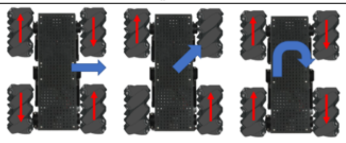
Differential Drive (Robot moves according to back two wheels' actions.)	
Skid Steering (Front and back motors stay in same side work simultaneously)	
Ackermann Steering (Front two wheels connect to same rotational ax.)	
Articulated Drive (Front left and right motors are independent from the back two motors)	
Independent Drive (All 4 motors can run independently and have different rotational axes)	
Omni Directional Wheel (Robot can move side and corner without change heading angle.)	

Figure 2.11: UGV Driving Types

Experiments conducted within the scope of the thesis made use of UGVs with skid steering.

The kinematic model is a mathematical description of the robot, and control algorithms are derived from the motion equations obtained from these models. Two-dimensional Cartesian coordinates may be used to represent the kinematic model of a skid steering UGV. This demonstration assumes that the robot rolls on its standard wheels without slipping. L represents the width of the robot, a represents the distance between the center of the wheels and the center of the mass, and $2r$ denotes the diameter of the wheels.

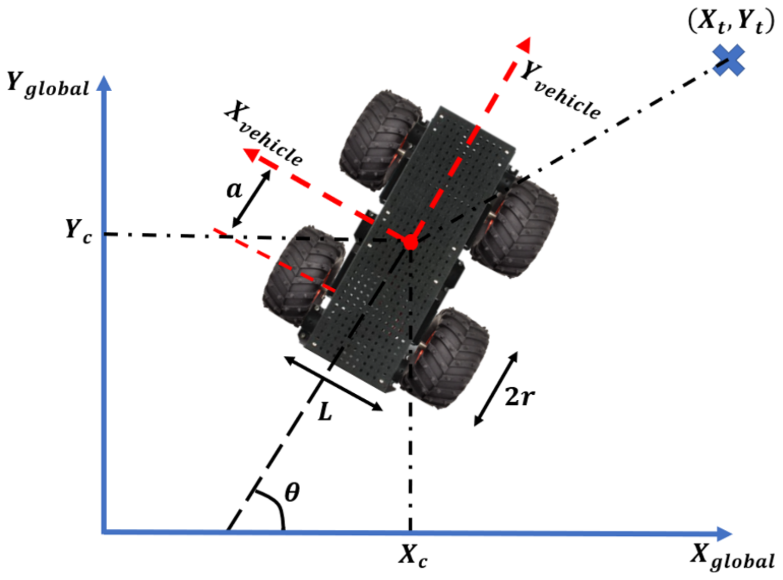


FIGURE 2.12: UGV in Cartesian Coordinate

When the slipping case is ignored, the UGV's equation of motion can be represented as:

$$\dot{x_c}(t)cos\theta(t) - \dot{y_c}(t)sin\theta(t) = a\dot{\theta}(t) \tag{2.5}$$

The equations can be represented as follows with pure rolling constraints:

$$\begin{aligned} \dot{x_c}(t)cos\theta(t) + \dot{y_c}(t)sin\theta(t) + L\dot{\theta}(t) = r\dot{\phi}_r(t) \\ \dot{x_c}(t)cos\theta(t) + \dot{y_c}(t)sin\theta(t) - L\dot{\theta}(t) = r\dot{\phi}_l(t) \end{aligned} \tag{2.6}$$

$$\begin{bmatrix} v_x(t) \\ v_y(t) \\ \dot{\theta}(t) \end{bmatrix} = \begin{bmatrix} r/2 & r/2 \\ 0 & 0 \\ -r/L & r/L \end{bmatrix} \begin{bmatrix} \omega_l(t) \\ \omega_r(t) \end{bmatrix} \tag{2.7}$$

$$\begin{aligned} v(t) &= r[\tfrac{\omega_r(t)+\omega_l(t)}{2}] \\ \dot{\theta}(t) &= r[\tfrac{\omega_r(t)-\omega_l(t)}{L}] \end{aligned} \tag{2.8}$$

$$\begin{bmatrix} \dot{x}(t) \\ \dot{y}(t) \\ \dot{\theta}(t) \end{bmatrix} = \begin{bmatrix} cos(\theta) & 0 \\ sin(\theta) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} V \\ \omega \end{bmatrix} \tag{2.9}$$

where x_0, y_0 are the initial x-coordinate of UGV, x_t, y_t are the target coordinates, x_c, y_c are instantaneous x and y coordinates, $v_x(t), v_y(t)$ are the longitudinal and lateral velocities, θ is heading angle of the UGV, and finally, $\omega_r(t), \omega_l(t), \omega(t)$ are right, left and resultant velocities of the UGV.

These kinematic equations permit remote control of the UGV. Trajectory control and parking control are the most fundamental UGV control problems. In order for the UGV to follow a trajectory decently, the difference between the trajectory and the robot's path must be constantly controlled, and the motors must receive input to compensate for this error.

Even though skid steering UGVs are powered by four engines, the engines on the same side receive the same inputs. Therefore, the vehicle operates similarly to UGVs with differential drive but is more powerful than differential drive vehicles due to a greater number of engines. In contrast, skid steering UGVs have two inputs and three outputs (x, y position, and heading angle) and are therefore underactuated.

In trajectory, the relation between the control inputs, where the u_1 is v and u_2 is ω, and the equation of motion of the skid driving UGV can be expressed as:

$$\begin{aligned} \ddot{x} &= \dot{u}_1 cos\theta - u_1 u_2 sin\theta \\ \ddot{y} &= \dot{u}_1 sin\theta + u_1 u_2 cos\theta \end{aligned} \tag{2.10}$$

From here it is easy to write:

$$\begin{aligned} \dot{u}_1 &= \ddot{x} cos\theta - \ddot{y} sin\theta \\ u_2 &= \tfrac{1}{u_1}(\ddot{y} cos\theta - \ddot{x} sin\theta) \end{aligned} \tag{2.11}$$

Reference trajectory and error between the reference and actual positions can be expressed as:

$$\begin{bmatrix} u_{1_{ref}} \\ u_{2_{ref}} \\ \theta_{ref} \end{bmatrix} = \begin{bmatrix} \dot{x}_{ref} cos\theta_{ref} + \dot{y}_{ref} sin\theta_{ref} \\ \ddot{y}_{ref}\dot{x}_{ref} - \ddot{x}_{ref}\dot{y}_{ref})/(\dot{x}_{ref}^2 \dot{y}_{ref}^2) \\ arctan(\dot{y}_{ref}/\dot{x}_{ref}) \end{bmatrix} \tag{2.12}$$

The tracking errors of $x, y\theta$ can be transformed to the pose of the actual robot by the following equation:

$$\begin{bmatrix} e_1 \\ e_2 \\ e_3 \end{bmatrix} = \begin{bmatrix} cos\theta & sin\theta & 0 \\ -sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} e_x \\ e_y \\ e_\theta \end{bmatrix} \tag{2.13}$$

At this stage, since the norms of $[e_x \;\; e_y \;\; e_\theta]^T$ is equal to $[e_1 \;\; e_2 \;\; e_3]^T$, the transformed errors are bounded when only the tracking errors are bounded. The relation between the control inputs and the transformed trajectory errors can be expressed as:

$$\begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} -k_1 e_1 + u_{1_{ref}} cose_3 \\ -u_{1_{ref}} sine_3/e_3 - k_{t_2} e_3 + u_{2_ref} \end{bmatrix} \tag{2.14}$$

where k_{t_1} and k_{t_2} are positive control gains.

The only difference between the parking problem and the trajectory problem is the reference inputs. In the parking problem, there are no continuous reference x, y, θ values. Therefore, the case can be expressed by following the same steps but making these reference signals equal to zero. In light of it, the parking problem can be formalized as:

$$\begin{bmatrix} u_{1_p} \\ u_{2_p} \end{bmatrix} = \begin{bmatrix} -k_{p1} e_{1_p} \\ -k_{p_2} e_{3_p} + e_{2_p}^2 sin(t) \end{bmatrix} \tag{2.15}$$

where the $e_{1_p} and e_{2_p}$ are the errors between the reference parking locations and actual locations, and the e_{3_p} represents the difference between the desired heading angle and the actual one.

The ground vehicle must therefore follow the waypoints displayed on the images transmitted by the UAV while avoiding both dynamic and static obstacles. In areas that cannot be scanned by the UAV, such as under trees, the path should be performed using onboard sensors on the UGV. The block diagram for this situation is shown below.

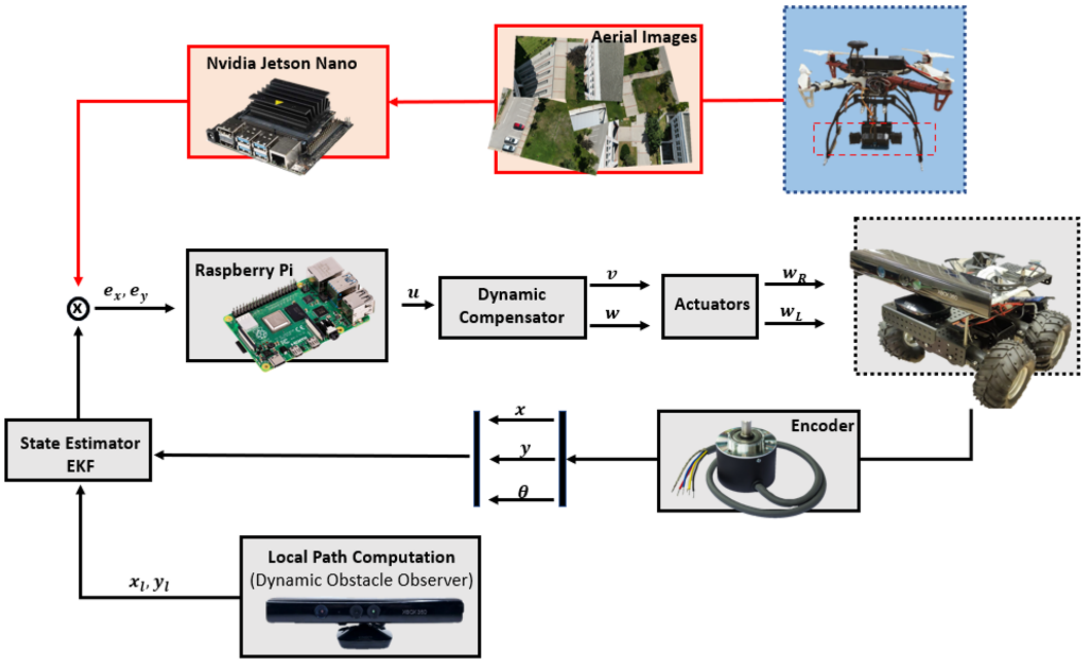


FIGURE 2.13: UGV Control Block Diagram

After the waypoints have been transferred, various methods for UGV position control can be implemented. One is to continuously follow the UGV from the air throughout the journey with the UAV, and the other is to use a sensor, such as an encoder, that provides position feedback to the UGV's engines. In the experiments conducted for the thesis, an encoder was used to control the motion of the UGV.

2.4 Image Stitching Techniques

Image stitching is a method of combining multiple images by utilizing the characteristics or pixel intensities of the overlapping fields of the input images to generate a panoramic image or increase its resolution.

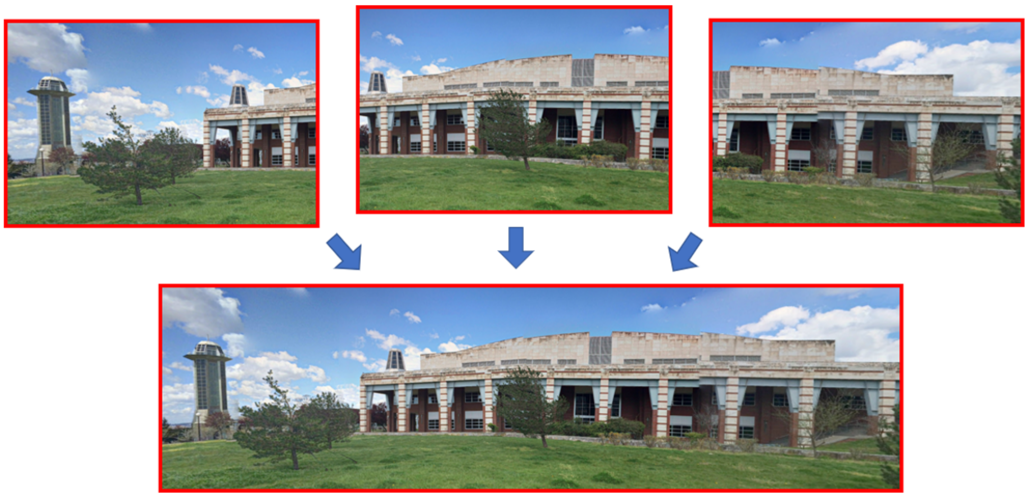


FIGURE 2.14: Image Stitching Example

The input images must contain a sufficient number of common areas and detectable features or pixel value differences. In the absence of these conditions, parallax errors may occur. Wide-angle cameras, such as a fisheye camera, can be used to overcome parallax errors in a confined area. However, because these cameras attempt to capture larger areas in a single shot, the image resolution degrades and the imaged objects become more curved. It is also possible to create blind stitching by employing a feature-based alignment method; however, these methods typically produce inaccurate results.

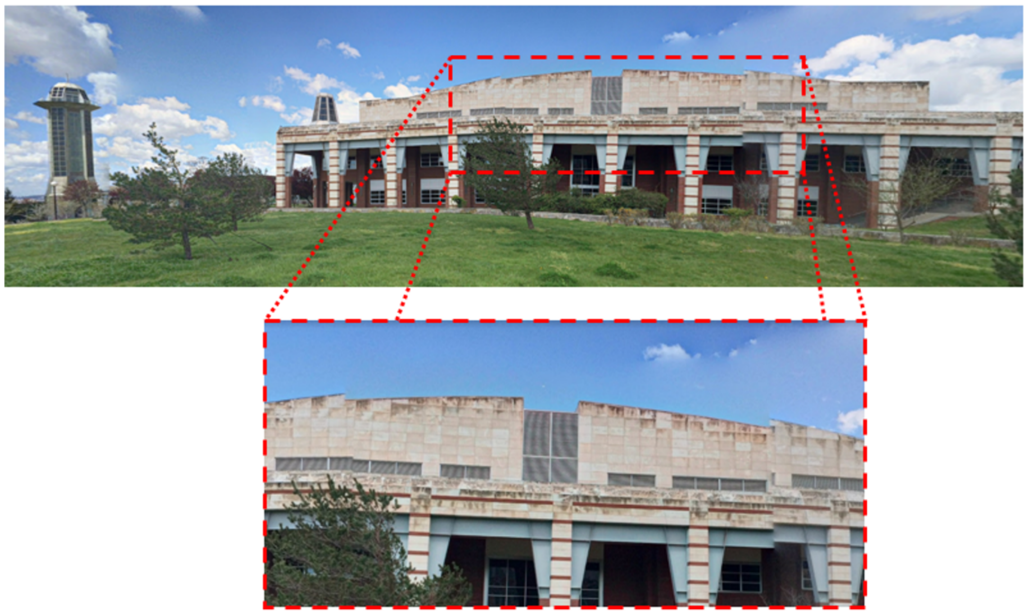


FIGURE 2.15: Parallax Error

There are two primary methods for image stitching: the Direct Method, which focuses on the pixel intensities of the overlapping section, and the Feature-based Methods, which utilizes the features of the common fields of input images.

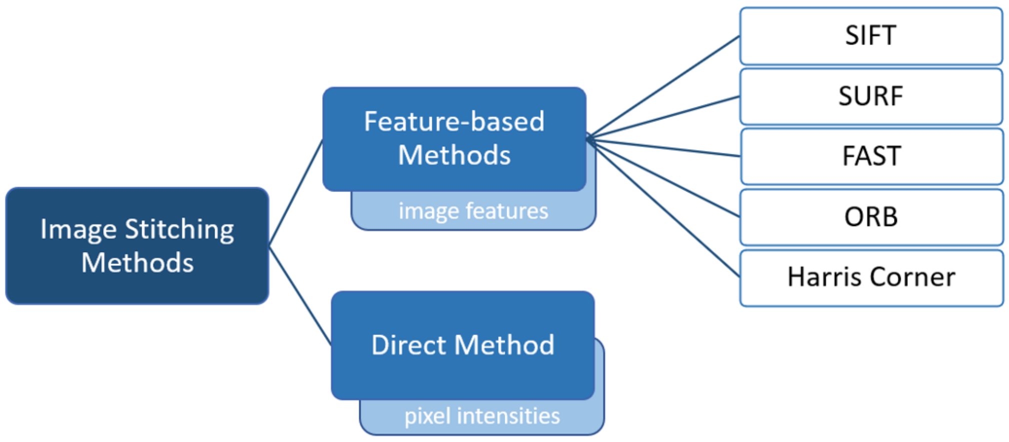


FIGURE 2.16: Stitching Method Classifications

The Direct Method operates on the basis of comparing the pixel intensities of the two images. The evaluation of input images at the pixel level is the technique's most advantageous characteristic. Thus, it becomes possible to identify significant similarities between the images. In contrast, the computation time of pixel-level evaluation and the possibility of multiple identical pixel nodes make the technique quite complex. In addition, the limited convergence range and restrictions on scale and rotation are significant drawbacks of the technique.

Multiple methods, including SIFT, SURF, FAST, ORB, and Harris Corner, constitute Feature-based Techniques. After extracting the overlapping fields, these techniques all aim to stitch images by utilizing the common features in the overlapping fields. The method yields faster results than the Direct Method because it detects the similarity of input images based on a few key feature points rather than scanning all pixels. In addition, because of the size difference between features and pixels, the use of common features on overlapping fields is more reliable than pixel-level comparison [36]. In contrast, outputs from feature-based image stitching techniques are unsuitable when input images have a single color or an entirely flat background. In addition, the interpretation of loss data of input images in traditional feature-based techniques such as SIFT, SURF, FAST, and ORB is quite limited. Using methods of deep learning, a number of novel approaches developed over the past decade attempt to satisfy this specification.

- SIFT: Scale-Invariant Feature Transform (SIFT) is an algorithm for computer vision that detects local features in RGB images based on the gradient value. It works quickly and produces desirable results against light and noise. Due to the fact that its efficiency decreases proportionally with pixel quality, it cannot produce robust results in remote sensing images with low pixel quality. In 1999, the technique was developed.
- SURF: Speeded Up Robust Features (SURF) is a stitching technique designed to accelerate the feature-matching process. It accelerates the removal of detected features by monitoring the quality of their pixels. Consequently, the SURF method also increases the speed of the SIRF method. Moreover, the computational cost of SURF is less than that of SIFT because it does not compare features of low-pixel-quality images. However, the evaluation of pixel quality generates additional computational load. It was first mentioned in a publication in 2006.
- FAST: Features from the Accelerated Segment Test (FAST) method function by referencing the corners. It provides faster results than all other feature-based algorithms and does not degrade pixel quality. The only significant drawback of the algorithm is its lack of orientation and rotation capabilities. It was introduced in 2006 but is still in use today.
- ORB: ORB is a feature detection algorithm that aims to produce output images faster than its alternative, SIFT. Although the ORB technique was developed 12 years after the SIFT method, it exhibits comparable light and image noise performance. However, it compares input image features in less time and with less computational memory.
- Harris: Harris is an algorithm for detecting corners that detect points by comparing the difference in pixel intensity in a window-sized region. It is a well-known method in computer vision applications and is also utilized in image stitching

operations by matching the same image corners. It identifies the corners with high accuracy in overlapping fields of the input images. However, the case may cause excessive processing time if the scanned area contains a large number of corners, or it may fail if the overlapping area consists of a solid color and flat structure.

Chapter 3

Terrain Analysis and Topographic Map Building for Heterogeneous Robot Systems

3.1 Depth Reconstruction Techniques for Terrain Analysis

Depth estimation is a computer vision task designed to estimate depth from an image or image pairs. This technique is highly favored in a variety of fields, including autonomy and the space industry. There are various methods for deriving the depth from stereo images. In addition, after providing a properly calibrated camera and employing computer vision techniques and machine learning algorithms, real-time detection is possible. Nonetheless, in order to interpret the results, it is necessary to obtain the key concepts and major technical descriptions. It is essential to understand the definitions and significance of stereo vision, depth recognition, stereo and monocular images, and projection types.

Depth analysis can be successfully performed with various ultrasonic or infrared sensors. However, access to these sensors and their outputs may not always be useful. Consequently, using camera images, image differences, and intrinsic and extrinsic camera parameters, the distances of the image's objects to the camera can be estimated with Decisive accuracy. If the distance between the cameras, known as the baseline, and the camera parameters are known, depth analysis of an area can be decoded using two cameras from a fixed point. This technique is known as stereo reconstruction.

Additionally, depth analysis can be performed using a single camera. This method cannot perform depth analysis from a fixed point, unlike stereo vision. In order to detect depth, it captures images from various locations and analyzes the images and camera positions to identify common features. This technique is also known as structure from motion.

In the following section, the performance of various machine learning classifiers on the stereo depth reconstruction algorithm are discussed.

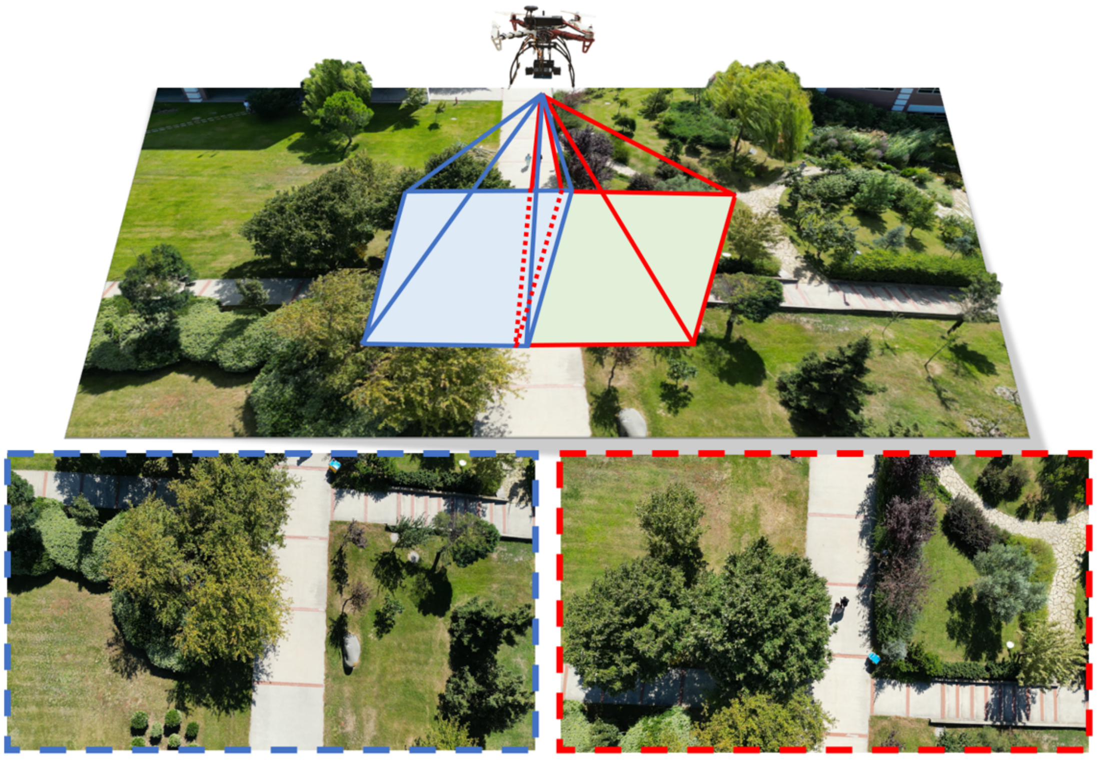


FIGURE 3.1: Terrain Depth Reconstruction with Stereo Vision

3.1.1 Stereo Vision Technique for Depth Reconstruction of Terrains

The estimation of depth is one of the most important techniques in the field of 3D representation. Stereo vision is also one of the most frequently used terms for recognizing depth in real-time and offline applications. Charles Wheatstone researched this term in 1838, despite the fact that it is commonly used today. Several new terms, including stereopsis, which is short for the retinal disparity, were added to the lexicon as a result of this study. After focusing on stereopsis, he invented the stereoscope and stereoscopy grew in popularity among the public. Due to its increasing popularity, more researchers have begun to investigate its limitations and the relationships between single vision. Since the 19th century, researchers have devoted time to this subject in order to make discoveries.

The first step in estimating depth from stereo imagery is to divide the images into positive and negative patches. Due to the time-consuming nature of manually aligning stereo images, this must be accomplished using vision techniques. Therefore, both the right and left images should split at least 3x3 pixels of uniform colors. The differences between the patches are based on their shapes, sizes, and relative geometries. Then comes the matching operation. There are a number of methods for matching stereo images, but they can be grouped into two primary categories: area-based correlation methods and feature-based methods [37].

In the correlation phase, methods scan pixel by pixel to identify relationships between the small rectangular areas of the first and second images. Feature-based methods, on the other hand, seek to identify a certain class of feature in two images, with the primary goal of matching images in less time and extracting higher-level information from them. After matching pixels from two input images, all collected results are stored in a map known as a disparity map. As it stores the horizontal distance between two matching pixels and their coordinates as $d(x, y)$, the map can be described in greater detail. However, to provide a basic understanding, the fundamental equations are provided below.

$$\begin{gathered}
x_{right} = x_{left} - d_{left}(x_{left}, y) \\
x_{left} = x_{right} - d_{right}(x_{right}, y) \\
x_{search} = x_{ref} + s d_{ref}(x_{ref}, y) \\
\text{when } s = 1 \text{ and } s = -1 \\
y = y_{left} = y_{right}
\end{gathered} \tag{3.1}$$

So the relations of $d_{left}(x; y)$ and $d_{right}(x; y)$ can be seen above equations, it is also possible to represent special conditions such as coordinate shifting with these equations.

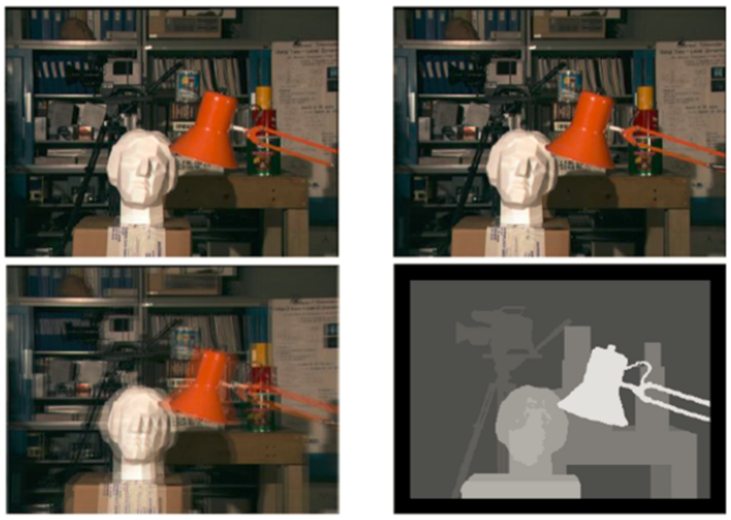


FIGURE 3.2: a) reference (left) image b) search(right) image c) superimposed left and right view d) disparity map

As mentioned in previous parts, the depth can be calculated from the geometrical approaches such as triangulation that comes from the stereo image pair.

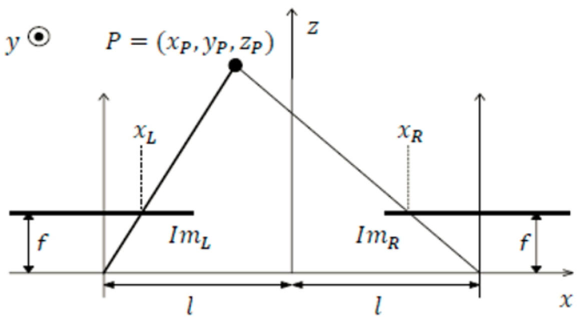


FIGURE 3.3: Geometric Representation of Stereo Vision

A similar schematic is already explained in Figure 3.2, and the mathematical relations are also expressed below:

$$\begin{gathered} d = x_L - x_R = f((x_P + l)z_p - \frac{x_p - l}{z_p}) = \frac{2fl}{z_p} \\ z_p = \frac{2fl}{d} \end{gathered} \tag{3.2}$$

If the camera lens parameters are known, such as focal length f and baseline distance $T = 2l$, it is possible to calculate depth. As evidenced by the preceding definitions, the position and specifications of certain physical elements, such as the lens properties of the camera and the position of the object, are essential for depth calculations. In addition, it is essential to capture adequate light in an appropriate environment while avoiding reflection, transparency, and mirror surfaces. Consequently, the quality of images captured is directly impacted by these conditions. On the other hand, using a physical setup is not the only way to conduct this experiment; it is also possible to use a dataset.

3.1.2 Comparative Analysis of Hand-Crafted Feature-based Approaches and Convolutional Neural Network (CNN) Approach in Stereo Depth Reconstruction

There are a number of techniques for estimating depth from stereo image pairs. In this section, the performance of various machine learning and deep learning methods for terrain analysis is compared to determine the optimal method. A dataset is utilized because the accuracy of the methods can only be determined when the ground truth is known.

As previously described, stereo matching is typically performed using block-based or feature-based methods. Small blocks or patches are matched on the left and right images based on their similarity measurement for block-based matching. There are fundamental similarity measurements, such as SAD or Sum of Absolute Differences, that measure how similar these two patches are, i.e., the smaller the value of SAD, the more similar the patches [38].

Various machine learning algorithms are employed in the hand-crafted feature-based method for the similarity measurement task. Specifically, binary classification is required to determine whether two given blocks or patches are identical or not, i.e. binary classification methods are used for measuring similarity. In addition, deep learning techniques are used as feature extractors and classifiers to compute depth based on similarity measurement.

3.1.2.1 Hand-Crafted Feature-based Approaches

a) Methods:

In this section, various machine learning classifier methods are used to calculate disparity depth maps for stereo images. When two blocks or patches from the left and right images are compared for similarity, classification occurs. A classification method is essentially trained to determine whether these two patches from the left and right images represent the same object or location. In order to achieve the block-based stereo matching, machine learning classifiers are used to measure similarity.

Since this is a classification issue, there are essentially two phases called training and testing. In this case, the feature vectors $x \in R^n$ and their corresponding labels $Y \in -1, 1$are required to train the classifier models. First, positive and negative pairs comprised of patches from the left and right images must be generated. For the positive pair, the left image is traversed pixel-by-pixel from $i_1 = 0$, $j_1 = 0$ to $i_1 = width$, $j_1 = height$, the ground truth disparity value d is retrieved at that $i1, j1$ position, and the position of the same location in the right image is calculated by $i_2 = i_1 + d$ and $j_2 = j_1$. Using the ground-truth disparity value, identical patches on the left and right images are retrieved and coupled to positive pairs.

To create a negative pair consisting of less similar patches, the sum of the absolute differences between the patches is used to calculate their similarity. Since the positive pair consists of very similar or identical patches, their sum of absolute differences should be small, whereas the negative pair's sum of absolute differences should be much larger. Consequently, a negative pair is generated by traversing the right image at the same j_2 location as the positive pair, creating patches by sliding pixels in the same row j_2, coupling this right patch with the left patch which is identical to the positive pair, and comparing their similarity measurements by calculating the sum of absolute differences. Then the right patch is chosen based on its similarity value, which is significantly greater than the positive value when paired with the left patch to form a negative pair. In addition, the pairs are created using various block sizes, including 7, 11, and 15. However, it has been determined that the best performance is achieved with a block size of 7, so this block size is utilized for the remainder of the experiments.

The next step is to craft feature vectors, x as defined above, after creating the positive and negative pairs. For each pair, a separate feature vector representing the positive and negative features is required. Therefore, these vectors must be distinguishable based on their respective labels $Y \in [-1, 1]$, which represent a positive or negative pair. To achieve this, the absolute differences between the patches of the pairs are used to create a single patch, which is then flattened into a vector. Since the positive pair consists of similar or identical patches, the feature vector of the positive pair should contain smaller values, whereas the feature vector of the negative pair should contain larger values because it consists of dissimilar patches. It also investigated the possibility of using the histogram of oriented gradients (HOG) feature descriptors to distinguish between positive and negative pairs. There were too many positive examples for each patch, and there was no correlation between the similarity of the HOG values and the disparity of the patches in relation to the disparity of the ground truth. Therefore, it is determined that the HOG feature descriptor would not serve the desired purpose (it is typically used to identify specific objects in an image) and decided to continue generating feature vectors using the absolute differences between each pair of patches.

During the training phase, once the feature vectors are created with the corresponding labels, they are fed into the classifier in order to fit the data. In the testing phase, the models predict the disparity for each pixel using the feature vectors for each pixel in the test images. From this, a new disparity map is generated, the results are compared to the disparity map based on the ground truth, and a report on the model's efficacy is generated. There are a number of hyperparameters that must be adjusted in order to alter the results of these tests. There are 'block size' and 'negative offset multiplier' hyperparameters for the creation of feature vectors (size of patches used in measuring similarity between a patch on the left image and a patch on the right image) For testing purposes, there is a 'number of disparities' hyperparameter that specifies the maximum pixel shift to search for similarities. It is essential to remember that each model has its own hyperparameters. The majority of used models can be retrieved from the OpenCV and Scikit-Learn libraries.

b) Dataset:

There are numerous datasets for stereo imagery, including Kitti, CVC, Middlebury, and Flickr1024, among others. Their shared characteristic is that they consist of various images taken from various angles, as well as the hyperparameters, position, and specifications of cameras and used physical materials.

Middlebury 2001 stereo dataset containing piecewise planar scenes, close-shot images, and ground-truth disparity maps for 55 image pairs in 65 is preferred during the process. Piece-wise planar scenes include six images per set and a ground-truth disparity map for left and right images. Since the feature vectors that are fed to classifiers are derived from the images, only six images are required to generate a massive dataset. When patches are 7x7 in width and height, there are 8836 feature vectors for an image with dimensions of 100 x 100. Consequently, nearly 900k feature vectors were generated from these six images. In addition, each ground-truth disparity map is scaled by an eightfold factor.

The dataset is divided into train and test sets, each of which contains four and two images, respectively. Using the training images, train and validation features with their corresponding labels are generated. Using the test images, test features and their corresponding labels are generated. The number of trains, validation examples, and sample tests is listed in the table 3.4.

	Train	Validation	Test	Total
Positive Pairs	370216	92554	233321	696091
Negative Pairs	370216	92554	233321	696091
	740432	185108	466642	1392182

FIGURE 3.4: Dataset Details

c) Experimental Results:

Various machine learning and deep learning method performances are compared in order to identify the optimal classifier. There are two distinct methods for constructing training and testing datasets' features. There is an additional method used to generate feature vectors distinct from absolute differences between patches. This is accomplished by directly flattening and concatenating the left and right patches without applying any operation. Therefore, it is intended to determine if these patches can be distinguished on their own. As anticipated, these types of features are not particularly explanatory, so classifiers struggled to determine whether the given vector represents a positive pair or a negative pair. It is possible to view the utilized classifiers and their respective performance.

- **Multinomial Naive Bayes**

Training begins with the Multinomial Naive Bayes method. As this is a binary classification task, there are only two classes to predict: positive or negative, indicating that the feature vector is composed of positive patches (similar or the same) or negative patches (dissimilar). Consequently, there are two parameters to estimate using maximum a posterior probability θ_y. In addition, the parameters are estimated using a modified version of maximum likelihood:

$$\hat{\theta}_{y_i} = \frac{N_{y_i}+\alpha}{N_y+\alpha n} \tag{3.3}$$

Here α is set to 1 to avoid the error associated with zero-occurrence examples. This classifier is not particularly effective. The reason for these outcomes is that the Multinomial Naive Bayes algorithm considers the features as the count of occurrences of that feature across the entire dataset. There is no Multinomial assumption because the features in this dataset are computed as the absolute difference between the positive and negative patches.

- **Gaussian Naive Bayes**

The second classifier to be trained in the Gaussian Naive Bayes classifier. In contrast to the Multinomial assumption, here it is assumed that the features are normally distributed, as explained below. Here, the parametersμ and σ parameters are estimated.

$$\hat{\theta}_{y_i} = \tfrac{N_{y_i}+\alpha}{N_y+\alpha n} \tag{3.4}$$

Since the features range from 0 to 255, they are likely to be Gaussian distributed across the entire dataset. Accordingly, this model achieves higher scores than the Multinomial model, as shown in Figure 3.6.

MSE = 1588 demonstrates that when the model is applied to the stereo method as a similarity measurement checker, it does not produce particularly good results. To increase the scores and improve the disparity map, various stereo parameters, such as the number of disparities, are utilized. However, the best MSE is obtained by using a block size of 7 and a number of differences of 16.

- **Decision Tree**

A Decision Tree is trained with the same training set and has partitioned the tree in a manner that allows it to generalize effectively. The Gini index measures the impureness of the nodes. Entropy was also used as a measure of impurity, but there was no significant performance difference, so experiments are conducted using the Gini index. Furthermore, since the dataset created with a patch size of

7 contains 49 features, maximum depth or maximum feature-like constraints are not applied. After training, the tree's depth is 20 and there are 231 leaf nodes.

The Decision Tree model performs better than the Multinomial Naive Bayes model but similarly to the Gaussian Naive Bayes model. In contrast to these models, however, Decision Tree generalizes better than the others. Consequently, this model outperforms the others based on the disparity map perspective. As depicted in the figure below, the disparity map computed using the Decision Tree is comparable to the ground truth and exhibits less MSE than other methods.

- **Logistic Regression**

Logistic regression is an additional linear classifier utilized in experiments to predict the dissimilarity between image patches. This probability for a given class A is calculated as follows:

$$P(Y|X) = \frac{exp(\omega_0 + \Sigma\omega_i X_i}{1 + exp(\omega_0 + \Sigma\omega_i X_i)}$$

$$\omega_0^{t+1} \leftarrow \omega_0^t + \eta \sum y^j - P(Y^j = 1|x^j, \omega^t) \tag{3.5}$$

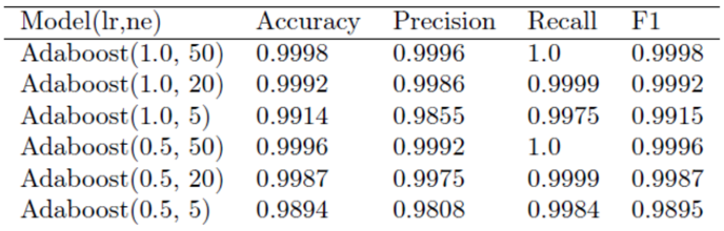
$$\omega_i^{t+1} \leftarrow \omega_i^t + \eta \sum x_i^j y^j - P(Y^j = 1|x^j, \omega^t) \ \ \textbf{for} \ \ i = 1, ..., d$$

where η is the learning rate of the gradient descent algorithm.

- **AdaBoost**

AdaBoost (short for Adaptive Boosting) is an algorithm that combines multiple weak classifiers to create a single robust classifier. AdaBoost is an iterative algorithm that trains a single classifier type on a dataset. Throughout the evaluations, AdaBoost Classifier accessed the scikit-learn library. The weak classifier chosen for the AdaBoost ensemble is a one-depth decision tree. This signifies that, given the 49 feature vectors from the patches of size 7, the Adaboost classifier selects the single feature vector that best distinguishes positive and negative pairs and builds an ensemble based on the errors of the previous classifiers. The AdaBoost classifier's hyperparameters are the learning rate (lr) and the number of estimators (ne). A higher learning rate correlates to a greater reduction in the contribution of each weak classifier, whereas an increase in the number of estimators correlates to an increase in the complexity of the classifier.

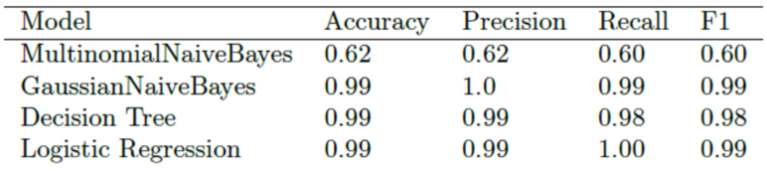
Given various sets of hyperparameters, the table below illustrates the performance of the AdaBoost Classifier in distinguishing between positive and negative pairs.



Model(lr,ne)	Accuracy	Precision	Recall	F1
Adaboost(1.0, 50)	0.9998	0.9996	1.0	0.9998
Adaboost(1.0, 20)	0.9992	0.9986	0.9999	0.9992
Adaboost(1.0, 5)	0.9914	0.9855	0.9975	0.9915
Adaboost(0.5, 50)	0.9996	0.9992	1.0	0.9996
Adaboost(0.5, 20)	0.9987	0.9975	0.9999	0.9987
Adaboost(0.5, 5)	0.9894	0.9808	0.9984	0.9895

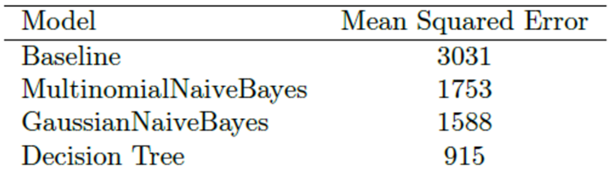
FIGURE 3.5: AdaBoost Hyperparameters

The performances of the employed ML algorithms on the same dataset can be seen in the tables below:



Model	Accuracy	Precision	Recall	F1
MultinomialNaiveBayes	0.62	0.62	0.60	0.60
GaussianNaiveBayes	0.99	1.0	0.99	0.99
Decision Tree	0.99	0.99	0.98	0.98
Logistic Regression	0.99	0.99	1.00	0.99

FIGURE 3.6: Comparative Performances of the Employed ML Algorithms



Model	Mean Squared Error
Baseline	3031
MultinomialNaiveBayes	1753
GaussianNaiveBayes	1588
Decision Tree	915

FIGURE 3.7: MSE of the Employed ML Algorithms

The performance differences between the algorithms can be seen better in the figures below:

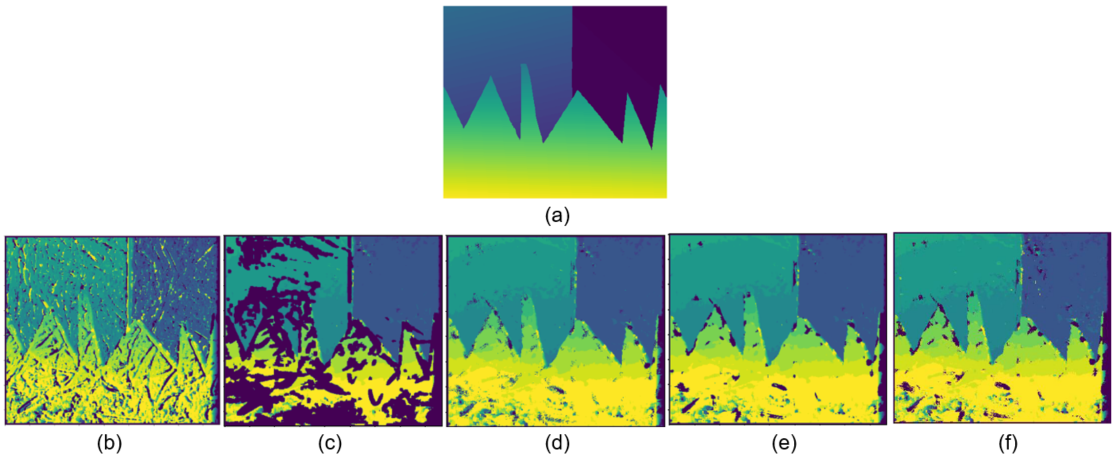


FIGURE 3.8: a) Ground Truth, b)Naive Bayes c)Gaussian Naive Bayes d)Logistic Regression e)Decision Tree e)AdaBoost

3.1.2.2 Convolutional Neural Network Approach

Up until this stage, every feature has been hand-crafted using absolute differences. Although these characteristics appear to be discriminatory, they help distinguish positive pairs from negative ones, whereas the computed disparity maps are less promising. Consequently, following all of these studies and experiments, neural network-based approaches are also utilized to allow the network to generate its own features. Thus, the application of convolutional techniques to the stereo vision problem is investigated. Fully Convolutional Neural Network (CNN) and Densely Connected Convolutional Neural Network methods are used in the scope of the experiments, with the results of the Densely Connected Convolutional Neural Network presented below because they are more lucid.

- **Densely Connected Convolutional Neural Network**

The primary objective of this method is to use convolution layers to generate robust features from stereo image pairs, which are then utilized within a cost function to compute the disparity map. To achieve this, typically multiple convolution layers are employed, and the resulting feature maps are then utilized within dense layers [39] or fed directly into the cost function, such as cosine similarity [40].

Similar to previous approaches, the first method expects a pair of patches as input, computes the feature maps on these patches, feeds the computed feature maps into a linear layer, and then outputs through a single neuron after applying Sigmoid to indicate whether or not these patches are similar [39]. This method is trained on the same dataset used to train other approaches. However, the model has not made any progress during training. There is no specific reason why the model did not acquire any knowledge. It is possible that the model described in the paper was not properly implemented, or that some of the hyperparameters were incorrectly configured. Consequently, it was unable to obtain meaningful results using this method.

After these failures, the other approach described in [40] is thoroughly investigated. Similar to the paper, the network is implemented with 5 consecutive convolution layers, with the output of each layer connected to the subsequent layers, as a DenseNet [41]. Convolution layers consist of various numbers of 3x3 kernel-sized filters ranging from 64 to 256. In addition, TanH is used as an activation function instead of ReLU because experiments [42] have demonstrated that TanH produces superior results. The overall model structure is depicted in the figure below, which is derived from [40].

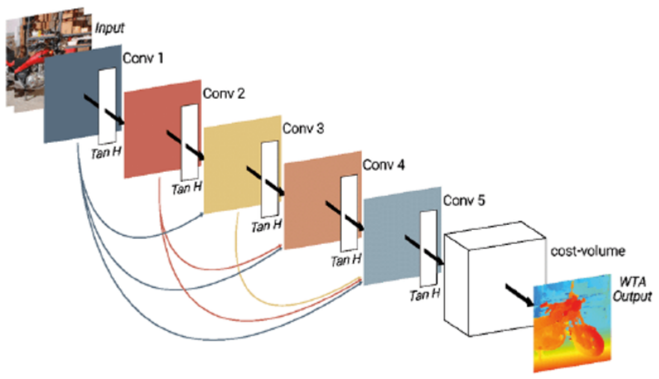


FIGURE 3.9: Densely Connected Convolutional Neural Network from FC-DCNN: A densely connected neural network for stereo estimation by Dominik Hirner and Friedrich Fraundorfer

In addition, a post-processing method is defined for applying to the computed disparity maps in order to remove noises and improve the accuracy of the maps.

A median filter with a 5x5 kernel is applied to the computed disparity map to eliminate the salt-and-pepper noises caused by undefined disparity points.

To evaluate the performance of this CNN model on different data, another dataset containing images collected from the more recent Middlebury sets was created. In contrast to the previous dataset, this one includes 13 training images, 2 validation images, and 3 testing images. This dataset contains images with greater width and height than the previous ones, which had a width of 659 pixels and a height of 497 pixels. In addition, the lighting conditions of these images differ from those of the preceding images; consequently, it is a bit difficult to find corresponding features on the left and the images due to this light difference.

The model, a densely connected convolutional neural network, is trained multiple times on datasets created with varying patch size and O_{neg} parameters, another hyperparameter associated with negative patch locations. All training sessions lasted nearly 50 epochs and ended after the validation loss was not reduced three times. Since each model is only created and stored once, the training, validation, and test data are identical for all models. Consequently, it makes sense to compare the scores of the models.

On the same training set, pairs of patches measuring 7x7, 11x11, and 21x21 are used to train the models. In addition, the values 20 and 4 are selected from the aforementioned ranges for the O_{neg} value. After training with these parameters according to the training procedure described above, mean squared error (MSE) and 2 point-error (2-PE) scores on the validation set are computed as shown in the table below. The 2-PE score is the proportion of misclassified pixels, where the difference between each pixel in the computed disparity map and the ground truth disparity map is greater than two-pixel values.

Patch Size	O_{neg}	2-PE	MSE
7×7	20	0.25	21.04
7×7	4	0.24	21.57
11×11	20	0.23	20.52
11×11	4	0.24	21.64
21×21	20	0.23	19.25
21×21	4	0.22	19.74

FIGURE 3.10: Patch Size Effects on MSE and PE

This model was trained using the previous dataset, which was comprised of 7x7 patches; (O neg) was chosen based on the dissimilarity score. Additionally, the model is trained for 46 epochs and terminated according to the same early stopping criteria. As shown in the figure below, the computed disparity map is vastly superior to those produced by the previous method. In addition, MSE is significantly lower than all previous methods. Therefore, the convolutional neural network approach outperforms hand-crafted feature-based approaches by a wide margin.

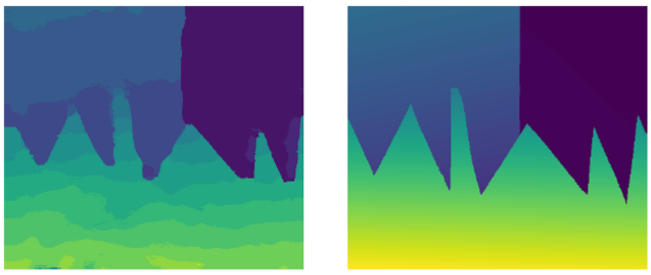


FIGURE 3.11: CNN Performance on the Input Image where 2-PE=0.39, MSE=14.53

These comparative analyses demonstrated that hand-crafted methods with machine learning classifiers and one of the most popular deep learning techniques, CNN, produce outputs with varying efficiencies due to their structures and hyperparameter settings. The Densely Connected Convolutional Neural Network method is used to compute the disparity map of the terrain images captured by the UAV in the scope of the thesis experiments, as the results indicate that CNN yields the most accurate and efficient results with the least amount of error.

3.1.3 Structure from Motion Technique for Depth Reconstruction of Terrains

Structure from motion is another technique for estimating the 3D structure of a terrain or an object using 2D images captured by a monocular camera from multiple angles or by multiple monocular cameras with known relative positions. Due to the restriction between the baseline and the UAV height, stereo cameras cannot be used to estimate the depth beyond a certain height during surface depth analysis. Consequently, a depth analysis was conducted using the monocular camera on the quadrotor, which was in motion using the structure from motion technique.

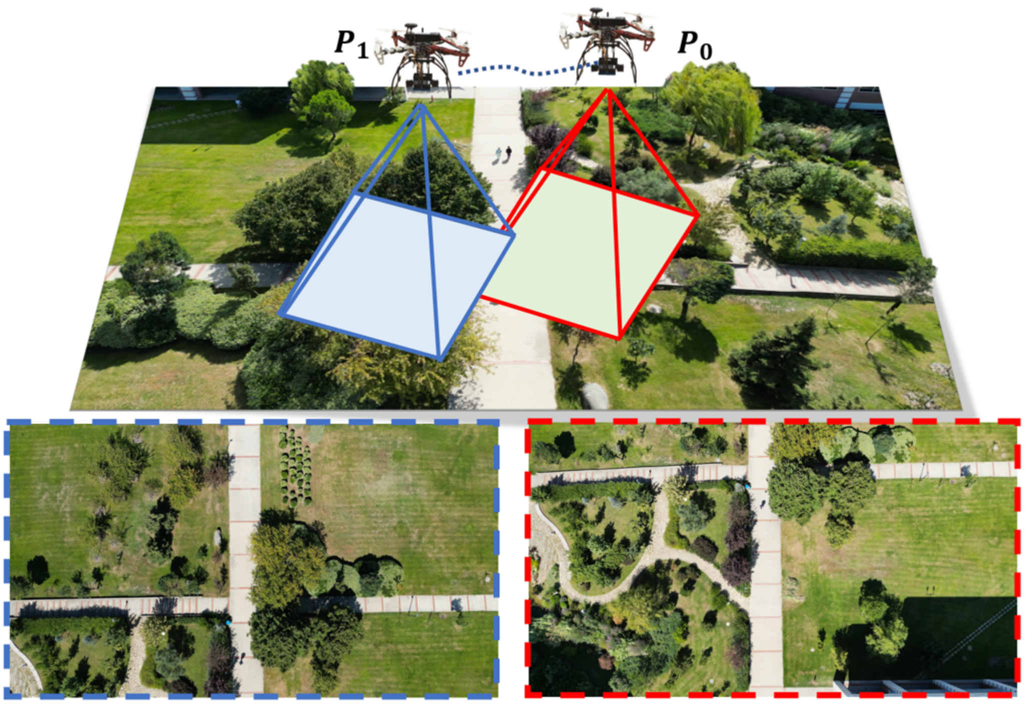


FIGURE 3.12: Terrain Depth Reconstruction with Structure from Motion

The steps of Structure from Motion (SfM) closely resemble the stereo vision depth reconstruction method. It also follows feature extraction and matching steps and then finds fundamental and essential matrices respectively. This step also finds the intrinsic and extrinsic parameters of the camera. Then relative camera or image poses are estimated using an essential matrix. The triangulation procedure then proceeds according to the geometric epipolar structure. Bundle adjustment, which is a technique for reconstructions, is the final step prior to obtaining the outputs.

The most distinguishing characteristic of this technique is that images are captured by moving the camera. That's why, the operation is performed by taking into account the camera location and orientation, and parameters. The most important step in camera pose estimation is to determine the intrinsic and extrinsic parameters of the camera. These parameters are obtained by calibrating the camera.

a) Camera Calibration

The cameras can be calibrated using two distinct methods, the first of which involves using the projection matrix and estimating the intrinsic and extrinsic parameters of the camera after applying the necessary filters and determining the pixel coordinates and world coordinates, and the second of which involves using the Caltech Camera Calibration Toolbox. In the following figure, it is possible to see the geometric logic behind the camera calibration process visually. Camera calibration is necessary for us since it provides to convert lengths that are in form of pixel to metric information.

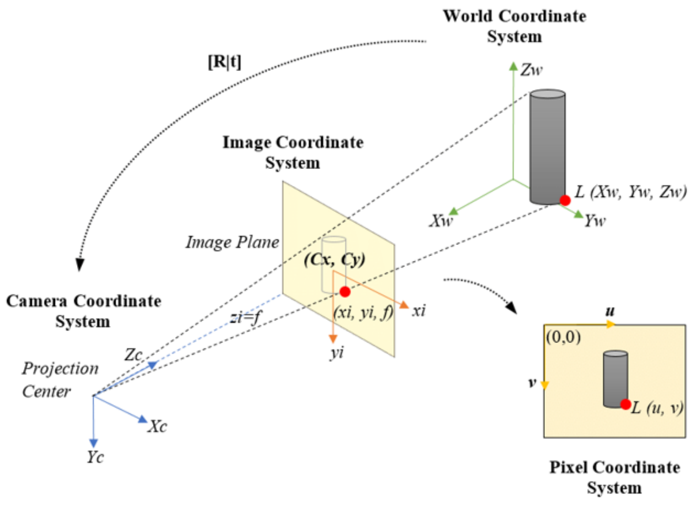


FIGURE 3.13: Camera Calibration Geometric Approach

As can be seen in the figure above, there are three different coordinate systems that are needed to deal with this. These coordinate systems are the World coordinate system shown by $[X_\omega; Y_\omega; Z_\omega]$camera coordinate system that is represented by $[X_C; Y_C; Z_C]$ and finally, the pixel coordinate frame which is more different than other two systems since it is a 2-dimensional system, that seen by $[u; v]$ In the algebraic relation between them, there exists extra columns and rows to make it matrix multiplications possible.

On the other hand, there are some internal and external parameters that affect the performance and working principle of the calibration operation. These parameters are represented inside the matrices which are called intrinsic and extrinsic parameter matrices. Consequently, the matrix is called a camera projection matrix.

As a first step of both of the operations, it is created a precise calibration rig (3D calibration object), and taken pictures of 2 checkerboards stuck on the wall perpendicular to each other. So, the obtained 3D checkerboard view is used to associate a world coordinate system with the calibration rig and determine the metric (in mm) world coordinates of all the corner points on the rig (both planes) using simple geometry. After taking pictures of the calibration rig, it is adjusted in an editor to get rid of its background since it may make noise and decrease the efficiency of the operation. For the calibration rig, 2 pieces 10x7 square where one edge of the square is 25mm, are used.

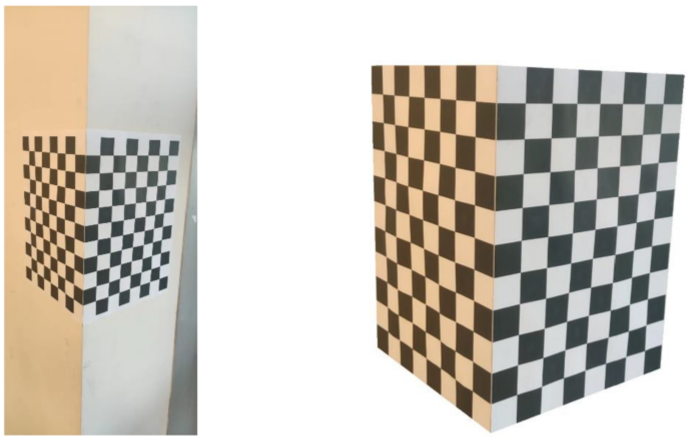


FIGURE 3.14: Pure Checkerboard

After that, it is defined a reference frame to clarify the relationship between the corners' pixel coordinate values and world coordinate values. However, in order to make this translation, it is also required to find camera coordinate system values with intrinsic and extrinsic parameters. Then it is applied the Harris Corner Detection Algorithm to detect the corners of both sides of used checkerboards.

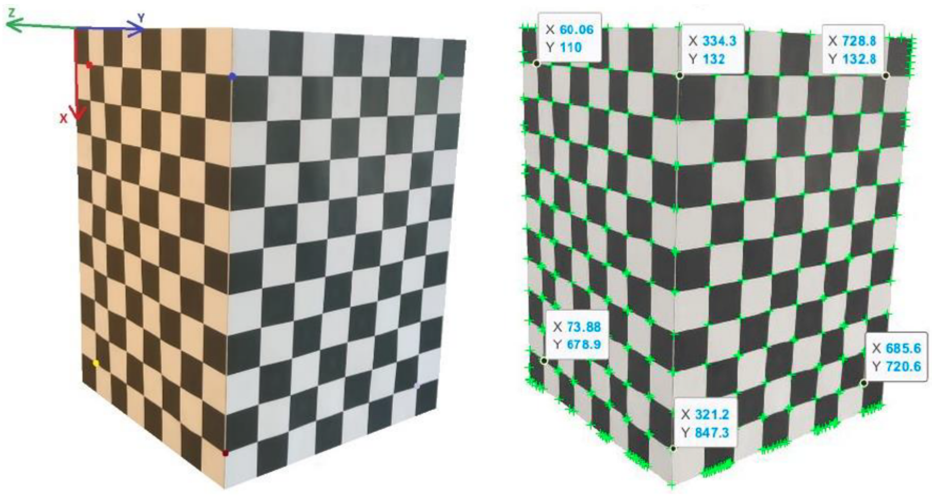


FIGURE 3.15: Assigned Coordinates on Checkerboard

Here, it is also important to mention that is selected 6 different corners to create at least 11 equations because of there are 11 DoF in intrinsic and extrinsic parameters as the details are explained in the following section. The pixel coordinates of the selected corners already can be seen in the figure above, but their real-world coordinates of them, calculated based on the length of a one-unit square edge, are given in the following table.

$P = (X_w, Y_w, Z_w)$	
$P_1 = (25, 25, 0)$	$P_2 = (25, 200, 0)$
$P_3 = (25, 200, -175)$	$P_4 = (250, 25, 0)$
$P_5 = (250, 200, 0)$	$P_6 = (250, 200, -175)$

FIGURE 3.16: SfM

In order to see the relation between the coordinates of a physical object in the pixel coordinate, it is necessary to find the intrinsic and extrinsic parameters. However, it is also pretty important to know which of them should be found first when there is no clear information regarding any exact knowledge except world coordinates and pixel coordinates. Before go more further, it is quite important to see the general matrix relation between these coordinates.

$$\begin{aligned} p &= K[R|T]P_\omega \\ p &= KP_c \\ P_c &= [R|T]P_w \end{aligned} \tag{3.6}$$

where K is intrinsic, R and T are extrinsic parameters matrices and they can be expressed like;

$$\gamma \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha & \theta & u_0 \\ 0 & \beta & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} \tag{3.7}$$

$$\gamma \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha & -\alpha cos\theta & u_0 \\ 0 & \beta/sin\beta & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} \tag{3.8}$$

where θ is the skew angle, and $\alpha = kuf$ and $\beta = kvf$ where ku and kv are sensor scales for the pixel-size in both of the dimensions while f is focal length.

The focal length of the camera can also be used to make transformations between the camera coordinate and pixel coordinates. However, in this question, since it is known the pixel and real-world coordinates of selected corners, the focal length can be found and go on to estimate other important parameters which are α, β, θ. The results can be demonstrated by using simple relation between camera and image coordinates;

$$u = f\tfrac{X_c}{Z_c} \quad v = f\tfrac{Y_c}{Z_c} \tag{3.9}$$

The extrinsic parameters of the camera can be represented in matrix form such as the following;

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} \tag{3.10}$$

where $[r_{11}...r_{33}]_{3x3}$rotation matrix and matrix and $[t_x; t_y; t_z]$ is transition matrix. When it is combined both of these parameter matrices in a single form, the following expression can be obtained:

$$\gamma \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha & -\alpha cos\theta & u_0 \\ 0 & \beta/sin\beta & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} \tag{3.11}$$

It is called the first two matrices as a single form matrix M, and then it is tried to estimate these 11 unknown parameters, where it is required to have at least 6 corner points that give 12 equations. At that point how much the number of equations is higher than the number of the unknown parameters, the solution matrix becomes that much more accurate. Here, when the matrix M is written into the equation:

$$\gamma \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & 1 \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} \tag{3.12}$$

$$M = \begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & 1 \end{bmatrix} \tag{3.13}$$

In this point, the values of the matrix M can be computed by using selected corners' world frame and pixel frame coordinates. During the process, the least square estimation method is used to estimate the calibration matrix. In light of this purpose, the equation that should be followed can be seen below:

$$QM = 0$$

At that level, firstly, it is performed SVD then QR factorization is made.

$$\underbrace{\begin{bmatrix} X_w^1 & Y_w^1 & Z_w^1 & 1 & 0 & 0 & 0 & 0 & -u_1X_w^1 & -u_1Y_w^1 & -u_1Z_w^1 & -u_1 \\ 0 & 0 & 0 & 0 & X_w^1 & Y_w^1 & Z_w^1 & 1 & -v_1X_w^1 & -v_1Y_w^1 & -v_1Z_w^1 & -v_1 \\ & & & & & & & \vdots & & & & \\ X_w^n & Y_w^n & Z_w^n & 1 & 0 & 0 & 0 & 0 & -u_nX_w^n & -u_nY_w^1 & -u_nZ_w^1 & -u_n \\ 0 & 0 & 0 & 0 & X_w^n & Y_w^n & Z_w^n & 1 & -v_nX_w^1 & -v_nY_w^1 & -v_nZ_w^1 & -v_n \end{bmatrix}}_{\text{Q matrix }\textbf{(known)}} \underbrace{\begin{bmatrix} m_{11} \\ m_{12} \\ m_{13} \\ m_{14} \\ m_{21} \\ m_{22} \\ m_{23} \\ m_{24} \\ m_{31} \\ m_{32} \\ m_{33} \\ m_{34} \end{bmatrix}}_{\text{M matrix }\textbf{(unknown)}} = \begin{bmatrix} 0 \\ 0 \\ . \\ . \\ . \\ 0 \\ 0 \end{bmatrix} \Rightarrow Q.M = 0$$

Once it has determined M, it is possible to recover the intrinsic and extrinsic parameters by remembering that:

$$M = K(R|T)$$

$$\begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \end{bmatrix} = \begin{bmatrix} \alpha_u & 0 & u_0 \\ 0 & \alpha_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \tag{3.14}$$

$$\begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \end{bmatrix} = \begin{bmatrix} \alpha r_{11} + u_0 r_{31} & \alpha r_{12} + u_0 r_{32} & \alpha r_{13} + u_0 r_{33} & \alpha t_1 + u_0 t_3 \\ \alpha r_{21} + v_0 r_{31} & \alpha r_{22} + v_0 r_{32} & \alpha r_{23} + v_0 r_{33} & \alpha t_2 + v_0 t_3 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \tag{3.15}$$

Then the camera matrix M can be represented by intrinsic and extrinsic parameters such as:

$$M = \begin{bmatrix} \alpha r_1^T - \alpha cot\theta r_2^T + u_0 r_3^T & \alpha t_x - \alpha cot\theta t_y + u_0 t_x \\ \frac{\beta}{sin\theta} r_2^T + v_0 r_3^T & \frac{\beta}{sin\theta} t_y + v_0 t_z \\ r_3^T & t_z \end{bmatrix} \tag{3.16}$$

where the expression in the first column consists 3 by 3 matrix depending on various α values like:

$$\rho \begin{bmatrix} \alpha_1^T \\ \alpha_2^T \\ \alpha_3^T \end{bmatrix} = \begin{bmatrix} \alpha r_1^T - \alpha cot\theta r_2^T + u_0 r_3^T \\ \frac{\beta}{sin\theta} r_2^T + v_0 r_3^T \\ r_3^T \end{bmatrix} \tag{3.17}$$

At the end of these operations, the matrix M is found as:

$$M = 1.0e+04 \begin{bmatrix} -0.0099 & 0.0186 & -0.0643 & 2.7839 \\ 0.0637 & -0.0165 & -0.0064 & 3.8648 \\ 0 & -0.0001 & 0 & 0.0369 \end{bmatrix}$$

Other than that, it is quite possible to achieve to intrinsic and extrinsic parameters of the camera. According to the matrix and the estimated values, the intrinsic parameters can be found as:

$$\theta = 1.4355 \qquad u_0 = 73.8925 \qquad \alpha = 666.5667$$

$$v_0 = 97.6980 \qquad \beta = 648.1119$$

Then the extrinsic parameters which are the rotation and the transition matrices can be found as:

$$R = \begin{bmatrix} 0.0007 & 0.3653 & -0.9309 \\ 0.9924 & -0.1148 & -0.0443 \\ -0.1231 & -0.9238 & -0.3626 \end{bmatrix} \qquad T = \begin{bmatrix} 1.4214 \\ 3.9961 \\ 368.8311 \end{bmatrix}$$

The error parameters became as:

$$error = 1.0e+3[0.50530.71961.51821.6097]$$

Thanks to the calibrated camera, pixel values can be converted to metric values. After that, unique features that are invariant under radiometric and geometric changes in the images taken are extracted and can be detected from different angles. Common features extracted from each input image using methods such as SIFT key points, and panorama stitching, are matched with feature matching operation.

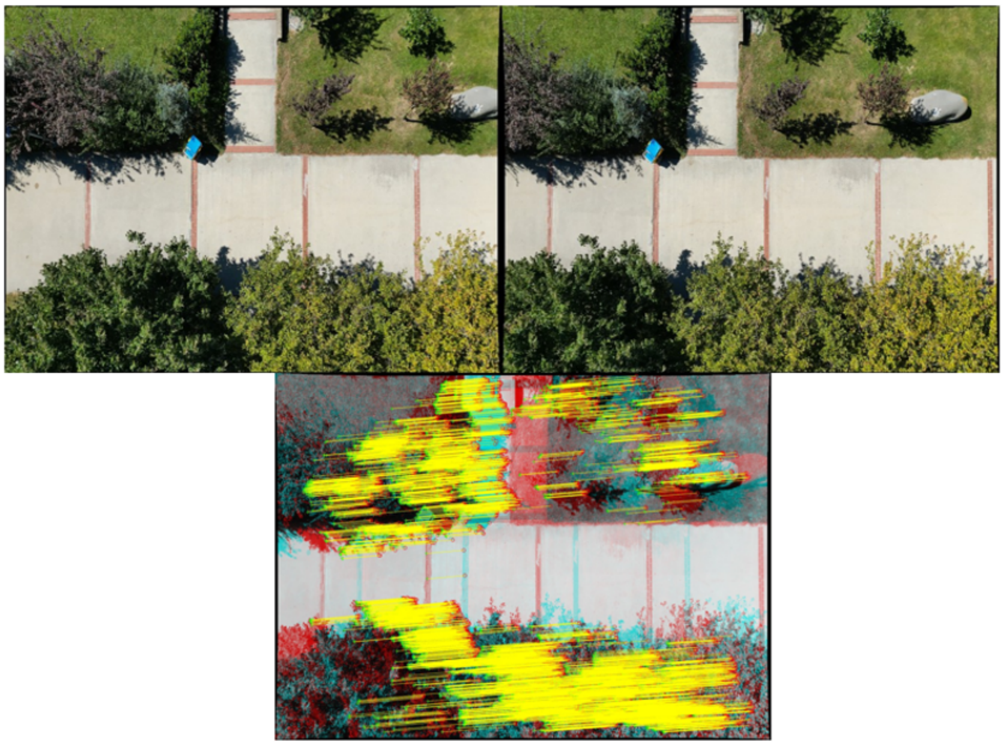


FIGURE 3.17: SfM

b) Pose Recovery of a Calibrated Camera Through Essential Matrix Using N-Point Algorithm

SfM differs from Stereo Vision in that it estimates the camera's position during processing. For camera pose estimation, the fundamental matrix and N-point algorithm are used. It is difficult to mathematically represent relative pose estimation because it is non-convex and notoriously plagued by local minima and ambiguous solutions. Uncalibrated and calibrated cameras are typically characterized for this purpose by a fundamental matrix and an essential matrix, which give the geometric relationship between selected features on input images.

Initially, it is necessary to identify the points that correspond to the same locations of the objects in the images but have different coordinates due to camera rotation and transitions. These parameters are also known as extrinsic parameters of the camera, where the rotation matrix and transition matrix each contain three unknowns for a total of six degrees of freedom. To solve these six unknowns, at least six different equations must be derived from at least six different corresponding points. However, sometimes in the literature the transition is taken in a pixel frame and its degree of freedom is reduced to 2 due to the fact that the pixel frame is used in 2D images. Due to the scale ambiguity of transition, calibrated cameras have five degrees of freedom (DoF) for the relative pose, including three for rotation and two for transition. With the exception of degenerate configurations, 5-point correspondences suffice to establish the relative pose. Given five-point correspondences, the five-point methods employing essential matrix or rotation matrix parametrization can efficiently determine the relative pose. The previously mentioned solvers are referred to as minimal solvers. When point correspondences contain outliers, minimal solvers are typically integrated into a hypothesize-and-test framework, such as RANSAC, in order to identify the solution corresponding to the maximal consensus set.

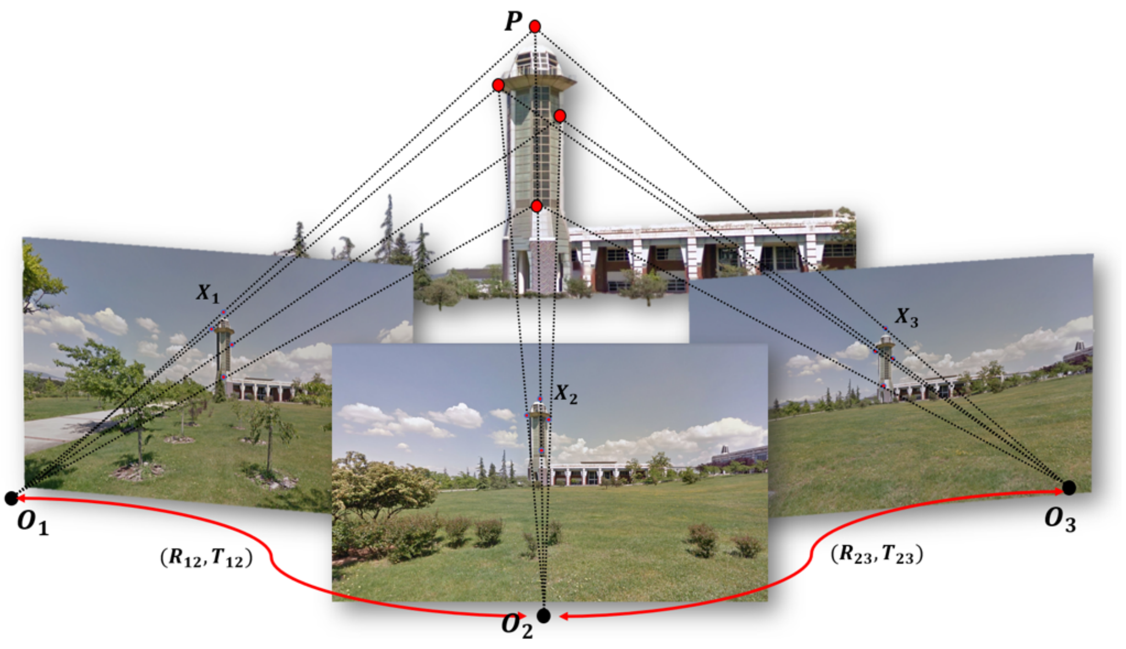


FIGURE 3.18: Geometric Relation

As can be seen given Figure 3.18, P represents a unique reference point which can be seen in all given camera positions O_1, O_2, and O_3. In this stage, triangulation is applied to Epipolar geometry. The pixel coordinates of the point P on images captured from various angles are represented by X_i and the rotation and transition between the camera locations are given as (R_{ij}, T_{ij}). After the scaling factor is taken into account the pixel coordinates of the selected feature P give an equation such as below:

$$\lambda_2 X_2 = R_{12}\lambda_1 X_1 + T_{12}$$

where the λ_2, λ_1, R,and T are unknowns and x_1 and x_2 are measurements. From this Euclidean transformation, it is possible to find the rotation, transition, and depth that reprojection error is minimized form with the following equation. This step is also called Bundle adjustment.

$$error = \sum_{k=1}^{n} ||X_1^k - \pi(R_{12}, T_{12}, X_2^k)||^2$$

In light of this knowledge, it is possible to represent the algebraic elimination of depth:

$$X_2^T \hat{T} R X_1 = 0$$

where the essential matrix E is equal to $\hat{T}R$. When the procedure followed up for the input images given in Figure 3.17, the obtained results can be seen as follows:

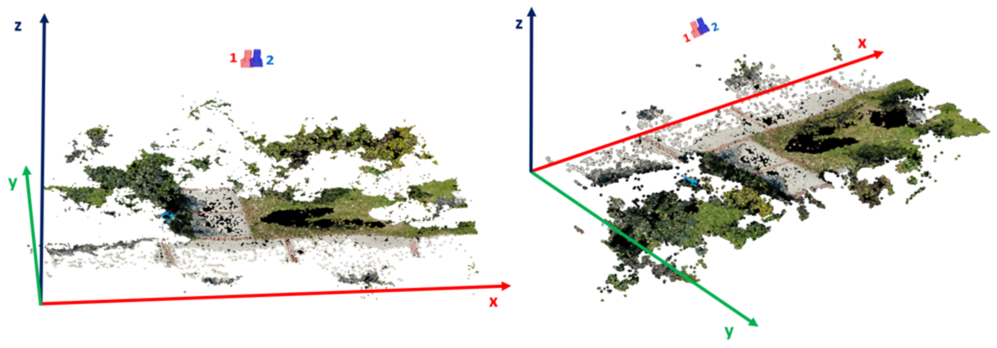


FIGURE 3.19: SfM Result

In addition, there is an additional technique known as multi-view stereo. This technique has characteristics of both stereo vision and SfM, as it attempts to create a 3D model from multiple images of an object or region. However, when SfM estimates camera location and orientation in addition to camera parameters, the multi-view stereo technique assumes that the intrinsic and extrinsic camera parameters are known from the start and creates a 3D cloud model.

3.2 Aerial Image Stitching by Using Feature-based Techniques

Aerial image stitching has been used to combine UAV image captures to enable path planning over a larger area. Various feature-based techniques were evaluated during the experiments.

Feature-based stitching techniques decode and match the similarities between the two images. After this matching, the unmatched features are eliminated and homography estimation is conducted. The target image's orientation is changed in order to merge with the reference image, and then the target image and reference image are merged. This progress can be expressed as:

$$L_{PW} = ||P(I^A) - P(H(I^B))|| \tag{3.18}$$

where I^A and I^B represent the full reference and target images, function $P()$ extracts an image patch from the image, and function $H()$ is used for warping the target image to align with reference image using estimated holography. There are numerous studies aimed at enhancing the efficacy of functions $P()$ and $H()$. In some studies, the images are scanned with patches of varying sizes, while in others, only half of the reference and target images are scanned, as the overlapping section is typically located in these regions. All of these methods, however, require a sufficient number of extracted image features in order to function properly. Therefore, it is crucial that images contain overlapping regions and distinct features that can be distinguished from the primary subject matter. During the experiments, images captured by an unmanned aerial vehicle (UAV) are stitched using feature-based techniques, and the outputs are obtained successfully with the exception of the following areas:

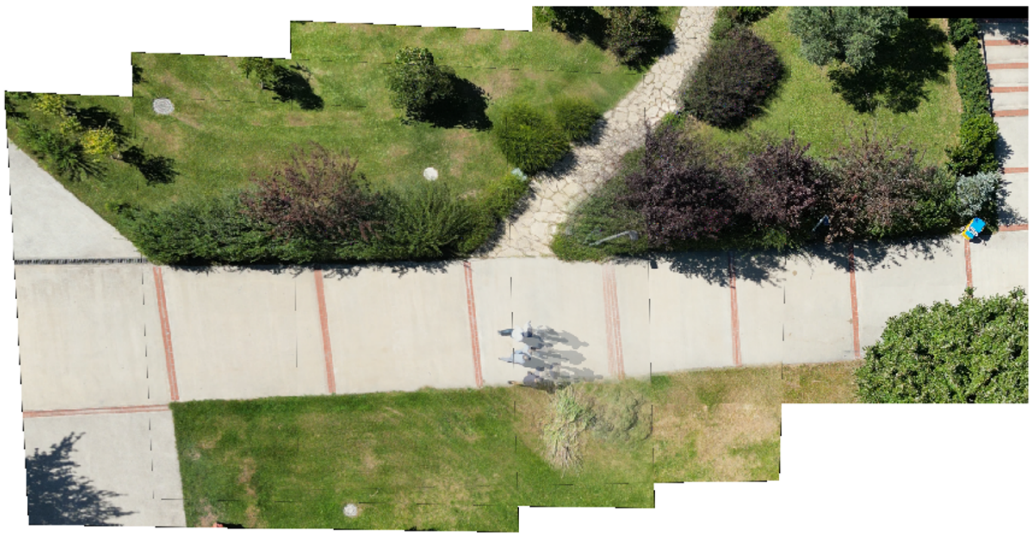


FIGURE 3.20: Feature-based Stitching Algorithm Output

In some instances, however, the feature-based stitching techniques do not function properly. These techniques have not been able to produce satisfactory results when the UAV is flying over a completely grassy area with a distinct background color, or when there is a small overlapping area and insufficient image features match.

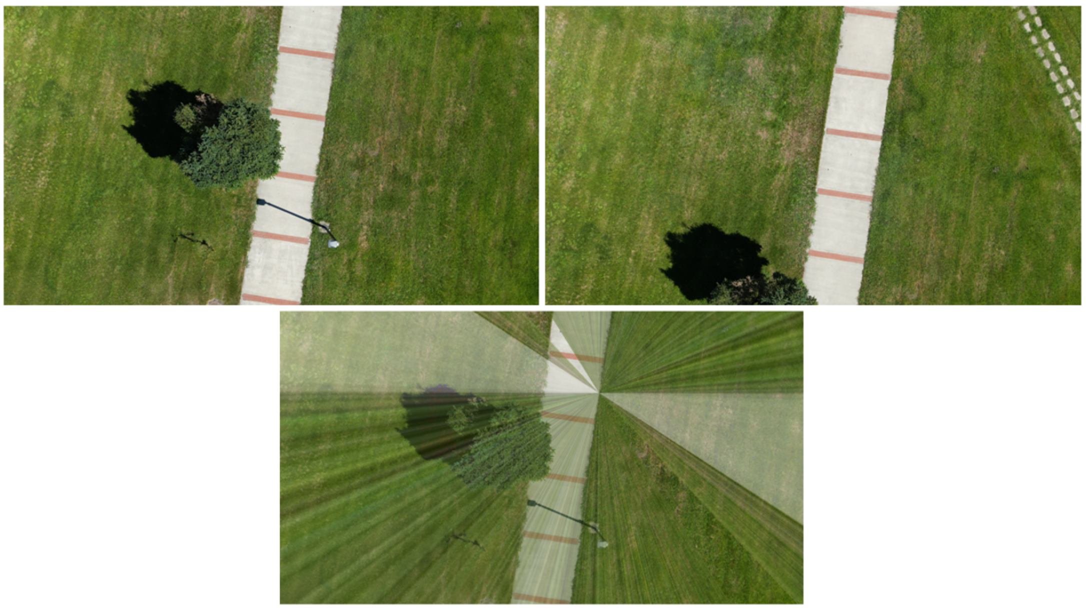


Figure 3.21: Fail of Feature-based Stitching Algorithm

As shown in the figure below, the feature-based image stitching algorithm identified the points of interest on the tree's shadow. Although there is a way to pass through large regions, the method matches the wrong features of the reference and target images because it was unable to identify any clear reference pairs that are distinct from the image's main basis.

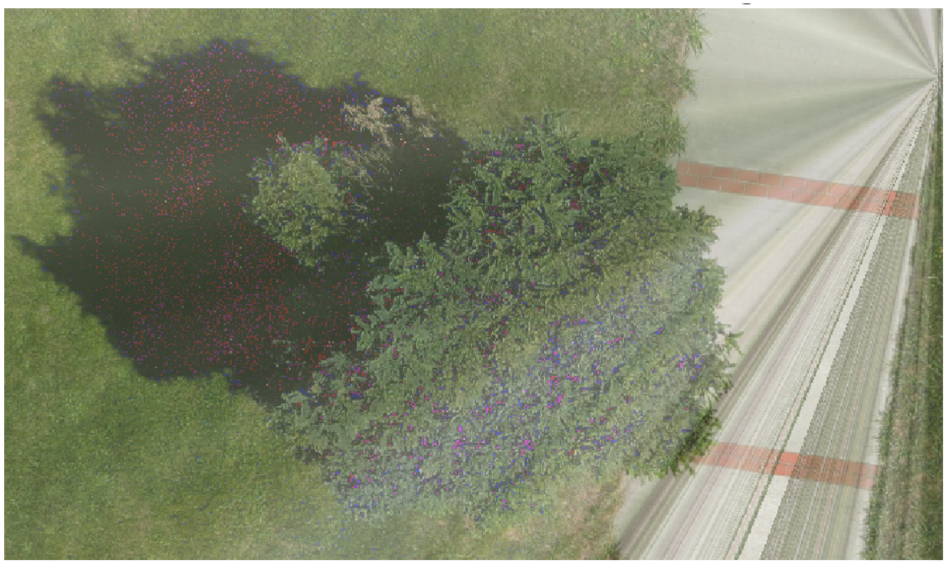


Figure 3.22: Assigned Key-points Representation

3.2.1 Aerial Image Stitching with IMU Data of a UAV

Image stitching techniques are examined under two main headings: feature-based methods and direct methods, as described in the preceding sections. In addition, the majority of published research focuses on feature-based techniques and their development. Regardless of the camera position, each of these techniques is accomplished by identifying unique key points in the image. It is crucial for image pairs to share a common area for this reason. In instances where the background is a single color, there is an insufficient number of unique reference key points, or there is insufficient overlapped space between image pairs, these algorithms do not function adequately and may produce inaccurate results.

In the experiments conducted for the thesis, a quadrotor equipped with an IMU takes images of the surface. Thanks to the IMU and the marker at the takeoff point, the quadrotor is able to record its position in real-time while taking images. In the proposed method, the relative positions of the images captured using these records were determined, and the images were then stitched. In this method, no image features were extracted as in feature-based techniques, nor were the pixel intensities of the images deciphered as in the direct method, nor was the similarity between the images investigated. Therefore, the proposed technique should be evaluated independently from these two primary topics.

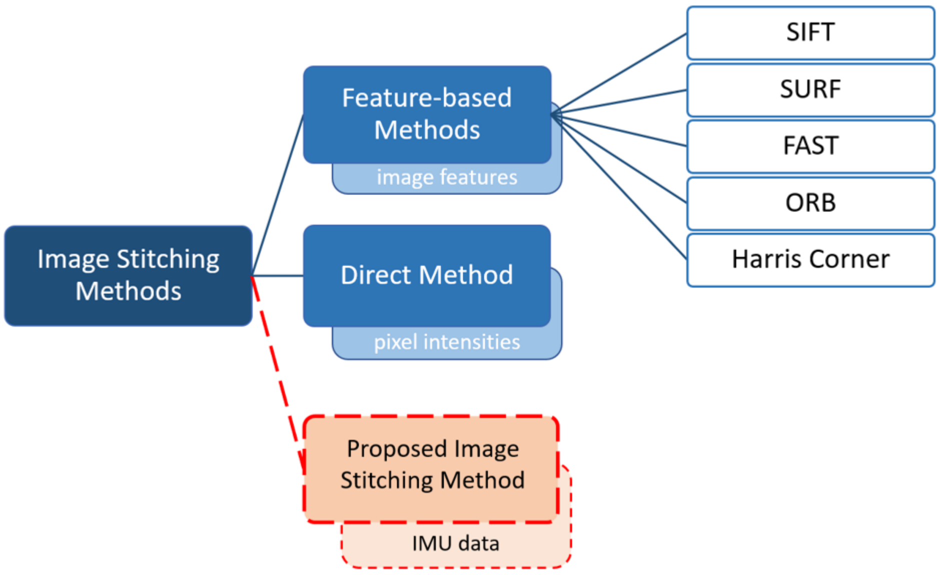


FIGURE 3.23: Proposed Technique for Image Stitching

The IMU data utilized by the proposed method includes position and attitude information for the quadrotor's x-y plan. The ultrasonic sensor located beneath the UAV provides information regarding its altitude. In addition, the UAV is equipped with a calibrated onboard camera, and both extrinsic and intrinsic parameters are known. In light of this information, the stitching procedure is carried out as follows:

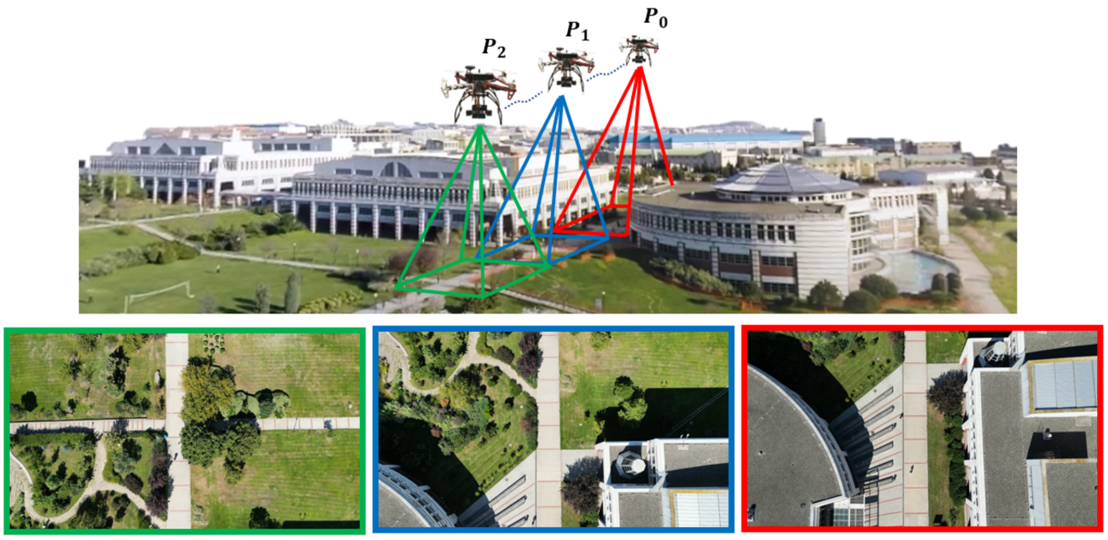


FIGURE 3.24: UAV Captured Images for Image Stitching

As shown in the figure above, images were captured from multiple points, including P_0, P_1, and P_2, and the position of the UAV at the time the images were captured was recorded. This data includes altitude, x-y position information, and yaw and pitch axis changes.

a) Z-axis Data – Altitude:

The scale of the received images is determined by the height of the UAV at the time the images were captured. Regardless of altitude, the UAV camera consistently captures images with the same resolution. This causes the image's objects to expand or contract as the height changes. If the UAV heights are different at P_k and P_{k+1} moments, this effect can be eliminated by changing the height and multiplying the intrinsic parameters of the image captured at a low altitude.

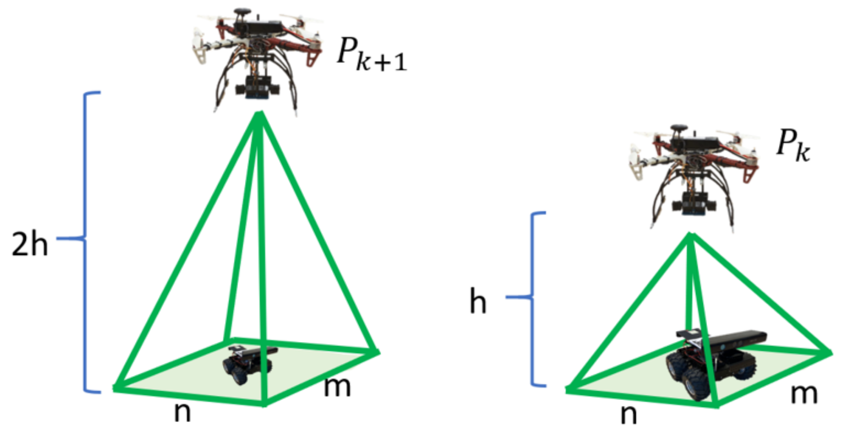


FIGURE 3.25: IMU Altitude Data Usage in Image Stitching

b) X-Y axes Data – XY Plane Position:

The distance between the waypoints where images are captured determines how far and in which direction the received images are Decoupled. This displacement amount is also multiplied by the camera's parameters and is measured in pixels. For this undertaking:

$$\begin{bmatrix} P_{x_t} \\ P_{y_t} \end{bmatrix} = K \begin{bmatrix} R|T \end{bmatrix} \begin{bmatrix} x_{P_{k+1}} - x_{P_k} \\ y_{P_{k+1}} - y_{P_k} \end{bmatrix} \tag{3.19}$$

where P_{x_t} and P_{y_t} represents the amount of the target image's offset.

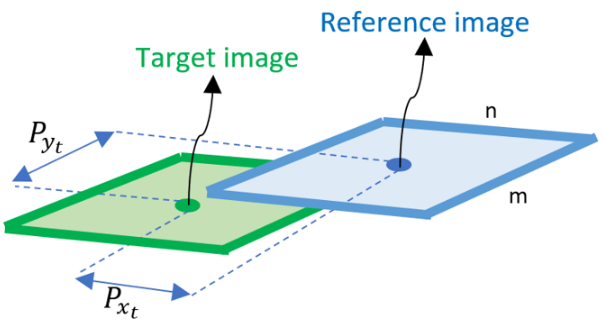


FIGURE 3.26: IMU X-Y Position Data Usage in Image Stitching

c) Yaw Axis Motion Effect:

The change in the yaw axis corresponds to the angle of the UAV's heading. Therefore, when the UAV turns in a different direction, the images it captures also rotate accordingly. If there is a difference in the yaw axis when capturing images at P k and P (k+1) points, the target image must be rotated on the z-axis by the same amount.

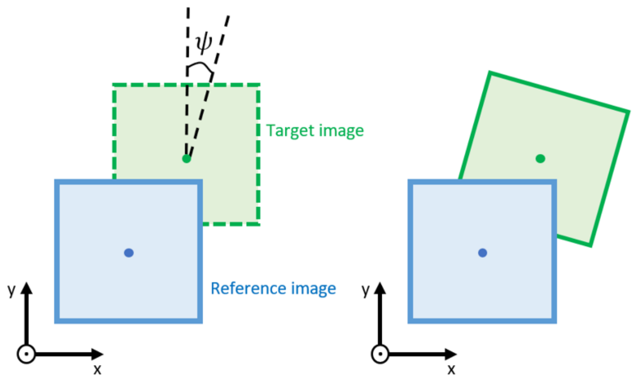


FIGURE 3.27: Yaw Axis Motion Effect on Image Stitching

d) Pitch Axis Motion Effect:

In some instances, the quadrotor may capture images while in motion. Therefore, when the gimbal is not in use, the image view lacks perspective. The diagram below depicts the differences in perspective of the same object in the same location when viewed from different pitch angles.

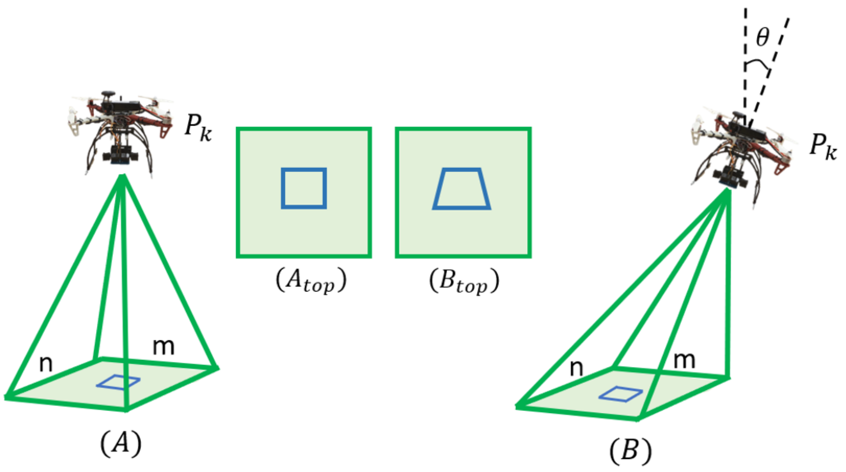


FIGURE 3.28: Pitch Axis Motion Effect on Image Stitching

In instances where the pitch angle is greater than zero, the captured images undergo a skew perspective transformation prior to processing. The resulting corrected image is used for stitching.

The images captured at the points P_0, P_1, and P_0 depicted in Figure 3.24 are stitched by following these steps in succession. The figure below shows the output of the proposed stitching technique:

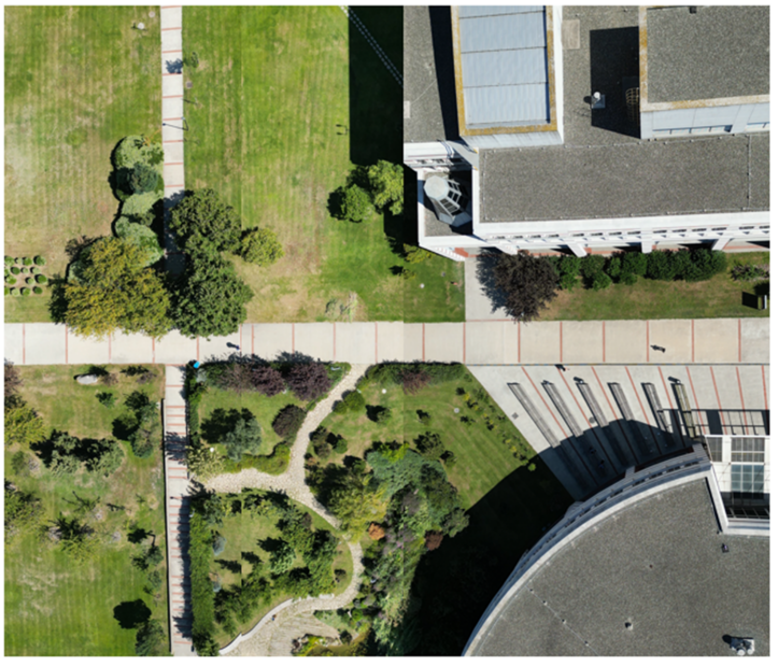


FIGURE 3.29: Result of Proposed Image Stitching Technique

Finally, Figure 3.30 shows the obtained result when the proposed method is applied to the images where the feature-based method stitched successfully as well. Moreover, as shown in Figure 3.30, the results obtained are promising, except for the parallax error caused by people walking into the frame while images are being taken. In addition, the edges of the images become smoother, which improves the image quality. Furthermore, Figure 3.31 shows the performance of the proposed stitching technique on the image pairs which couldn't be stitched by the feature-based technique. In cases where the feature-based technique failed, it is possible to observe that the proposed technique yields healthy results. The obtained results indicate that IMU data can be used to stitch images and may produce higher-quality results than feature-based techniques.

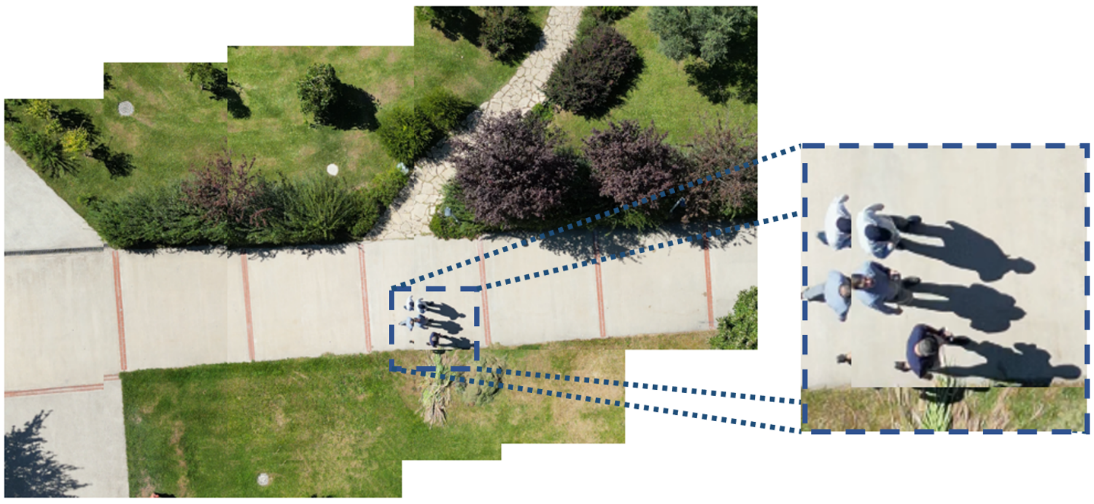


FIGURE 3.30: Result of Proposed Image Stitching Technique with Parallax Error

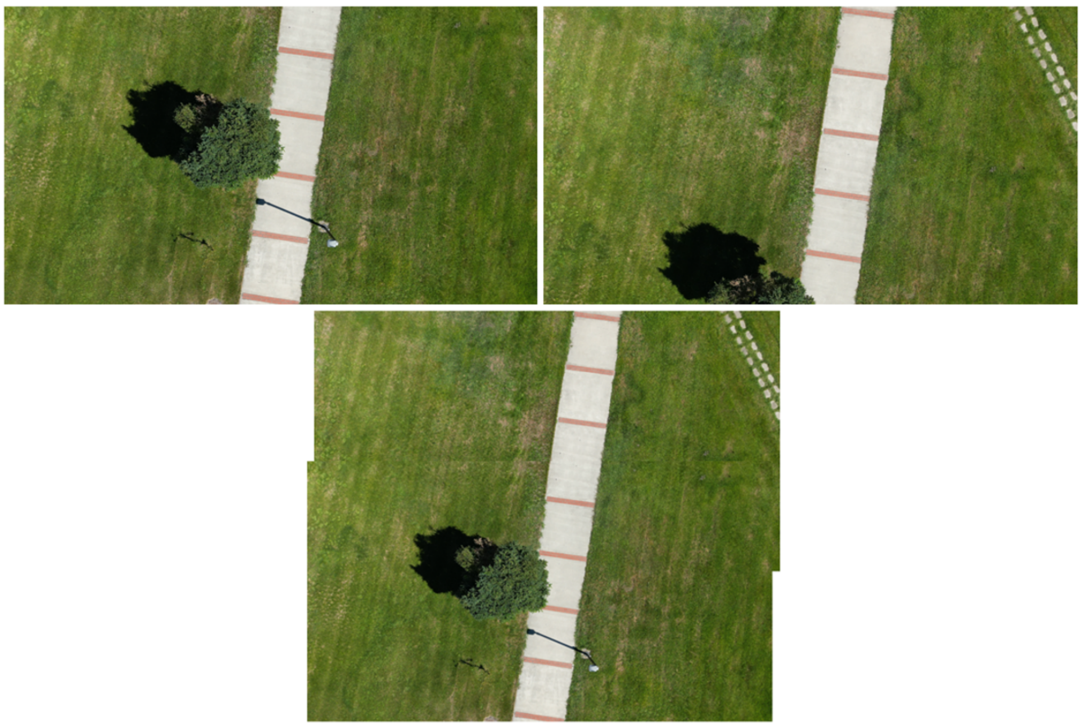


FIGURE 3.31: Result of Proposed Image Stitching Technique

Chapter 4

Image Based Motion and Path Planning for Heterogeneous Robot Systems Using Stereo Vision

4.1 Image-based Path Planning Algorithm Using a UAV Equipped with Stereo Vision

In this section, a novel image-based path planning algorithm [43] developed using computer vision techniques is described, along with its comparative analysis with well-known deterministic and probabilistic algorithms, namely A* and Probabilistic Road Map algorithm (PRM). The depth of the terrain has a substantial effect on the calculated path safety. In a two-dimensional image, craters and hills on the surface cannot be distinguished. The proposed method makes use of a UAV-created disparity map of the terrain. Several computer vision techniques, including edge, line, and corner detection methods, as well as the stereo depth reconstruction technique, are applied to the captured images, and the disparity map is used to determine the trajectory's candidate waypoints. The initial and desired points are detected automatically using the estimation of marker pose and circle detection techniques with the ArUco marker. After presenting the mathematical model and vision techniques, the developed algorithm is compared to well-known algorithms on various virtual scenes generated by the V-REP simulation program and a physical setup generated in a laboratory. The outcomes are encouraging and demonstrate the efficacy of the proposed algorithm.

Existing algorithms cannot provide an exact high-efficiency solution to the specified problem. Therefore, the primary motivation for this section is that there is no algorithm that finds the shortest and safest path in a shorter amount of time for a mobile robot on the ground by considering the features of the image form of the map and the depth of the terrain.

The primary objective is also to implement the concept using computer vision techniques alone. Since depth analysis is performed in this context using stereo vision, the terrain is displayed from multiple angles, and the detection rate of pits or hills that cannot be seen from a single angle is also increased.

In addition, one of the primary goals of developing a path planning algorithm based on computer vision and basic geometric rules is to ensure that the developed algorithm finds the path more quickly than deterministic and more logically (straighter) than probabilistic algorithms. The V-rep simulation environment has been used to create the virtual scenes that are studied in accordance with these objectives.

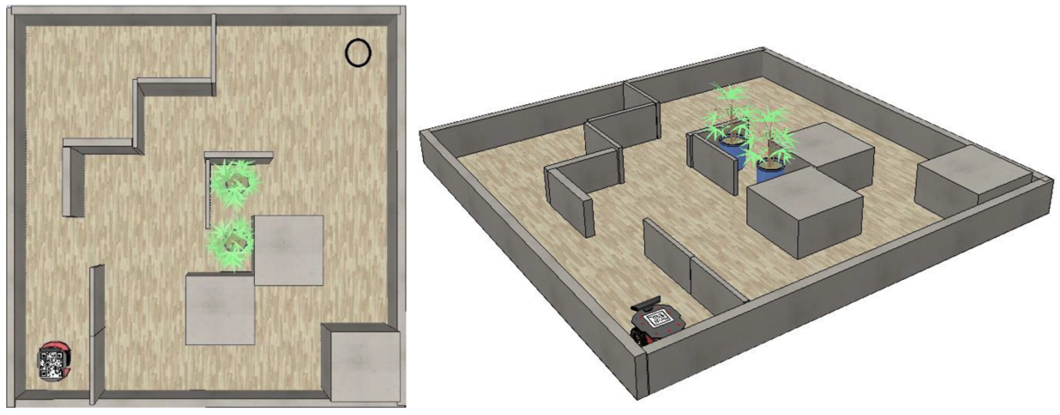


Figure 4.1: Virtual Scene 1 (V-rep)

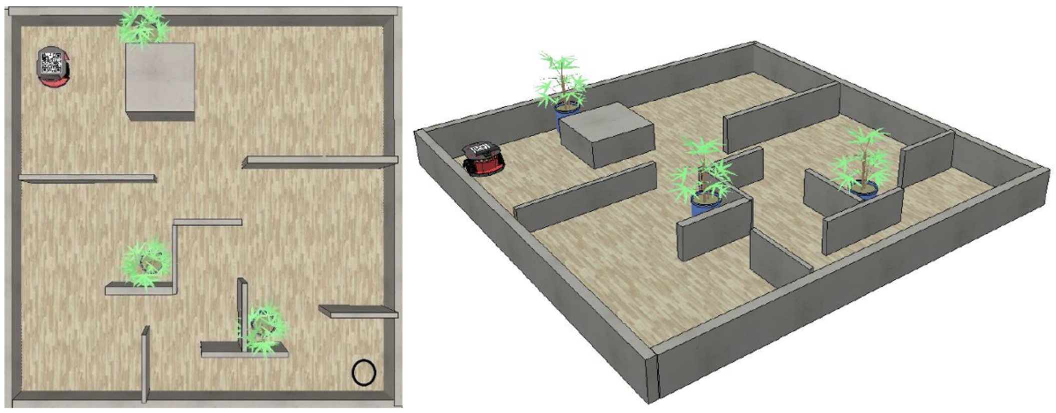


FIGURE 4.2: Virtual Scene 2 (V-rep)

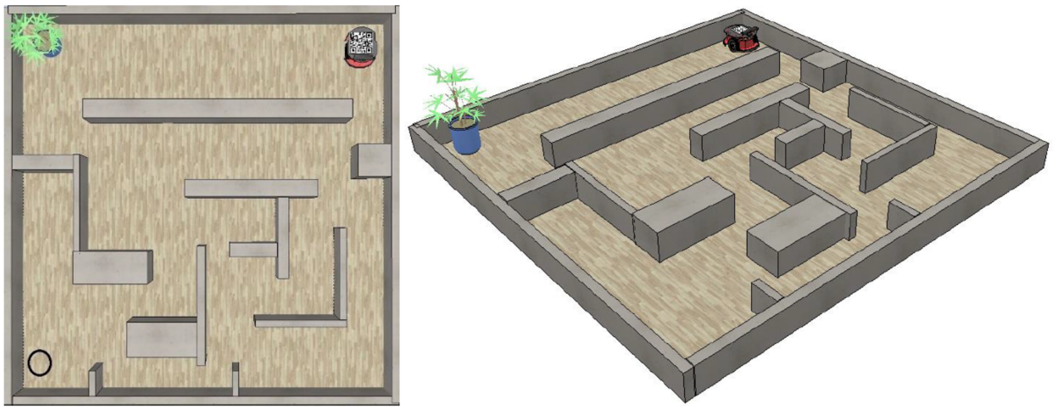


FIGURE 4.3: Virtual Scene 3 (V-rep)

In addition to these three simulation-based scenes, the developed novel algorithm and a selection of well-known pre-existing path planning algorithms are tested on a laboratory-created real scene. A DJI Tello quadrotor autonomously captures and transmits images of the actual scene to the base computer.

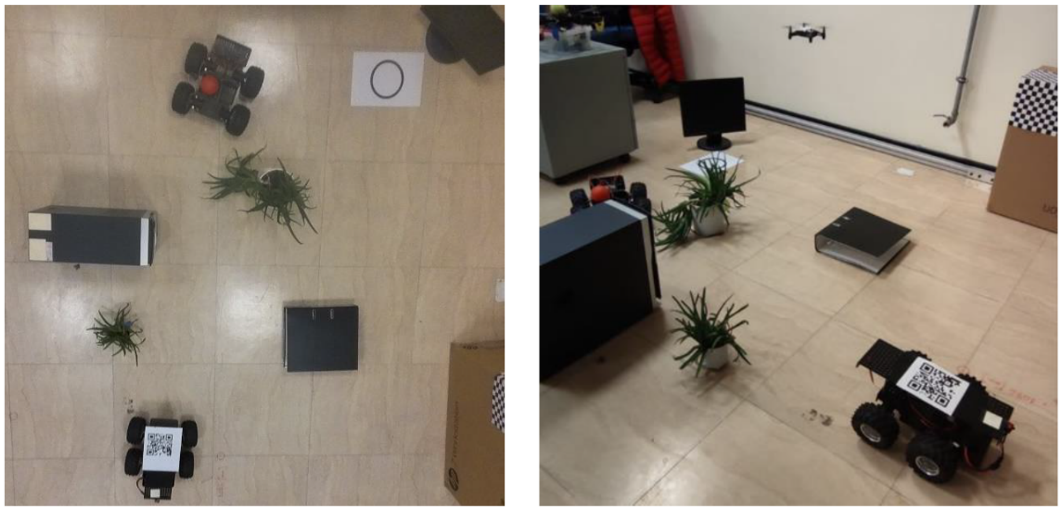


Figure 4.4: Actual Scene (CVR Lab)

4.1.1 Employed Computer Vision Techniques and Results

The developed algorithm utilized the following computer vision (feature extraction) techniques, and calculated the path accordingly.

a) Edge Detection

Edges appear at the boundary between two distinct image regions. These algorithms use a number of mathematical techniques to find edges, which are curves in a digital image where the image brightness abruptly changes or, more formally, contains discontinuities. One of the first order derivative edge detection filters Canny (1st order) is employed as a pre-processing step in this context due to the excessively noisy results produced by the corner and line detection methods. This filter was specifically chosen due to its smoothing property.

During the process, since it is faced with highly noisy results from the corner and line detection algorithms, it is implemented first and second derivative edge detection filters which are Prewitt (1st order), Sobel (1st order), and Laplacian of Gaussian (2nd order) as a pre-processing step.

Prewitt:	1st Derivative Filter	$filter_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$	$filter_y = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$
Sobel:	1st Derivative Filter	$filter_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$	$filter_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$
LoG:	2nd Derivative Filter	$filter_{xy=} \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	

FIGURE 4.5: Edge Detection Filters

These filters are specially selected since they have smoothing effects as well. The changes of the outcoming results on corner detection algorithm such as following:

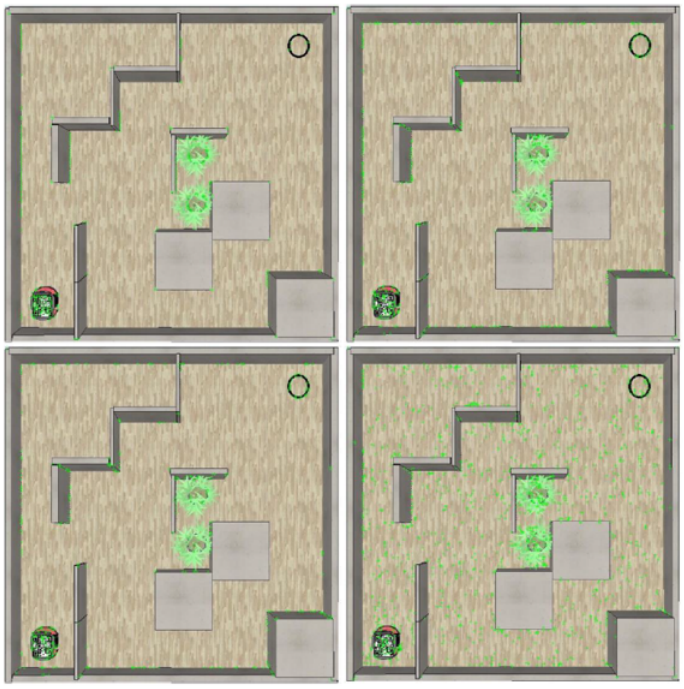


FIGURE 4.6: Corner Detection Algorithm Outputs: Left Top Image: Without Applying Any Filter, Right Top Image: After Applied Prewitt, Left Down Image: After Applied Sobel, Right Down Image: After Applied LoG

b) Corner Detection

The corner detection algorithms follow similar steps until the completion of the H matrix computation, which is a matrix whose elements are computed by different combinations of image gradients. The eigenvalues of H matrix and the f value, which can be computed as $f = det(H)/trace(H)$, are used in comparison with predefined thresholds. The FAST corner detection technique is one of the quickest feature extraction techniques, such as the difference of Gaussians (DoG) employed by the SIFT, SUSAN, and Harris detectors. In its operating principle, a 16-pixel circle, also known as the Bresenham circle, is utilized to classify the candidate point p that remains in the circle's center. Each pixel in the circle is labeled with an integer from 1 to 16 in a clockwise direction, and the corner point is identified by comparing the pixels of the two points. In this study, the FAST feature technique is favored because it produces corners with less noise than the other techniques.

In the scope of the project, it is applied three different corner detection algorithms which are Harris that is the most popular algorithm in this field, Kanade-Tomasi Technique which is also called Minimum Eigenvalue Algorithm, and lastly the FAST Features Method. These algorithms follow similar steps till the end of the H matrix computation that is a matrix whose elements compute by different combinations of image gradients as can be seen below:

$$H = \begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix} \tag{4.1}$$

where the I_x and I_y are the image gradients of a window along x and y directions, respectively. When these algorithms are applied to test scene 1, the outcoming results becomes as follows:

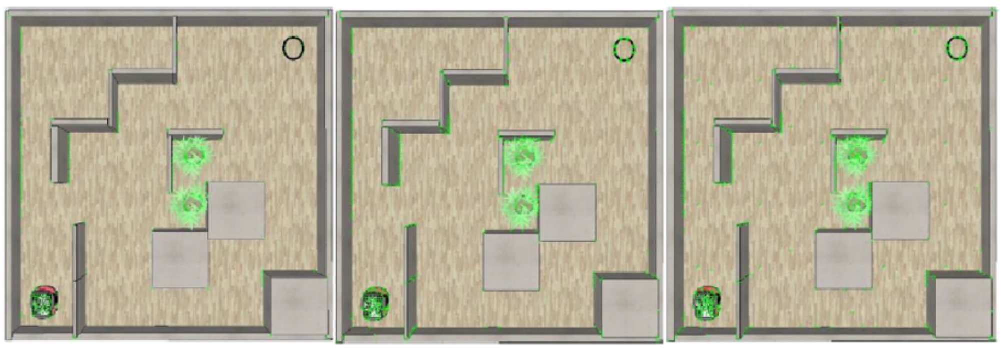


FIGURE 4.7: Corner Detection Algorithm Outputs-After Found the Best Threshold Values: Left Image: FAST Features, Middle Image: Harris Corner Detection, Right Image: Kanade-Tomasi Algorithm (MinEigenValue Theorem)

c) Line Detection

The line detection algorithm is one of the most crucial steps of the proposed method, as it clarifies the obstacle's borders. Despite the fact that the corners of the obstacles were identified in earlier steps, they provide no information regarding the boundary locations and directions. It is important to merge the corner points according to a rule in order to give them meaning. Due to the presence of several noisy corners in the input images, it is challenging to identify the right matches that define the obstacle edges. In this stage, the Hough line detector is therefore employed by specifying particular conditions for the head and tail nodes of the lines. In the implementation, nodes are assigned a distance condition, and the whole nodes inside the circular area whose radius smaller than the defined reference distance are merged.

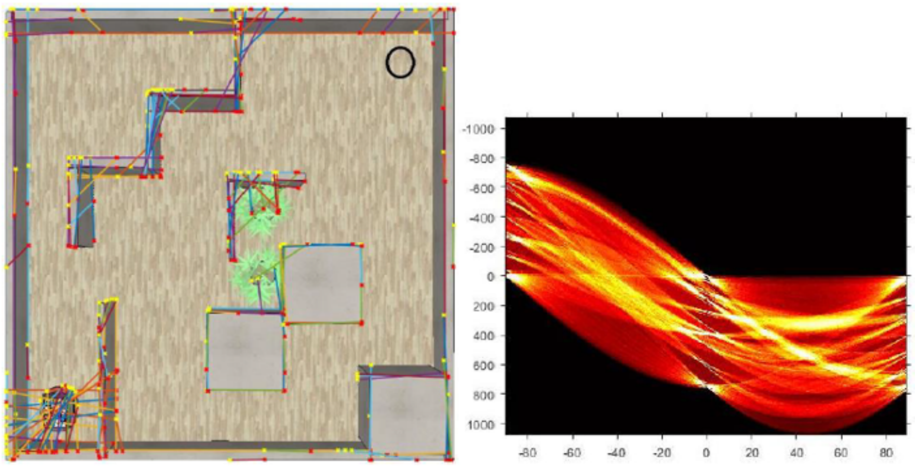


FIGURE 4.8: Hough Line Detection Algorithm Outputs

As can be seen figure above, the drawn lines clarified the obstacle boundaries successfully, although the features of the UGV are also included as an obstacle. At this point, it is possible to say that the image is segmented as terrain that the robot can move and obstacles that the robot should stay far away from. Thereby, at the end of the day, the line merging algorithm is used as also object recognition (without assigning the names of the obstacles on top of them) and image segmentation algorithms since the boundaries of the obstacles are defined and split from the region where the UGV can move.

d) Marker Detection

The ArUco marker is a synthetic square marker composed of a wide black border and an inner binary matrix that determines its identifier (id). The black border facilitates its fast detection in the image and the binary codification allows its identification and the application of error detection and correction techniques. The marker size determines the size of the internal matrix. For instance, a marker size of 4x4 is composed of 16 bits.

Figure 4.9: The Used Marker in the Virtual and Real Setups

In the developed algorithm, a marker is placed on top of the UGV to automatically detect the initial point without manual intervention. Since this technique provides both the position and orientation of the object, ArUco marker is chosen. Figure 4.10, depicts the initial point locations for all test scenes, which were determined using the marker mounted to the UGV. In these figures, the centroid of the detected marker is shown by a red dot, and the beginning point, which is denoted by a small blue square, is automatically found based on the red dot.

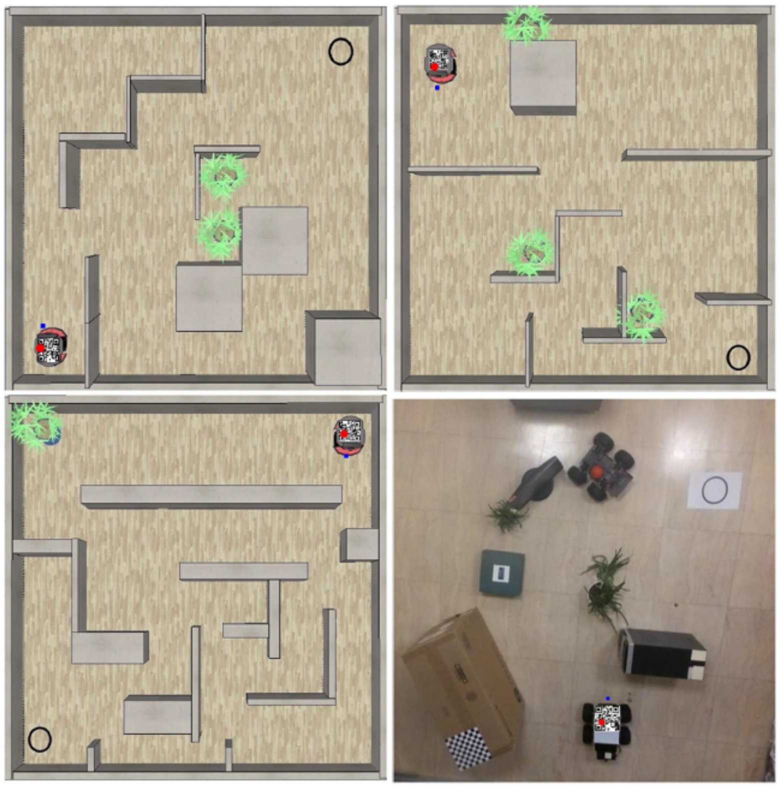


FIGURE 4.10: ArUco Marker Pose Estimation of UGV in All Test Scenes Left Top Image: Scene-1(V-rep), Right Top Image: Scene-2(V-rep), Left Down Image: Scene-3(V-rep), Right Down Image: Scene-4 (Real Setup)

e) Circle Detection

The Hough circle detection technique is used to locate the target point, which is likewise denoted by a circle in the scene similar to the usage goal of the marker detection method. The Circle Hough Transform (CHT) is a fundamental feature extraction technique used in digital image processing to detect incomplete circles in images. The candidates for circles are generated by “voting” in the Hough parameter space and then selecting local maxima from an accumulator matrix. A circle can be characterized in a two-dimensional space as $(x-a)^2 + (y-b)^2 = r^2$, where (a, b) is the circle’s center and r is its radius. In 3D space, the circle parameters can be identified via the intersection of several conic surfaces created by 2D circle points. This procedure consists of two steps. First, the radius is fixed, and then the optimal center of circles in a 2D parameter space is determined. In the second step, the ideal radius in one-dimensional parameter space is computed.

Since the algorithm can recognize another circular shape and assign it as a target point, the maximum/minimum radius of circle and sensitivity value are defined as hyperparameters:

$$R_{max} = 40 \qquad R_{min} = 10 \qquad Sensitivity = 0.9$$

The figure below illustrates the acquired outcomes.

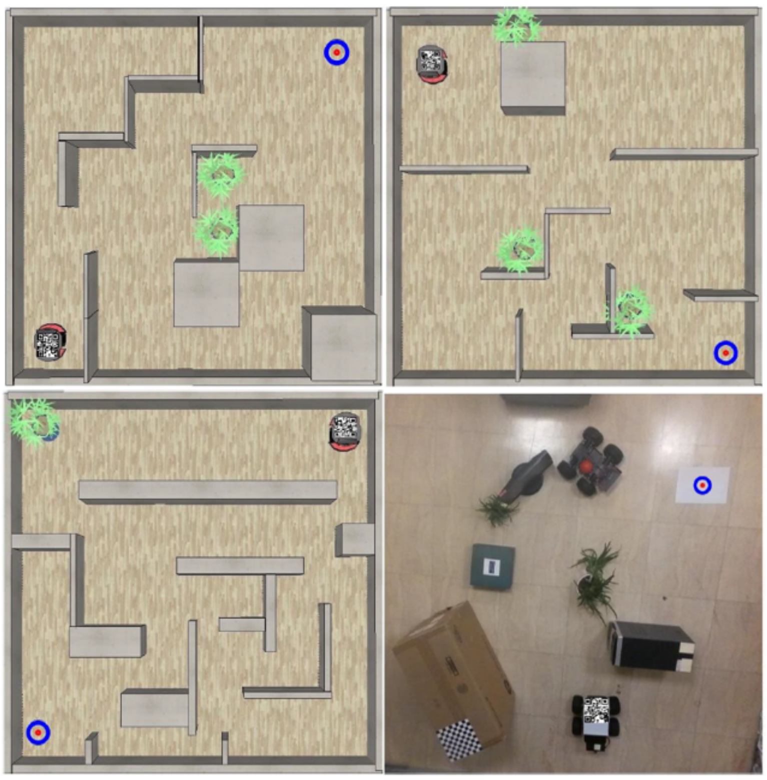


FIGURE 4.11: Circle Detection (Desired Point Detection) Left Top Image: Scene-1(V-rep), Right Top Image: Scene-2(V-rep), Left Down Image: Scene-3(V-rep), Right Down Image: Scene-4 (Real Setup)

f) Stereo Depth Reconstruction

The terrain's depth information is one of the pillars of the created path-planning algorithm. The stereo vision algorithm makes it possible to assess the trajectory's slope and choose a direction with a gentle gradient. The initial stage in determining depth from stereo vision is breaking the images into positive and negative patches by dividing the right and left images into patches. After the patches are separated according to their shapes, sizes, and relative geometries, the matching operation initiates. There are various ways for matching stereo images, however they can be grouped primarily into two main categories: area-based correlation techniques and feature-based techniques. In the correlation method, the right and left images are scanned pixel by pixel to identify relationships between the small rectangular portions. In contrast, feature-based approaches strive to detect a certain class of feature between two images. After matching pixels from two input images, all detected features are stored in a map known as a disparity map. If the camera lens parameters are known, such as focal length f and baseline distance T = 2l, the depth can be calculated from the disparity map. As can be seen in the following equations, the position and specifications of the stereo camera system, such as the properties of camera lens and the distance of the camera pairs, are essential for calculating depth. Additionally, it is vital to capture adequate light in an appropriate environment while avoiding reflection, transparency, and mirror surfaces for high-quality results.

$$\begin{gathered} d = X_L - x_R = f(\tfrac{x_p+l}{z_p}) - (\tfrac{x_p-l}{z_p}) = \tfrac{2fl}{z_p} \\ \Rightarrow z_p = \tfrac{2fl}{d} \end{gathered} \tag{4.2}$$

The hyperparameters of the stereo depth estimation algorithm have a significant impact on the output. These hyperparameters are referred as ω, which defines the color map pixel value interval, and the window size utilized for pixel scanning. High values of the window size affect the disparity map quality negatively, but the processing time decreases. During the experiments, the shadow effect on the results was minimized by comparing the two input images with different window widths. Figure 4.12 shows the disparity maps of the test scenes and the associated hyperparameter values. As seen in Figure 4.12, when the same hyperparameters are used, the disparity map of the actual scene does not seem as good as those of the other scenes. There are two primary causes for this problem. The first is the image pairs that have been used: Since the autonomous UAV that is used to capture images of the actual environment has a monocular onboard camera, the right and left images are taken after moving the UAV along an axis. The transition of the UAV is taken as a baseline and the disparity map is generated under this condition. However, since the image pairs in the stereo vision technique should be aligned in the same direction, the unanticipated axis shifts that occur during the transition of the UAV negatively influence the quality of the captured image pairs and disparity map. The second reason is the impact on the environment. Due to the fact that the surface of the terrain reflects the indoor lighting, a distinct group of artificial colors appeared on the image. Despite the fact that the depth estimate algorithm did not produce a satisfactory result in a real-world scenario, the generated path planning algorithm operates without any issue since it takes into account the corner spots discovered from the disparity maps.

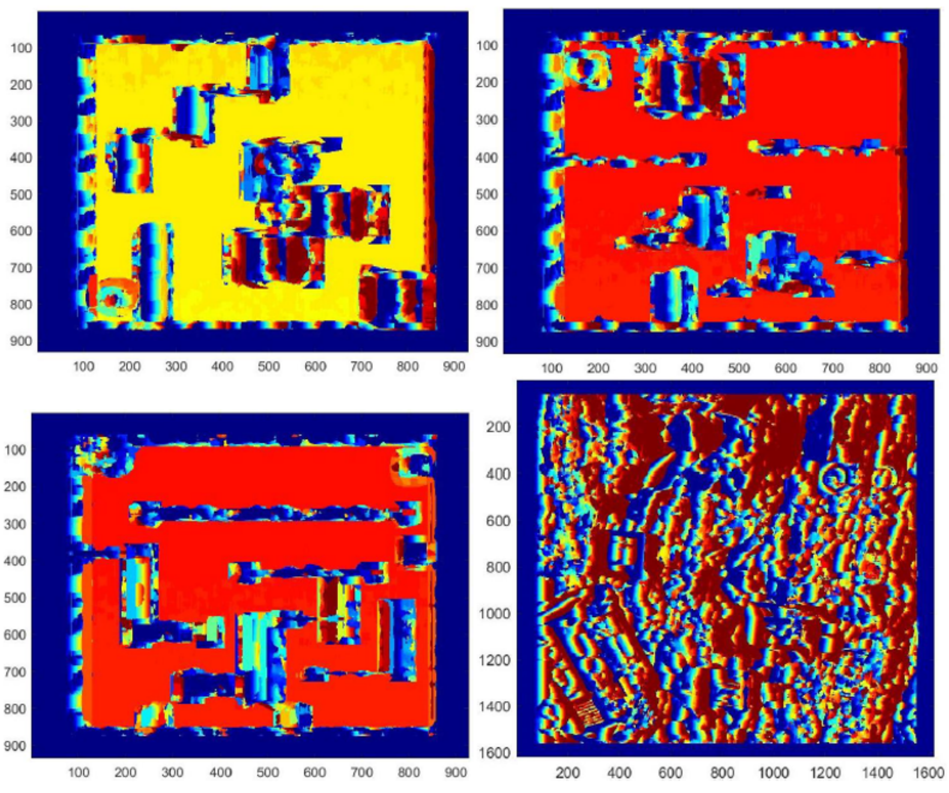


FIGURE 4.12: Disparity Maps of All Test Scenes $\omega = 50, WindowSize = 10$ Left Top Image: Scene-1(V-rep), Right Top Image: Scene-2(V-rep), Left Down Image: Scene-3(V-rep), Right Down Image: Scene-4 (Real Setup)

g) Line Intersection Algorithm

The line intersection method is the most essential aspect of the developed path planning algorithm. After locating the corner points that come from the terrain image and the disparity map, the best corner points should be identified as way-points for use in building a trajectory by connecting them with straight lines. It is also necessary to prevent any intersections with other lines that are drawn by the Hough line detection method to detect the obstacle edges in the terrain. In accordance with this approach, a line-drawing method is developed along with a condition to avoid line intersection scenarios by utilizing simple geometrical and analytical data. In Figure 4.13, a line intersection scenario is depicted. In this diagram, the black line indicates the shortest path between the initial and final positions when the line intersection requirement is deactivated. In addition, point P_1 represents a point on the trajectory, point P_2 indicates a location on one of the edge lines that intersects with the trajectory, and point P_3 represents the intersection point. Under these conditions, when the angles between the horizontal axis and the blue and black lines are referred to as θ_1 and θ_2, respectively, the line equations become:

$$L_1 = P_1 + \lambda_1 \begin{bmatrix} cos\theta_1 \\ sin\theta_1 \end{bmatrix}, \quad L_2 = P_2 + \lambda_2 \begin{bmatrix} cos\theta_2 \\ sin\theta_2 \end{bmatrix} \tag{4.3}$$

The intersection point P3 can be incorporated into the equations of L1 and L2:

$$P_3 = P_1 + \lambda_{1_{P_3}} \begin{bmatrix} cos\theta_1 \\ sin\theta_1 \end{bmatrix}, \quad P_3 = P_2 + \lambda_{2_{P_3}} \begin{bmatrix} cos\theta_2 \\ sin\theta_2 \end{bmatrix} \tag{4.4}$$

These equations can be solved simultaneously as:

$$P_1 + \lambda_{1_{P_3}} \begin{bmatrix} cos\theta_1 \\ sin\theta_1 \end{bmatrix} = P_2 + \lambda_{2_{P_3}} \begin{bmatrix} cos\theta_2 \\ sin\theta_2 \end{bmatrix} \tag{4.5}$$

Assuming the matrix is invertible, the equation can be expressed as follows:

$$\begin{bmatrix} \lambda_{1_{P_3}} \\ \lambda_{2_{P_3}} \end{bmatrix} = \begin{bmatrix} cos\theta_1 & -cos\theta_2 \\ sin\theta_1 & -sin\theta_2 \end{bmatrix}^{-1} (P_1 - P_2) \tag{4.6}$$

The coordinates of the intersection point can be determined by substituting one of the λ values found in preceding step into one of the line equations specified in the first step. In this stage, the λ values act an important role in the selection of the candidate way-points since their values between 0 and 1, refer to the intersection of the lines.

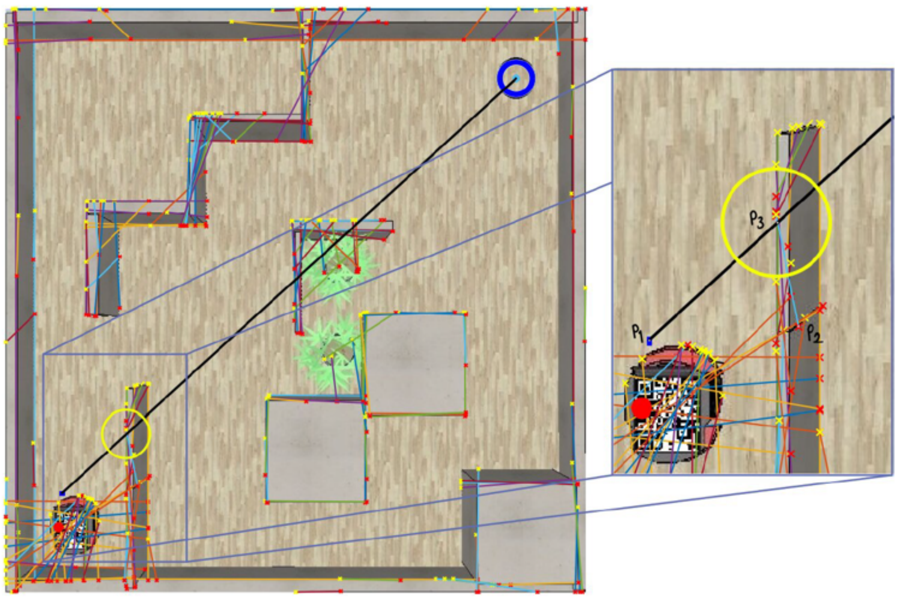
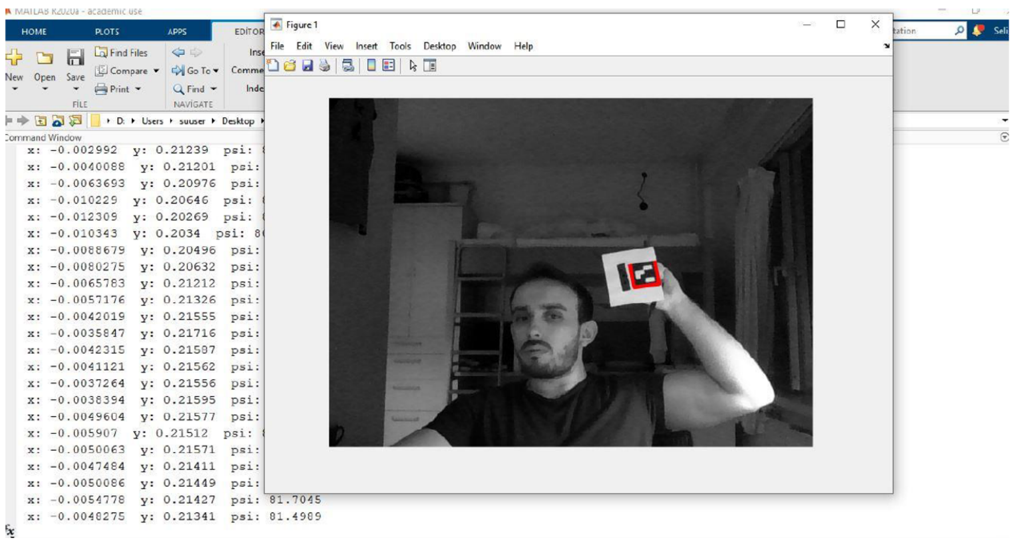


FIGURE 4.13: Line Intersection Case in Scene 1

4.2 System Identification of a Quadrotor Using Onboard Vision System

System identification is a method that uses the input and output signals of the system to construct mathematical model of dynamical systems. This method can find the mathematical model where the input and output data fit in the most appropriate way thanks to the linear and nonlinear models it contains. Moreover, during this process, it can take into account different numbers of input and output data, so it can also give satisfactory results on models with highly nonlinear and complex structure, such as quadrotor.

Thanks to system identification, modeling of Black-box systems can be performed, unknown parameters of grey-box systems can be estimated, or the controller designed for a dynamic system can be optimized. In the experiments carried out within the scope of this thesis, a black-box model was made using the onboard vision system, IMU data and motor speeds of the DJI Tello quadrotor and a separate grey-box model was made using the known system parameters. The position and attitude responses of the obtained models were compared using the same inputs to evaluate the consistency of the onboard vision system.



```
x: -0.002992  y: 0.21239  psi:
x: -0.0040088  y: 0.21201  psi:
x: -0.0063693  y: 0.20976  psi:
x: -0.010229  y: 0.20646  psi:
x: -0.012309  y: 0.20269  psi:
x: -0.010343  y: 0.2034  psi: 8
x: -0.0088679  y: 0.20496  psi:
x: -0.0080275  y: 0.20632  psi:
x: -0.0065783  y: 0.21212  psi:
x: -0.0057176  y: 0.21326  psi:
x: -0.0042019  y: 0.21555  psi:
x: -0.0035847  y: 0.21716  psi:
x: -0.0042315  y: 0.21587  psi:
x: -0.0041121  y: 0.21562  psi:
x: -0.0037264  y: 0.21556  psi:
x: -0.0038394  y: 0.21595  psi:
x: -0.0049604  y: 0.21577  psi:
x: -0.005907  y: 0.21512  psi:
x: -0.0050063  y: 0.21571  psi:
x: -0.0047484  y: 0.21411  psi:
x: -0.0050086  y: 0.21449  psi:
x: -0.0054778  y: 0.21427  psi: 81.7045
x: -0.0048275  y: 0.21341  psi: 81.4989
fx
```

FIGURE 4.14: Marker

4.2.1 Quadrotor Modeling

Quadrotors are highly nonlinear and underactuated systems since although there are just 4 control inputs which are speeds of motors, they are able to move 6 degrees of freedom which are 3 translational and 3 rotational axes. Due to the complex structure when they move up on just the vertical axis without changing other position states, their attitude states change since it is not possible to control 6 degrees of freedom with only 4 control inputs which are u(1), resulting in the thrust of the four rotors; u(2), roll angle change so the difference of thrust between the motors on the x-axis; u(3), thrust angle change so the difference of thrust between the motors on the y axis; and lastly u(4), the difference between the clockwise and counterclockwise rotors.

As mentioned in the previous paragraphs, the quadrotors are highly nonlinear systems and many various external and structural disturbances affect the dynamics of the system such as aero-elastic effects, internal dynamics of the engines, and flexibility of the wings. Since the disturbances make the equations hard to solve and they are generally neglected by assuming as external noises that can be compensated by the designed controller. However, since the system is still pretty complex and highly nonlinear even after neglecting disturbances, sometimes some other order reduction operations can be preferred by designers because of 12 different states which are wanted to control which are x-y-z positions, orientation vector (roll-pitch-yaw), rotational speeds (p, q, r), and linear accelerations. Hence, it is necessary to make some assumptions to arrive at the systemic equations such as the following:

• The quadrotor structure is rigid and symmetrical according to the center of mass.

• The thrust and drag of each motor are proportional to the square of the motor velocity.

• The propellers are rigid and blade flexibility is neglected.

• External disturbances are neglected.

As noticed the equation of motion of the quadrotors occurs from various sub-analyses. Translational kinematics is the first of them. It is necessary to define a rotation matrix to transform variables between global and local coordinate axes, and the matrix can be computed as follows:

$$x_b = R_G^b X^G = R(\phi)R(\theta)R(\psi)X^G$$

$$R(\psi) = \begin{bmatrix} c\psi & s\psi & 0 \\ -s\psi & c\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad R(\theta) = \begin{bmatrix} c\theta & 0 & -s\theta \\ 0 & 1 & 0 \\ s\theta & 0 & c\theta \end{bmatrix} \quad R(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c\phi & s\phi \\ 0 & -s\phi & c\phi \end{bmatrix}$$

$$R_G^b = \begin{bmatrix} c\psi c\theta & s\psi c\theta & -s\theta \\ c\psi s\phi s\theta - c\theta s\psi & s\psi s\phi s\theta & c\theta s\phi \\ c\phi c\psi s\theta & s\psi c\phi s\theta & c\theta c\phi \end{bmatrix}$$

$$X^G = R_b^G x^b = R(\phi)^T R(\theta)^T R(\psi)^T x^b = (R_G^b)^T x^b$$

$$R_b^G = \begin{bmatrix} c\psi c\theta & c\psi s\pi s\theta - c\phi s\psi & c\phi c\psi s\theta + s\phi s\psi \\ c\theta s\psi & s\psi s\phi s\theta + c\phi c\psi & s\psi c\phi s\theta - s\phi c\psi \\ -s\phi & s\phi c\theta & c\theta c\phi \end{bmatrix}$$

(4.7)

where X^G, Y^G, Z^G are the global North, East, and Down position of the quadrotor; x^b, y^b, z^b are the local North, East and Down positions of the quadrotor in the body frame; ϕ, θ, ψ are roll, pitch, and yaw angle; and p, q, r are roll, pitch and yaw rates respectively.

Furthermore, the rotational kinematics, which represents the angular rates between body frame and Euler angles that are defined in the middle of coordinate frames, are important as much as the transitional kinematics since the calculations of p, q, r states come from here:

$$\begin{gathered}
\omega = \begin{bmatrix} p \\ q \\ r \end{bmatrix} = R(\phi)R(\theta)\begin{bmatrix} 0 \\ 0 \\ \dot{\psi} \end{bmatrix} + R(\theta)\begin{bmatrix} 0 \\ \dot{\theta} \\ 0 \end{bmatrix} + \begin{bmatrix} \dot{\phi} \\ 0 \\ 0 \end{bmatrix} = S\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \\
S^{-1} = \begin{bmatrix} 1 & s\phi t\theta & c\phi t\theta \\ 0 & c\psi & s\phi \\ 0 & \frac{s\phi}{c\theta} & \frac{c\phi}{c\theta} \end{bmatrix} \\
\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & s\phi t\theta & c\phi t\theta \\ 0 & c\psi & s\phi \\ 0 & \frac{s\phi}{c\theta} & \frac{c\phi}{c\theta} \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix}
\end{gathered} \tag{4.8}$$

After all these kinematics computational analyses, the motion equations of the motors are important to go one more step ahead. The quadrotors are a member of the electro-mechanical systems family and have four rotors whose two-diagonal turn CCW and the others in opposite directions. Hence, there is a relationship between the force and torque created by the motors. Therefore, it is necessary to find the relation between these two terms in terms of current and voltage.

$$\begin{gathered}
\tau = K_t(I - I_o) \\
V = IR_m + K_v\omega \\
P_m = \frac{(\tau + K_t I_o)(R_t I_o R_m + \tau R_m + K_t K_v \omega)}{K_\tau^2}
\end{gathered} \tag{4.9}$$

where τ is motor torque, K_τ is torque constant, I, I_0, V, R_m are the input current, current without load, voltage that feeds motor, and resistance of the motors respectively. Also, $K_v, \omega and P_m$ represent the back EMF coefficient, angular velocity of motor and motor power.

For simplification, the motor resistance and no-load current assumed as zero and the relation between power and torque can be represented as equation below:

$$\begin{gathered}
R_m \approx I_o \approx 0 \\
P_m = \frac{K_v \tau \omega}{K_t}
\end{gathered} \tag{4.10}$$

Herein, it is quite important to clarify that all these calculations are referring to optimum usage style, which is hover position, of the quadrotor; since the power consumption and the performance of the motors are directly and easily affected according to the ordered high performance required missions. From these equations, it is easier to describe the equation which gives the relation between motor speed and resultant force of thrust where $P_h, T, V_h, \rho, AandK_T$ represent power to hover, thrust force to hover, velocity at hover, air density area swept by the propellers and thrust coefficient respectively.

$$\begin{aligned} P_h &= Tv_h \\ T &= 2\rho A v_h^2 \\ \tau &= K_\tau T \end{aligned} \tag{4.11}$$

$$\begin{aligned} P_m &= \tfrac{K_v \tau \omega}{K_t} = \tfrac{K_v K_\tau T \omega}{K_t} \\ P_h &= \tfrac{T^2}{\sqrt{2\rho A}} \\ T &= [(\tfrac{K_v K_\tau \sqrt{2\rho A}}{K_t})\omega]^2 \\ T &= K_T \omega^2 \end{aligned} \tag{4.12}$$

where m is the mass of the quadrotor, g is gravity acceleration, F_g is a gravity force in the global frame, F_d is the drag force, F_T^G and F_T^b are total thrust force in the global frame and body frame, R_b^G is rotation matrix of the body to the global frame, F_i is thrust force of each motor, and finally, K_{dx}, K_{dy}, K_{dy} are the drag coefficients according to coordinate axes.

Furthermore, in order to achieve the equation of motion of a quadrotor, it is necessary to derive transitional and rotational dynamics equations as well. The calculations up until this part were standard for almost all different types of systems. However, the dynamics equations are directly bonded to the physical specifications of the systems. There are some new notations that are used during these computational analyzes such as the following:

$$
\begin{gathered}
m\ddot{X}^G = F_g - F_T^G - F_d \\
\ddot{X}^G = \begin{bmatrix} \ddot{X}^G \\ \ddot{Y}^G \\ \ddot{Z}^G \end{bmatrix} \qquad F_g = \begin{bmatrix} 0 \\ 0 \\ mg \end{bmatrix} \\
F_d = \begin{bmatrix} K_{dx} & 0 & 0 \\ 0 & K_{dy} & 0 \\ 0 & 0 & K_{dz} \end{bmatrix} \begin{bmatrix} \dot{X}^G \\ \dot{Y}^G \\ \dot{Z}^G \end{bmatrix}
\end{gathered}
\tag{4.13}
$$

$$
\begin{gathered}
F_T^G = R_b^G F_T^b \qquad F_T^b = \sum_{i=1}^4 F_i \\
F_T^G = R_b^G \sum_{i=1}^4 F_i = \begin{bmatrix} 0 \\ 0 \\ K_\tau \sum_{i=1}^4 \omega^2 \end{bmatrix}
\end{gathered}
\tag{4.14}
$$

where J_b, J_m, J_r, J_x, J_y, J_z are moment of inertia of quadrotor, motor, rotor, quadrotor body in $x_b, y_b, andz_b$ respectively. Furthermore, τ_g is gyroscopic effect torques, R is propeller radius, and l is length of motor arm.

After derivation of the transitional dynamics, the rotational dynamics which is the rotational equation of motion which represents the motion between the center of the quadrotor and not the center of the global frame different than the previous computations. The calculation steps of the rotational dynamics are relatively harder than the other operations since this equation is comprised of three terms including aerodynamic effects and motor torques. In order to try to achieve the result, it is required to take into account the gyroscopic effects and rolling, pitching, and yawing torques. Also, the drag force should be taken into account as well, but the force can be neglected sometimes for simplicity.

$$
\begin{gathered}
J_b\dot{\omega} = \tau_m - \tau_g - \omega J_b \omega \\
J_b = \begin{bmatrix} J_x & 0 & 0 \\ 0 & J_y & 0 \\ 0 & 0 & J_z \end{bmatrix} \; \dot{\omega} = \begin{bmatrix} \ddot{\phi} \\ \ddot{\theta} \\ \ddot{\psi} \end{bmatrix} \; (\omega J_b \omega) = \begin{bmatrix} \dot{\theta}\dot{\psi}(J_x - J_y) \\ \dot{\phi}\dot{\psi}(J_x - J_z) \\ \dot{\theta}\dot{\phi}(J_y - J_z) \end{bmatrix}
\end{gathered}
\tag{4.15}
$$

$$\begin{gathered} J_m\dot{\omega} = \tau_m - \tau_\psi \qquad \dot{\omega} = 0 \\ \tau_\psi = \tfrac{1}{2}R\rho C_D A(\omega R)^2 = K_d\omega^2 \\ \tau_m = \begin{bmatrix} \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{bmatrix} = \begin{bmatrix} lK_t(\omega_4^2 - \omega_2^2) \\ lK_t(\omega_1^2 - \omega_3^2) \\ K_d(\omega_1^2 - \omega_2^2 + \omega_3^2 - \omega_4^2) \end{bmatrix} \\ \tau_g = \begin{bmatrix} \dot{\theta} \\ -\dot{\phi} \\ 0 \end{bmatrix} J_r \textstyle\sum_{i=1}^{4}(-1)^{i+1}\omega_i = \begin{bmatrix} J_r\dot{(\theta)}(\omega_1 - \omega_2 + \omega_3 - \omega_4) \\ -J_r\dot{(\phi)}(\omega_1 - \omega_2 + \omega_3 - \omega_4) \\ 0 \end{bmatrix} \end{gathered} \tag{4.16}$$

By solving these equations in a line, it is possible to achieve the equation of motion of the quadrotor such as following:

$$\begin{gathered} \begin{bmatrix} \dot{X}^G \\ \dot{Y}^G \\ \dot{Z}^G \end{bmatrix} = \begin{bmatrix} c\psi c\theta & c\psi s\pi s\theta - c\phi s\psi & c\phi c\psi s\theta + s\phi s\psi \\ c\theta s\psi & s\psi s\phi s\theta + c\phi c\psi & s\psi c\phi s\theta - s\phi c\psi \\ -s\phi & s\phi c\theta & c\theta c\phi \end{bmatrix} \begin{bmatrix} \dot{x}^b \\ \dot{y}^b \\ \dot{z}^b \end{bmatrix} \\ \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & s\phi t\theta & c\phi t\theta \\ 0 & c\psi & s\phi \\ 0 & \frac{s\phi}{c\theta} & \frac{c\phi}{c\theta} \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \end{gathered} \tag{4.17}$$

$$\begin{bmatrix} \ddot{X}^G \\ \ddot{Y}^G \\ \ddot{Z}^G \\ \ddot{\phi} \\ \ddot{\theta} \\ \ddot{\psi} \end{bmatrix} = \begin{bmatrix} \frac{1}{m}[-(c\phi c\psi s\theta + s\phi s\psi)F_T^b - K_{dx}\dot{X}^G] \\ \frac{1}{m}[-(c\phi s\psi s\theta - s\phi c\psi)F_T^b - K_{dy}\dot{Y}^G] \\ \frac{1}{m}[-(c\phi c\theta)F_T^b - K_{dz}\dot{Z}^G] + g \\ \frac{1}{J_x}[(J_y - J_z)qr - J_r q(\omega_1 - \omega_2 + \omega_3 - \omega_4) + lK_T(\omega_4^2 - \omega_2^2)] \\ \frac{1}{J_y}[(J_x - J_z)pr - J_r p(\omega_1 - \omega_2 + \omega_3 - \omega_4) + lK_T(\omega_1^2 - \omega_3^2)] \\ \frac{1}{J_z}[(J_x - J_y)pq - K_d(\omega_1^2 - \omega_2^2 + \omega_3^2 - \omega_4^2)] \end{bmatrix} \tag{4.18}$$

From here, it is possible to write the matrix in terms of the control inputs such as below:

$$
\begin{aligned}
\ddot{x} &= \tfrac{U_1}{m}(c\phi s\theta c\psi + s\phi s\psi) \\
\ddot{y} &= \tfrac{U_1}{m}(c\phi s\theta s\psi - s\phi s\psi) \\
\ddot{z} &= \tfrac{U_1}{m}c\phi c\theta - g \\
\ddot{\phi} &= \tfrac{U_2}{J_x} + \dot{\theta}\dot{\psi}\tfrac{J_y - J_z}{J_x} - \tfrac{J_r}{J_x}\dot{\theta}(\omega_1 - \omega_2 + \omega_3 - \omega_4) \\
\ddot{\theta} &= \tfrac{U_3}{J_y} + \dot{\phi}\dot{\psi}\tfrac{J_z - J_x}{J_y} - \tfrac{J_r}{J_y}\dot{\phi}(\omega_1 - \omega_2 + \omega_3 - \omega_4) \\
\ddot{\psi} &= \tfrac{U_4}{J_z} + \dot{\phi}\dot{\theta}\tfrac{J_x - J_y}{J_z}
\end{aligned}
\tag{4.19}
$$

This formulation is used to create the system model in Simulink after implementing the DJI Tello specifications.

4.2.2 System Identification Process

System identification methods can be used in various missions such as dynamic model and parameter identification, model validation, control system design, online/real-time system identification, and fault detection. During these processes, various nonlinear and linear models can be used according to different purposes, and amount of the pre-knowledge about the system. The model selection depends on the dynamics of the system and application purposes, and the requirements may differ according to a related method.

In the experiments conducted for the thesis, it was determined whether a vision-based system consisting of a single camera and a marker could be used for quadrotor control. During operations, system identification was used to estimate the controller parameters designed for a grey-box model and to obtain a black-box model of the quadrotor using data derived from vision sensors.

In the black-box system model, the position data obtained from the vision system and the attitude data obtained from the onboard IMU sensor were used as the output. As an input, a test was taken by giving the motor speeds that can be obtained instantaneously from the experimental setup, and another identification test was conducted using the control inputs found with the motor speeds. The relationship between the control inputs and the motor speeds can be decoded as follows:

$$\begin{aligned} U_1 &= k(\omega_1^2 + \omega_2^2 + \omega_3^2 + \omega_4^2) \\ U_2 &= lk(-\omega_2^2 + \omega_4^2) \\ U_3 &= lk(-\omega_1^2 + \omega_3^2) \\ U_4 &= b(-\omega_1^2 + \omega_2^2 - \omega_3^2 + \omega_4^2) \end{aligned} \tag{4.20}$$

where U_1 is thrust force, U_2 is roll force, U_3 is pitch force and U_4 is yaw moment. In addition to the angular velocities of the motors, the length of the rotor arm (l), thrust (k) and drag (b) coefficients are also included. In the figures below, the performance of the black-box transfer function (TF) models obtained by using the motor speeds and control inputs, respectively, for position control is compared. As

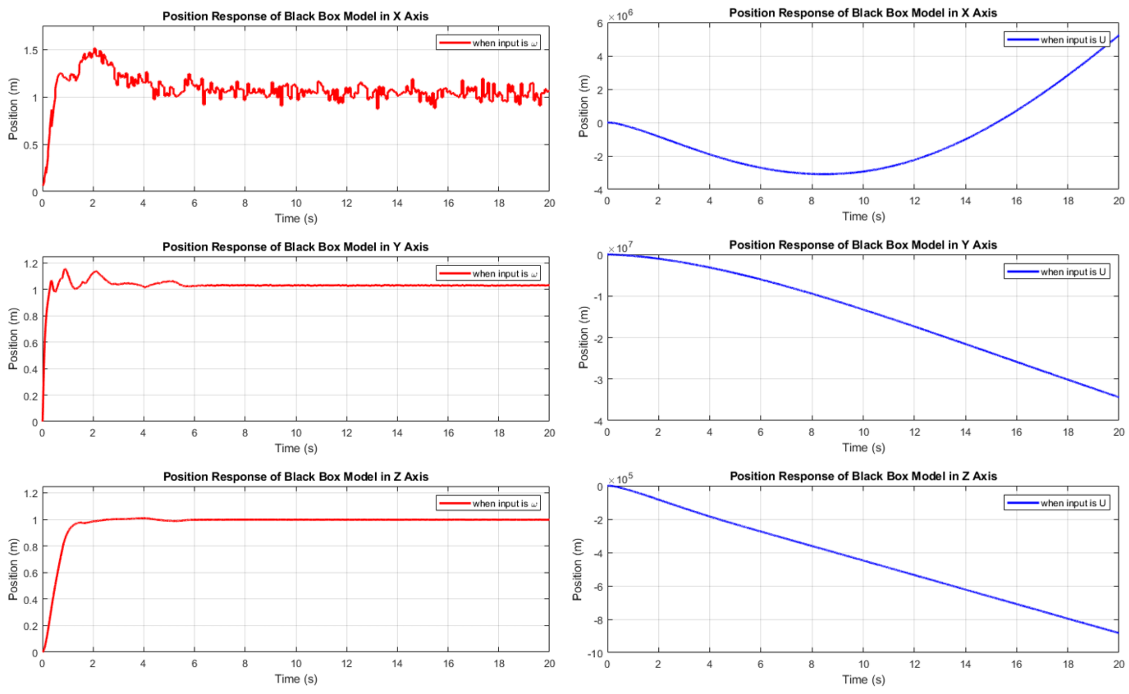


FIGURE 4.15: Effects of ω and U Usage on Estimated Black Box TF Models

can be understood from the results demonstrated in Figure 4.15, the results obtained when using motor speeds as input appear to be much more consistent than the results obtained when using control inputs (U) as input. When the linear TF model was extracted with System identification, thrust and drag coefficient effects did not appear, on the contrary, the values diverged. However, the coefficients of thrust and drag have significant effects on quadrotor control. The following section explains how these coefficients are calculated and their effects on quadrotor control.

a) Thrust Coefficient (k):

In order for the quadrotor to move, it is necessary to apply a force against gravity. This force, which is necessary for take-off, is also created by pushing the air remaining under the propellers downward. This force, which is necessary for movement, is called the thrust force.

Thrust coefficient (k) is the ratio between the square of the engine speed and the thrust force. Thrust force (T) is the force exerted by quadrotor propellers against gravity during movement. Although both the T mentioned here and the U_1 used in the control input equations represent the thrust force, the T is a constant value which calculated from system parameters and motor velocities in hover position, and U_1 is a variable which change according to motor velocities during different quadrotor motions. Since all four motors operate at the same speed and maintain stability against gravity, the hover position is chosen to determine k using the equations below.

In order to find the thrust coefficient, it is necessary to calculate the thrust Force. The thrust force can be determined using an experimental setup comprising a bending sensor, a propeller, and an autopilot. Since the equation of motion is quite clear when the quadrotor is in hover position, it is also possible to calculate the thrust coefficient mathematically. First, information regarding the quadrotor's engine speeds and propeller diameter is required when it is in the hover position. Since the system identification experiments were carried out in an indoor environment, the DJI Tello Quadrotor was used during the experiments. Under normal circumstances, the motors of the DJI Tello hover at a speed of 20,000 rpm. This information can be used to determine the angular velocity and the linear velocity v.

$$\begin{aligned} D &= 0.084m \\ \omega &= \tfrac{20000(2\pi)}{60} = 2.0944e+03rad/s \end{aligned} \tag{4.21}$$

where D is propeller diameter of DJI Tello, and ω is angular velocity of the quadrotor in hover position. From momentum theory, the thrust force T can be computed as following equation:

$$T = \tfrac{\pi}{4} D^2 \rho v \Delta v \tag{4.22}$$

where v is linear velocity of the propeller, Δv is velocity of air accelerated by propeller, and ρ is air density that is equal to $1.225kg/m^3$. From the general assumptions, the air velocity can be represented by $\Delta v = 2v$. From here, the thrust force can be found as:

$$T = \tfrac{\pi}{2} D^2 \rho v^2 = \tfrac{\pi D^4 \rho \omega^2}{8} = 105.0588N \tag{4.23}$$

The thrust coefficient can be computed by using the angular velocity and the thrust force at the hover position:

$$\begin{aligned} T &= k\omega^2 \\ k &= \tfrac{T}{\omega^2} = 2.3950e-05 \end{aligned} \tag{4.24}$$

where k is the thrust coefficient of DJI Tello quadrotor.

b) Drag Coefficient (b):

The thrust force that makes the quadrotor move is formed by pushing air particles under the propellers. The air particles present throughout the quadrotor trajectory also apply a force in the opposite direction to the movement of the UAV, depending on factors such as engine speed, propeller characteristics, air density. This force is called drag force which is also called as air resistance.

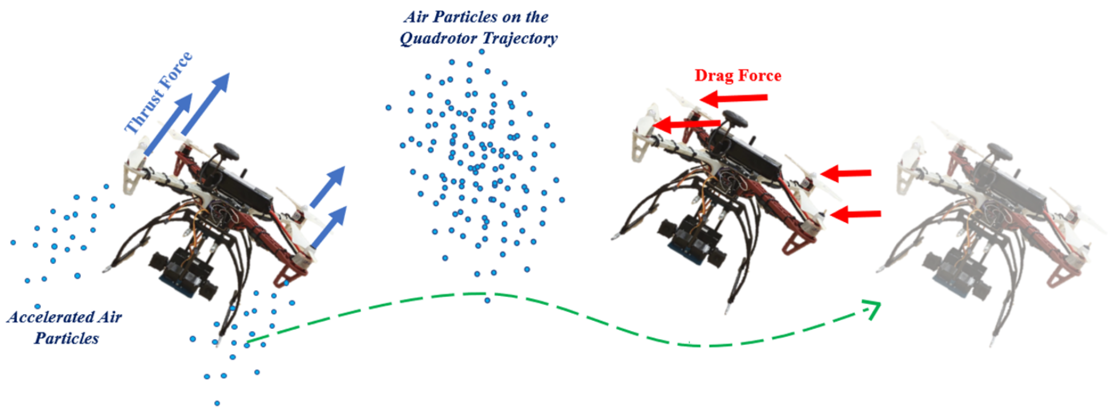

Air Particles on the Quadrotor Trajectory
Thrust Force
Drag Force
Accelerated Air Particles

FIGURE 4.16: Thrust Force and Drag Force Representations on Quadrotor Motion

Similar to thrust force, drag force is contingent upon air density, propeller characteristics, and engine rotation speed. Although there is a substantial operational distinction between thrust force and drag force, the only difference between them mathematically is that they are multiplied by different coefficients. A test bench is required to determine the drag coefficient, but this value can also be converged or estimated using analytical methods. During drag coefficient estimation, system identification methods can be used. In addition, since the parameters used in calculating thrust and drag force are common, it is also possible to establish a relationship between thrust and drag coefficients in nominal conditions. Publications comparing thrust and drag coefficients using different test benchmarks and the ratio of drag and thrust coefficients on quadrotors roughly similar to DJI Tello can be used to achieve convergence on the drag coefficient.

For a more exact result, it is necessary to perform an analytical calculation taking into account the moment of inertia of rotors and propellers with angular acceleration such as made in this source [44]. During the experiments, the mathematical operations in this source were followed and the drag coefficient was calculated as b=6.8429e-07 which is not the best but works for the assumed case.

There are various publications that say that since this coefficient is quite small, it can be neglected in the calculations. In the experiments performed, it was found that the omission of the drag coefficient did not have any negative effect on position, roll, and pitch control. However, as can be seen in U_4 equation, drag coefficient is used to calculate yaw moment.

Therefore, if this coefficient is ignored, even if the quadrotor reaches the desired position and remains in hover, the movement along the yaw axis may not stop and the quadrotor may continue to rotate around itself. The outcomes of the experiments conducted within the scope of the topic support this assertion:

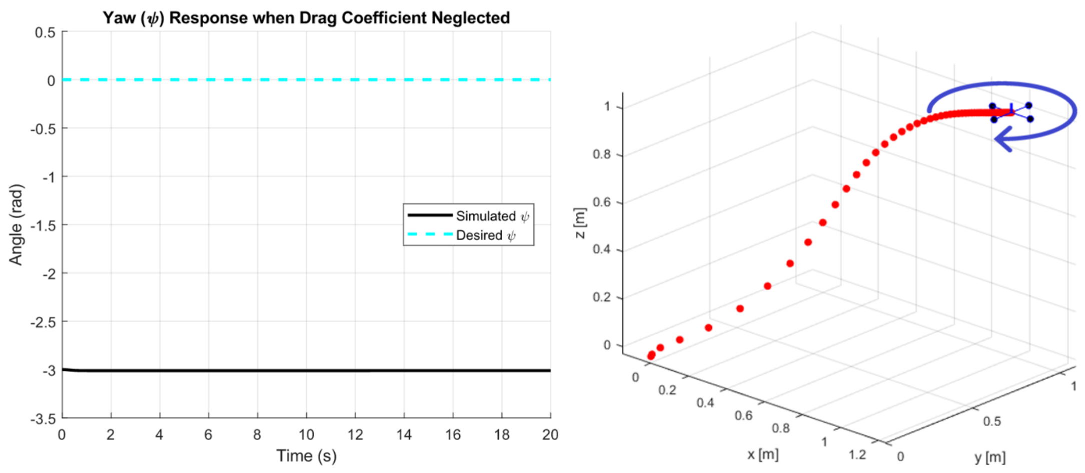


FIGURE 4.17: Quadrotor Motion when Drag Coefficient Neglected

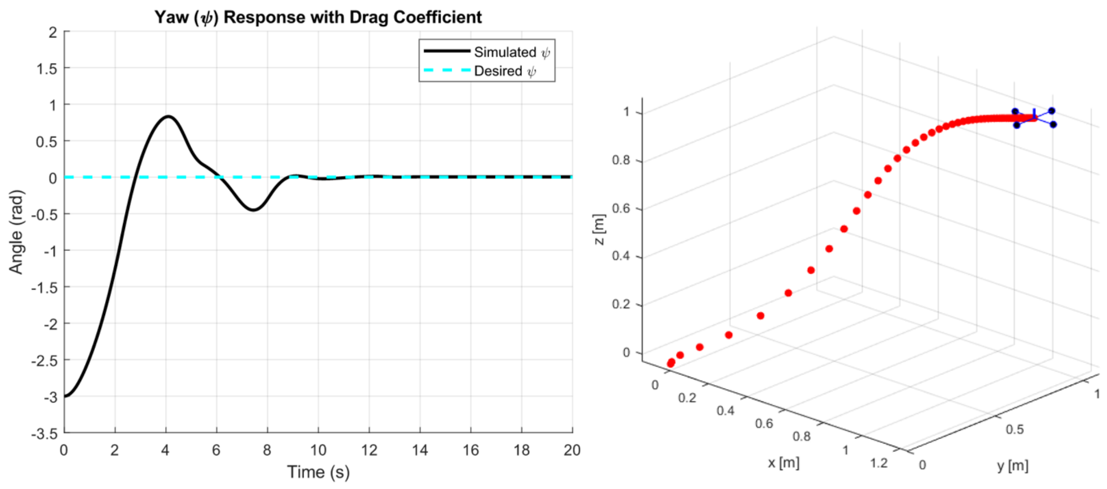


FIGURE 4.18: Quadrotor Motion with Drag Coefficient

These coefficients are multiplied by the motor speeds to find control inputs in designing of gray-box model. To compare the performance of the grey-box and black-box models, a case scenario has been developed. In this case scenario, the grey-box model is sent to the location where the quadrotor is sent in the physical setup. Consequently, LQR and PID controllers have been designed for simulated grey-box model to follow the purpose. At the end of the processing, identical desired position and attitude values were used to compare the outputs of the controller with the best performance in position control for the grey-box model, and the outputs of the identified black-box model.

4.2.3 Controller Design

Previous research has demonstrated that nonlinear control methods in quadrotor control can produce positive outcomes in more complex situations [45], [46] However, these highly nonlinear systems can be successfully controlled by linear control methods by linearizing under certain constraints.

In order to control the created grey-box system in a simulation environment, PID and LQR linear controllers have been developed. In the experiments, state space and transfer function models were created using the system id, and unknown parameters in a grey-box system were estimated. Using the state space model, the Q and R values for the LQR controller were also determined. In addition, the same system was also controlled by a multi-PID controller, and the resulting position outputs were compared.

4.2.3.1 State Space Model with LQR Controller

The system identification methods can be utilized in estimating the missing parameters of system or controller with various approaches. The techniques can be categorized based on the amount of known data. In the cases, where only a portion of data known, the online estimation techniques can be utilized. RLS (Recursive Least Squares) and Gradient Descent are suitable techniques to estimate controller parameters online. During these processes, the RLS gives better results in parameter estimation because of its forgetting factor. However, Gradient Descent performs better to decrease the error to desired level. There are also another types of models such as Model Reference Adaptive Control (MRAC), which cares more about the performance than parameter estimation during tracking or regulating operations. All of these approaches are suitable when estimating parameters online where the full dataset is not available. At this stage, state space matrices based on n4sid have been obtained in conducted experiments since all the data is known.

$$\begin{gathered}
Continuous - timeIdentifiedState - SpaceModel : \\
dx/dt = Ax(t) + Bu(t) + Ke(t) \\
y(t) = Cx(t) + Du(t) + e(t)
\end{gathered}$$

$$\begin{gathered}
A = \begin{bmatrix} -0.0027 & -0.0027 & 0.0032 & -0.0012 \\ -0.0120 & -0.0424 & 0.0044 & -0.0145 \\ -0.0395 & 0.0983 & -0.0356 & -0.0207 \\ 0.0729 & 0.0877 & 0.0012 & -0.0459 \end{bmatrix} \\
B = \begin{bmatrix} 0.0049 & 0.0136 & -0.0028 & -0.0119 \\ -0.0075 & -0.1944 & -0.0092 & 0.2046 \\ 0.0757 & 0.2486 & -0.0450 & -0.2956 \\ 0.0074 & -0.3258 & -0.0222 & 0.3404 \end{bmatrix} 1.0e-03 \\
C = \begin{bmatrix} -17.1058 & -6.7275 & 0.0356 & 1.1458 \\ -17.8818 & -9.9087 & 0.0276 & 1.8673 \\ -15.5999 & -2.2727 & -1.3994 & 3.7613 \\ -2.0327 & -7.3782 & 1.5360 & -2.1738 \\ -5.1228 & 1.4499 & 7.2792 & 6.8291 \\ 29.5386 & -24.5856 & -0.1792 & 8.3781 \end{bmatrix} D = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}
\end{gathered} \tag{4.25}$$

$$K = \begin{bmatrix} -0.8215 & -0.5575 & -0.4535 & -0.1623 & -0.1158 & 0.5741 \\ 0.5335 & 0.3024 & 0.4688 & -0.1113 & 0.1689 & -0.6258 \\ 0.8459 & 0.5837 & -0.0456 & 0.6862 & 0.8491 & -0.6359 \\ -1.6117 & -1.1064 & -0.3796 & -0.7931 & 0.1458 & 1.5603 \end{bmatrix} \tag{4.26}$$

The Q and R parameters are required for the LQR controller in addition to the states. The Q and R matrices used in these calculations are known as performance matrices, and their ratio provides a balance between the controller's energy consumption and the system's performance. The matrix Q is referred to as the performance effort coefficient, and it is set to a high value when the performance of the system is crucial and there is no concern regarding the controller's energy consumption. However, in other situations where energy consumption is at least as important as system performance, the coefficients R and Q should be chosen in close proximity to one another, somewhere between 0 and 1.

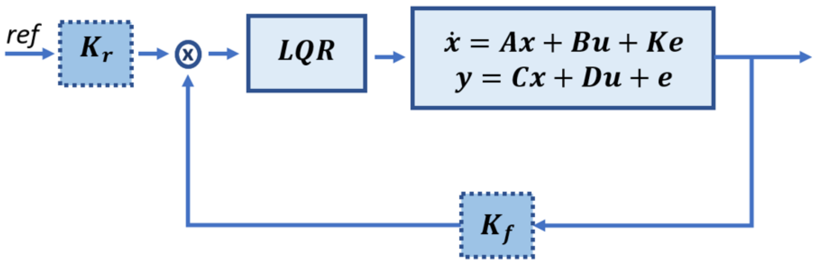


FIGURE 4.19: LQR Control

At this point, it is possible to summarize the effects of the Q and R coefficients on state variables as follows:

• If larger R values are chosen, K (the feedback gain found during LQR formulation) becomes smaller and the state variables approach zero more slowly.

• If lower R values are chosen, K increases and the state variables approach zero more quickly.

• If larger Q values are chosen, K increases and the state variables approach zero more quickly.

• If lower Q values are chosen, K decreases and the state variables converge on zero more slowly. For these purposes, the Q and R parameters were altered multiple times until Q=1 and R=0.001 were ascertained and the desired outcome was attained. Using the States and Q and R parameters, the feedback gain can be calculated using the following Matlab command:

$$[K_f SE] = dlqry(A, B, C, D, Q, R) \tag{4.27}$$

where K_f is feedback gain which is multiplying with states. It is also possible to update the reference gain with the expression below:

$$K_r = (B^T S B + R)^{-1} B^T [I - (A - BK_f)^T]^{-1} C^T Q \tag{4.28}$$

From these equations, the feedback and reference gains are obtained as:

$$K_f = \begin{bmatrix} 0.1253 & 0.8652 & -0.1758 & -0.2332 \\ 1.0345 & 3.1941 & -0.4993 & 1.0630 \\ 0.0866 & -0.2424 & 0.0787 & 0.2472 \\ -1.2509 & -3.9412 & 0.6282 & -1.0028 \end{bmatrix}$$
$$K_r = \begin{bmatrix} -4.1648 & -3.2998 & -0.3656 & -3.2558 & -4.0667 & 0.8079 \\ 4.4267 & 3.3237 & -1.9621 & 5.8107 & 5.0416 & -3.4641 \\ -3.8868 & -2.7508 & 3.7504 & -7.1213 & -5.0524 & 5.3398 \\ 3.6251 & 2.7271 & -1.4225 & 4.5664 & 4.0777 & -2.6834 \end{bmatrix} 1.0e+04 \tag{4.29}$$

The position response of the quadrotor with these gains becomes as in the figures below:

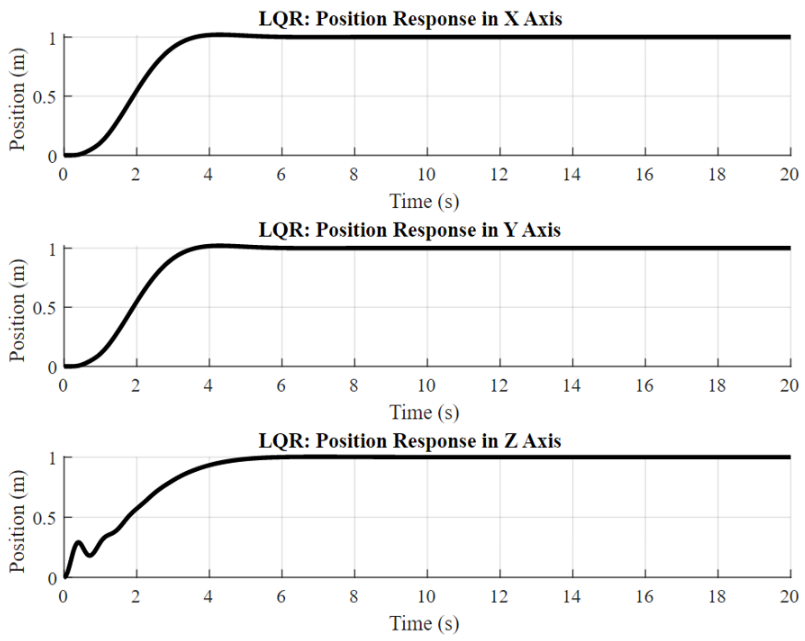


FIGURE 4.20: LQR Control Position Response

4.2.3.2 Transfer Function Model with Multi-PID Controller

PID control is a control method that attempts to reset the difference between the plant's reference values and actual values. In order to control the highly nonlinear quadrotor system, it is necessary to design separate controllers for position and attitude control. Despite the fact that separate controllers are designed for position and attitude control, these parameters are interdependent. To change the position of the quadrotor, different torques must be applied to the motors. This condition causes the quadrotor to remain trained on the pitch and roll axes for a period of time, allowing it to move forward-backwards or right-left. According to this fundamental motion relationship, the attitude control parameters that are attempted to be stabilized affect the position of the quadrotor. For this reason, it is also crucial for the position control of the quadrotor that these parameters reach the reference level much more quickly. To create a healthier control diagram, the block that controls position may be placed in the outer loop, while the block that controls attitude and altitude may be placed in the inner loop.

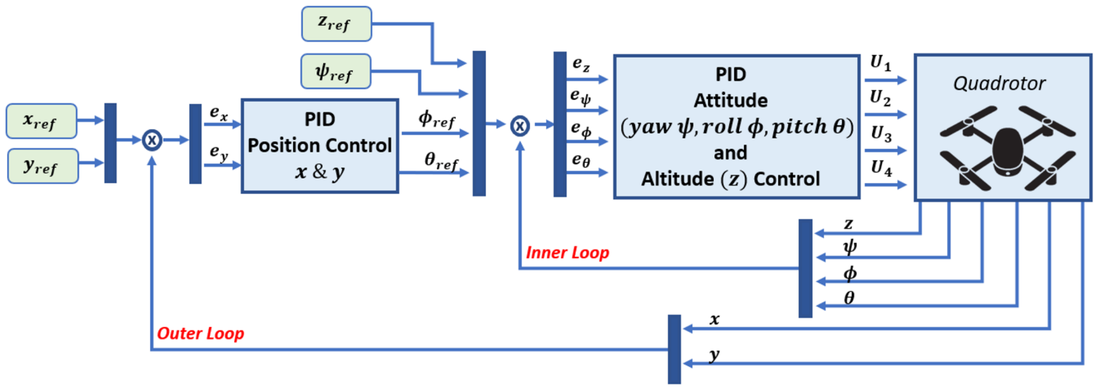


FIGURE 4.21: PID Control Block Diagram

The relationship between control inputs and rotor speeds was deciphered under the heading "System identification Process." This formulation permits the determination of control inputs only when the instantaneous engine speeds are known. However, when the motor speeds are unknown, the control inputs must be calculated using error dynamics and system parameters.

PID controllers are the most prevalent linear controllers where error dynamics and control issues intersect. In the experiments, 6 different PID controllers were used, to control each position and attitude error. The equations formed in this context are as follows. U_1, which represents thrust force, is found with x, y and z position errors, while attitude parameter errors are used to calculate control inputs U_2, U_3 and U_4:

$$\begin{aligned} U_1 &= (g + K_{z,D}(z_d - z) + K_{z,P}(z_d - z))m/(C_\phi C_\theta) \\ U_2 &= (K_{\phi,D}(\phi_d - \phi) + K_{\phi,P}(\phi_d - \phi))I_{xx} \\ U_3 &= (K_{\theta,D}(\theta_d - \theta) + K_{\theta,P}(\theta_d - \theta))I_{yy} \\ U_4 &= (K_{\psi,D}(\psi_d - \psi) + K_{\psi,P}(\psi_d - \psi))I_{zz} \end{aligned} \tag{4.30}$$

Moreover, the position parameters x and y are controlled by second PID block that connects outer loop. The mathematical expression can be represented as follows:

$$\begin{aligned} \ddot{X} &= (K_{x,P} + K_{x,I}/s)(x_d - x) \\ \ddot{Y} &= (K_{y,P} + K_{y,I}/s)(y_d - y) \end{aligned} \tag{4.31}$$

After determining the control inputs, the engine speeds can be calculated using the following equations:

$$\begin{aligned} \omega_1 &= \sqrt{\tfrac{U_1}{4k} + \tfrac{U_3}{2kl} + \tfrac{U_4}{4b}} \\ \omega_2 &= \sqrt{\tfrac{U_1}{4k} - \tfrac{U_2}{2kl} - \tfrac{U_4}{4b}} \\ \omega_3 &= \sqrt{\tfrac{U_1}{4k} - \tfrac{U_3}{2kl} + \tfrac{U_4}{4b}} \\ \omega_4 &= \sqrt{\tfrac{U_1}{4k} + \tfrac{U_2}{2kl} - \tfrac{U_4}{4b}} \end{aligned} \tag{4.32}$$

The PID parameters used during the experiments were tuned with the Simulink auto Tuner. The position outputs obtained with the PID and the LQR controller mentioned in the previous section are shared below:

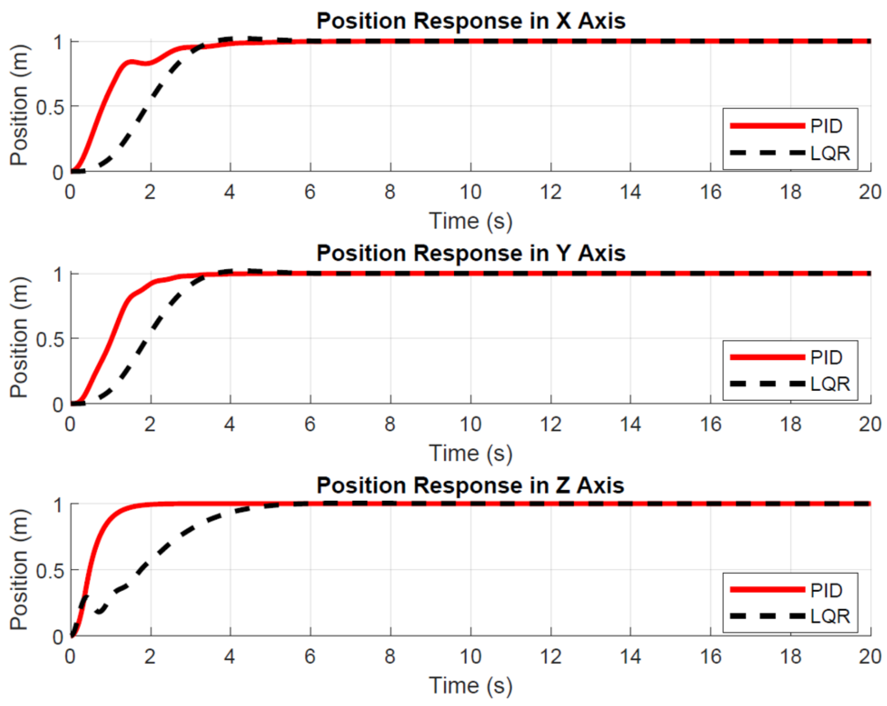


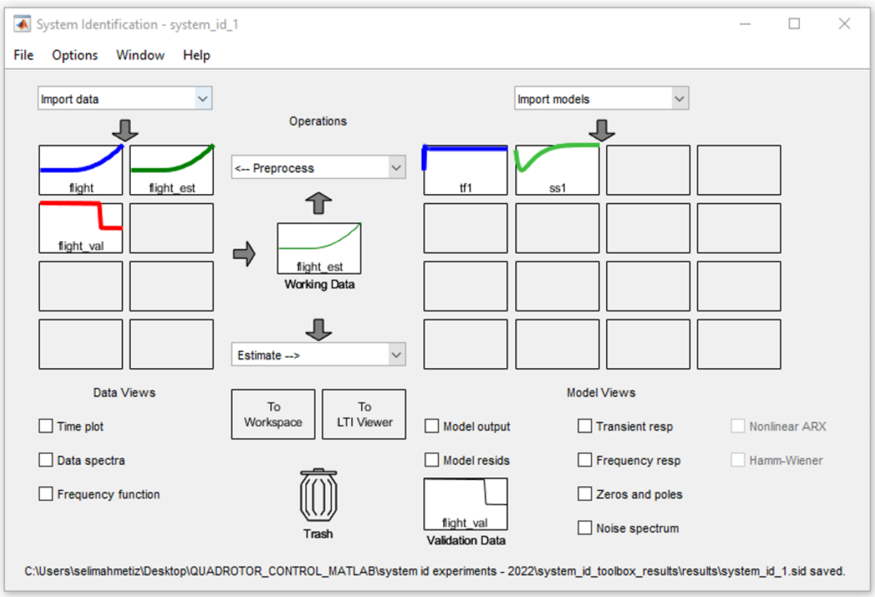
FIGURE 4.22: Comparison of LQR and PID Controller Performances on Position Control

When comparing the position response performance of LQR and PID controllers, it has been observed that the designed PID controller provides faster and more robust results. Therefore, in the subsequent section, the outputs of the simulation system, which is controlled by the PID controller, are used to compare the system's performance with that obtained by the black box system identification technique.

4.2.4 Onboard Vision System Performance on Position and Attitude Control of DJI Tello Quadrotor

In the experiments conducted in scope of the thesis, the attitude data is taken from IMU sensor, the position data is gathered from the onboard vision system and a marker on ground, as well as the instant motor velocities are stored thanks to SDK of utilized quadrotor. During these operations, a total of 49939 lines of input-output data were collected with a sample time of 0.001 over a duration of 50 seconds. The collected data were divided into 80% estimation and 20% validation data for use during the system identification process.

Then using these data, linearized models of the highly nonlinear quadrotor system were sought. In order to implement the PID controller, that uses the error dynamics of the system, in the system identification toolbox, the transfer function model has been developed, while the state space model has been developed for the LQR controller, which uses the state space matrices to control a system.



C:\Users\selimahmetiz\Desktop\QUADROTOR_CONTROL_MATLAB\system id experiments - 2022\system_id_toolbox_results\results\system_id_1.sid saved.

FIGURE 4.23: Toolbox Data Separation and Transfer Functions and State Space Models

In light of the decision made in the preceding section to compare simulation performance under the control of the PID controller, the black-box system identification transfer function model is chosen for use in the Simulink diagram.

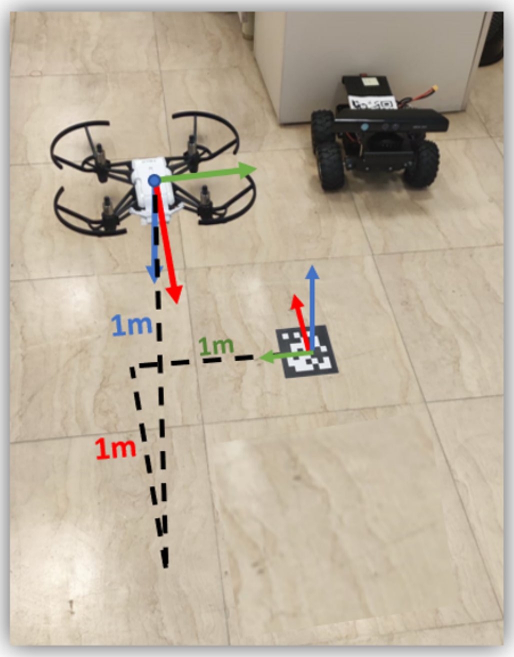


FIGURE 4.24: During Flight Data Gathering

In the case scenario, it is assumed that the quadrotor intends to hover at the desired location, which remains 1 meter away in the x, y, and z directions. Using a 3D-printed component and a mirror for the quadrotor's onboard camera's forward-facing position, data is collected for surveillance of the marker on the ground. The attitude data measured by the quadrotor's onboard IMU sensor. During the motion, the instantaneous motor speeds were measured as the quadrotor moved towards the desired position, and position and orientation data were obtained using the Aruco marker method and the onboard IMU sensor. As an output of the collected data, the position and attitude data were entered into the system identification toolbox. In the preceding section, it was demonstrated that the black-box system identification method produced divergent position responses when attempting to decode a linear model between control input and output. That's why, in order to obtain the TF model, direct motor speeds were used as input during the black-box system identification procedure.

There are 24 transfer functions installed between 4 inputs and 6 outputs in the resulting model. To compare the performance of the black-box system estimated by system identification to that of the grey-box system designed in a simulation environment, identical inputs were given to both models.

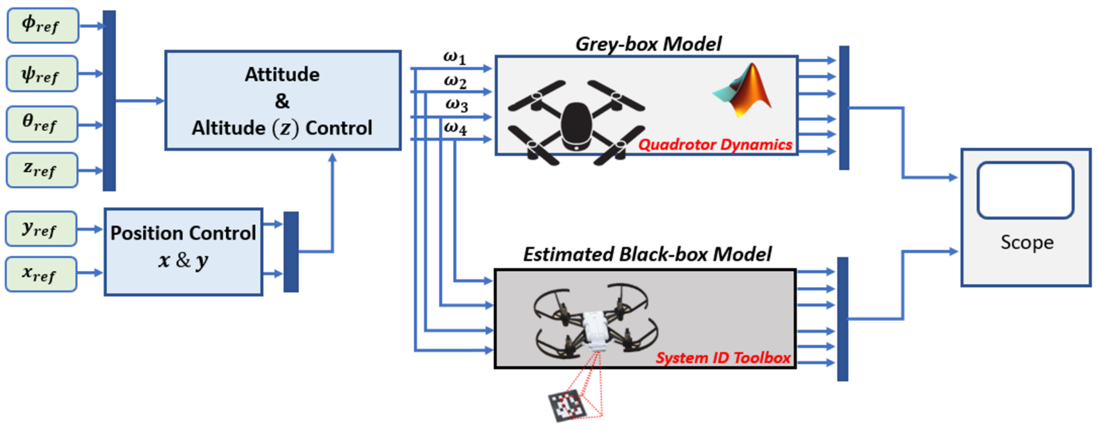
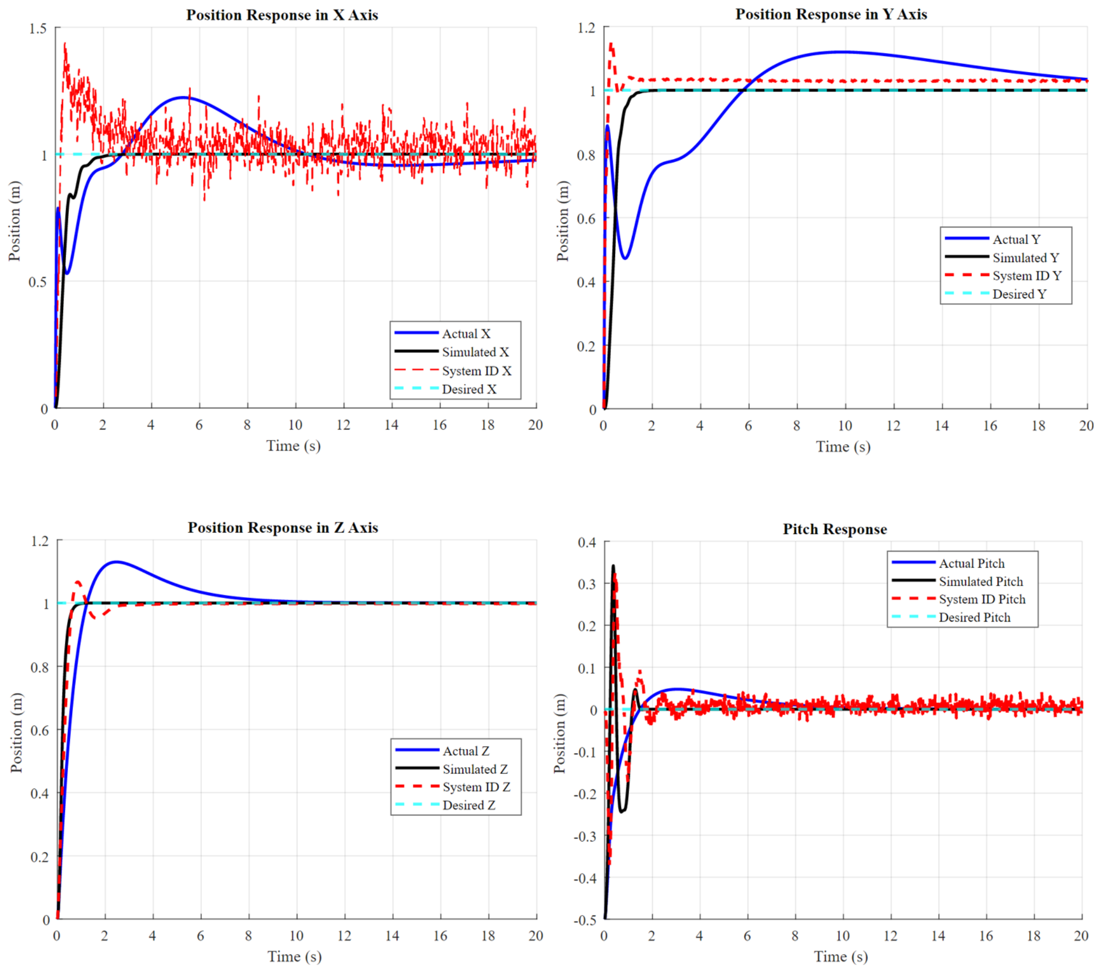


FIGURE 4.25: Grey-box and Black-box Model Comparison with Same Input Signals

The position and attitude responses of these two models, as well as the actual setup outputs, have been added to the graphs. The obtained results are as follows:



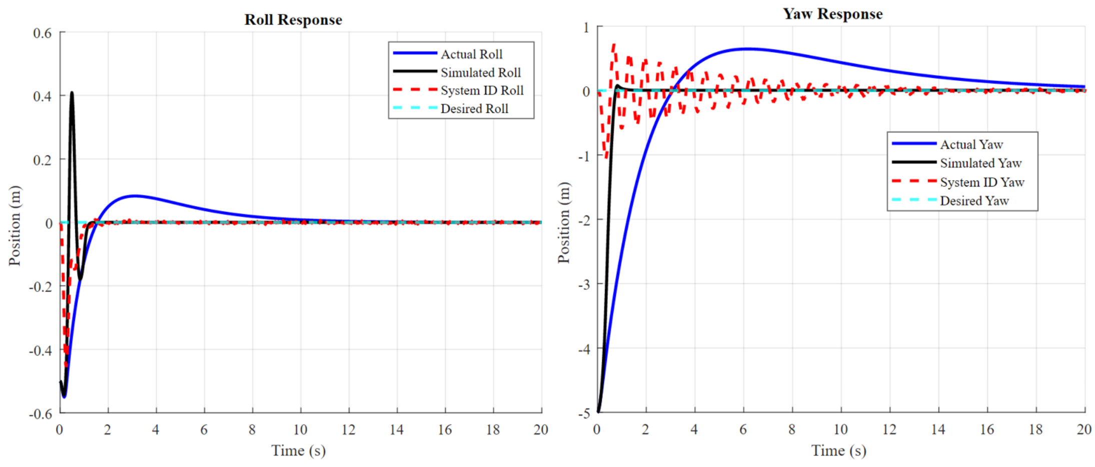


FIGURE 4.26: Comparison of Position and Attitude Responses

The obtained outputs demonstrated that the vision system, which consisted of a single camera and a marker, produced comparable position control results to the grey-box system. Similarly, the estimated attitude responses derived from IMU data became comparable to the simulated attitude responses. Thus, it has been demonstrated that the installed vision-based control system produces consistent quadrotor motion control results.

4.3 Vision-based Autonomous Landing of a UAV on a UGV in GNSS-denied Cases

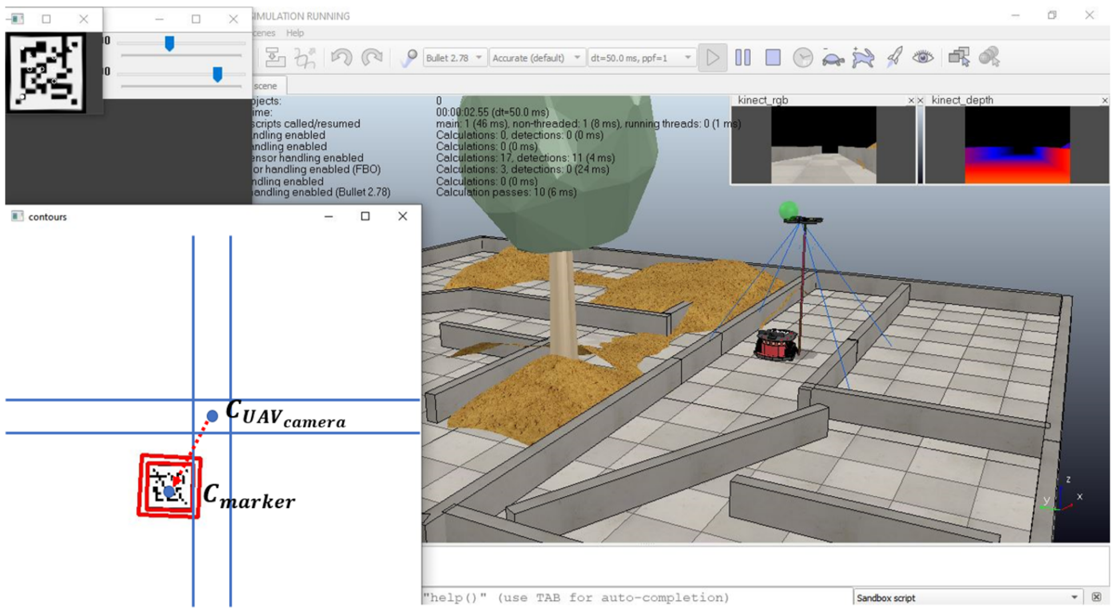
UAV landing on a desired platform is an enticing research field, particularly in the last few decades. Numerous studies have been conducted on landing on static and dynamic platforms. Within the scope of this thesis, a quadrotor system has been developed that autonomously takes off from a ground vehicle, performs the mapping process, and then returns to the ground vehicle using only onboard sensing and computing operations. No prior knowledge of the location or speed of the target platform on the ground vehicle is required for this operation. A motion tracking system that observes the movements of UAVs and UGVs from the outside is also unnecessary for the task at hand. Importantly, the UGV's marker must be detected in order to initiate the landing procedure.

After detecting the marker on the UAV ground vehicle, the system begins to wait for the optimal landing time. The UAV follows the UGV's movement based on a cost function created between energy consumption and performance, and lands vertically on the UGV when it reaches the appropriate level of stabilization. If the battery level falls below a certain threshold, the UGV attempts to land on it even if it is in motion. During this procedure, UAV motion control algorithms and estimated UGV states were utilized, as well as quadrotor state estimation, landing platform detection and landing platform state estimation, tragic planning, and UAV control steps.

The states of UAVs are estimated using the onboard IMU and onboard vision system. This procedure determined the orientation of the UAV based on its position, velocity, and UGV.

After completing the mapping process, the UAV returns to the area from which it took off and searches for the marker located on the hovering UGV. This marker was previously introduced to the UAV during system identification procedures, and the Aruco Marker method was utilized for pose estimation. For this reason, after detecting the UAV marker, it tries to capture stability, which is the appropriate moment for landing.

If the UGV is in motion, the quadrotor follows it from a predetermined height. The tracking process is determined by the distance between the detected marker's center and the center of the UAV camera's field of view. The process of tracking is determined by the distance between the center of the detected marker and the center of the UAV camera's field of view. This difference represents the position error, which is corrected by the UAV. Thus, he will pursue the UGV.



```
jects:                          0
ime:                            00:00:02.55 (dt=50.0 ms)
scripts called/resumed          main: 1 (46 ms), non-threaded: 1 (8 ms), running threads: 0 (1 ms)
ndling enabled                  Calculations: 0, detections: 0 (0 ms)
ndling enabled                  Calculations: 0 (0 ms)
ensor handling enabled          Calculations: 17, detections: 11 (4 ms)
or handling enabled (FBO)       Calculations: 3, detections: 0 (24 ms)
ndling enabled                  Calculations: 0 (0 ms)
andling enabled (Bullet 2.78)   Calculation passes: 10 (6 ms)
```

FIGURE 4.27: Vision-based Motion Control of the UAV According to Moving UGV

$$e = \begin{bmatrix} x_{c_{marker}} \\ y_{c_{marker}} \end{bmatrix} - \begin{bmatrix} x_{c_{UAV_{camera}}} \\ y_{c_{UAV_{camera}}} \end{bmatrix} \tag{4.33}$$

Since landing on a moving platform is a more hazardous action, stabilization for landing is prioritized in the UGV's cost function. This procedure is repeated until the UGV is stable or the UAV's battery level falls below a predetermined threshold. When the UAV's battery level falls below the predetermined threshold, landing operations on the moving platform commence. During this procedure, state estimation is performed using the non-holonomic Kalman Filter for UGV.

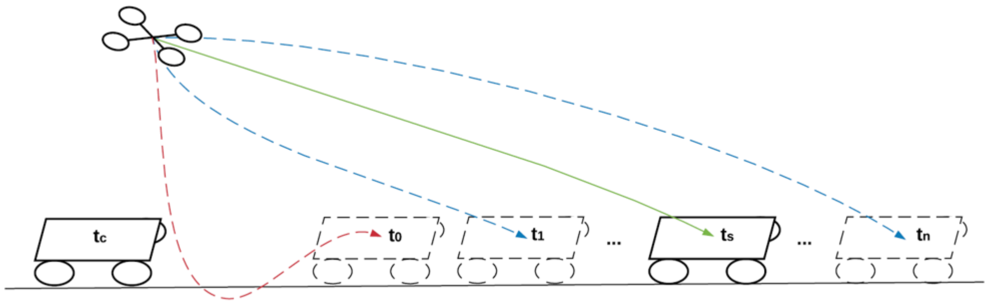


FIGURE 4.28: Vision-based Landing on Moving UGV

During the prediction, the Kalman filter predicts the UGV states according to following equation:

$$\dot{x}(t) = f(x(t), u(t)) + \omega(t) \tag{4.34}$$

where $x(t)$ and $u(t)$ are the states and input of the system where the $\omega(t)$ is process noise. The function $f(x, u)$ is the dynamical model of the moving platform. From the equation,the relation between the position coordinates and the inputs can be expressed as follows:

$$\begin{gathered}
\dot{p}_x = \dot{x}(t) = v(t)cos\theta(t) \\
\dot{p}_y = \dot{y}(t) = v(t)sin\theta(t) \\
\dot{p}_z = \dot{z}(t) = 0 \\
\dot{\theta}(t) = \omega(t) = u_1 \\
\dot{v}(t) = u_2
\end{gathered} \tag{4.35}$$

where p_x, p_y, p_z are the position coordinates of the UAV in world frame, θ is heading angle, and $v(t)$ is tangential velocity of the UAV where u_1 and u_2 represent the control inputs.

After state prediction, the phase correction is another important step of Kalman Filter which can be represented by the following equation:

$$\dot{\hat{x}}(t) = f(\hat{x}(t), u(t)) + K(t)(z(t) - h(\hat{x}(t)) \tag{4.36}$$

where the matrix K(t) is the Kalman gain, $f()$ is a kinematic constant velocity model, $h()$ is observation modeal and $z()$ is the measurements vector. In brief, the Kalman filter estimated the UGV's pose. In cases where the UAV misses the view of the landing marker, the UAV continues on its trajectory based on its last position and observation records from the Kalman Filter until it locates the marker. When the UAV detects the marker once more, the landing procedure begins.

The figures below show the autonomous landing process builds into V-rep simulation program. It is possible to see the position and velocity changes of the UAV in x, y, and z axes.

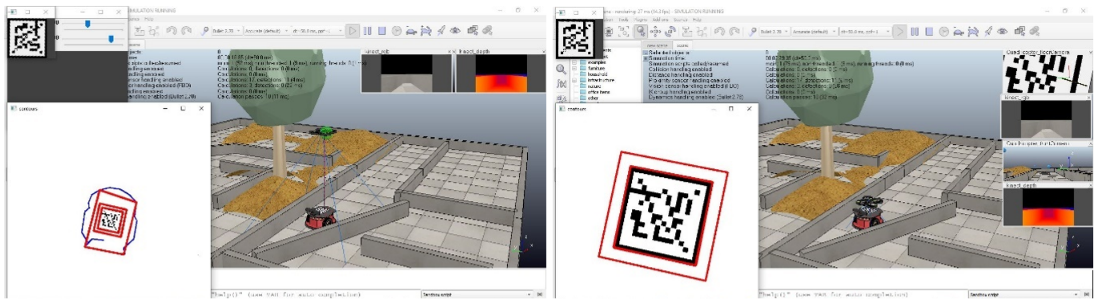


FIGURE 4.29: Vision-based Tracking and Landing Processes on V-rep Simulation Scene

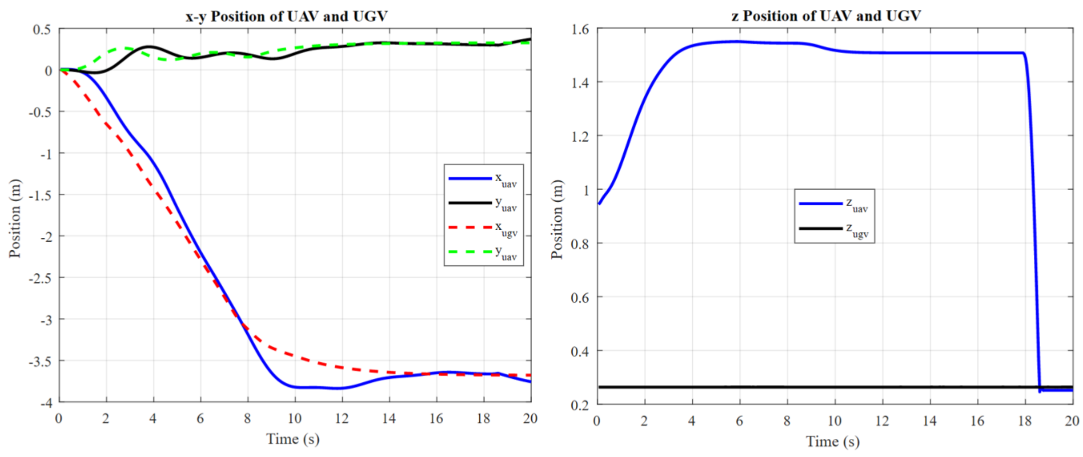


FIGURE 4.30: x-y-z Position of UAV and UGV

Chapter 5

Experimental Results

5.1 Experimental Setup

During the indoor and outdoor experiments conducted as part of the thesis, one UGV and two distinct quadrotors were utilized. These configurations are primarily comprised of controllers, autopilots, onboard surveillance systems, and communication modules.

5.1.1 Unmanned Aerial Vehicle Layout

UAV tests were conducted both indoors and outdoors as part of the scope of the thesis. For outdoor test environments, an autonomous quadrotor has been developed that can withstand environmental conditions, observe the terrain with an onboard stereo camera, and store images in its own memory, which can be transmitted to the base station in real-time if necessary.

Since this quadrotor was designed to operate in an indoor environment, the DJI Tello Quadrotor model, which was purchased in the laboratory for experimentation and has a monocular camera and control system access, was utilized. The common purpose of these two quadrotors is to perform terrain analysis in order to find the safest path to the ground vehicle, detect the marker on the ground vehicle, and provide motion control in accordance with the detected marker.

a) Built Quadrotor

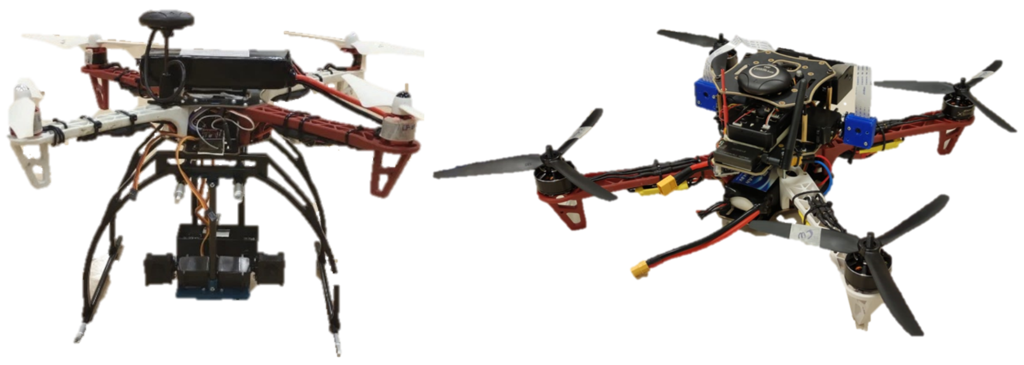


FIGURE 5.1: Built Quadrotor Setups-number1 on the left and number2 on the right

Two distinct quadrotors were constructed for use in the experiments, and then tests were conducted with the quadrotor numbered one in the figure because it flew more steadily. The built-in Quadrotors are primarily composed of an autopilot, controller, communication module, and stereo camera module, in addition to the essential flight equipment such as an ESC and engine. In addition to the design difference between these two models, the engines and propellers also vary significantly. The number one quadrotor uses the 10000kw-30A brushless motor, while the number two quadrotor prefers the 480kw-28A motor. Consequently, when the same lipo battery is utilized, quadrotor number two flies for a longer duration. In addition, the number one quadrotor had two blades-propellers, while the number two quadrotor had three blades-propellers. This distinction allowed the number two quadrotor to fly more quietly and steadily, while the number one quadrotor was capable of more aggressive maneuvering.

Autopilots play a crucial role in providing low-level control of robots in which they are implemented. Autopilots are distinguished from flight controllers by the fact that they have an interface for assigning waypoints and the ability to autonomously follow those waypoints. While selecting an autopilot for the experiments, various autopilots including Pixhawk, APM, DJI Naza, and Navio 2 were tested. These autopilots, excluding the DJI Naza, share the capacity to collaborate with an external controller, such as the Raspberry pi. DJI Naza blackbox is a controller, so it does not provide access to the control algorithm and is used in conjunction with the included GPS and IMU sensors. Due to its simple operation, this autopilot was utilized during stabilization tests after the quadrotor chassis was constructed. The only difference between Pixhawk and APM is the price and a few pins. Dec. Thanks to different frameworks, these autopilots can be used to control a variety of flying robots or ground vehicles. They function as a low-level controller of the robot and are equipped with an internal IMU and external sensors. The GPS they carry allows them to autonomously follow the assigned waypoints. For more complex vision-based autonomous missions, however, they require an external controller, such as a Raspberry Pi. These controllers, unlike the DJI Naza, are compatible with the Raspberry pi. The Navio 2 is one of the most convenient autopilots compatible with the Raspberry pi. Navio can be mounted directly on the Raspberry Pi card using its pins, and Pixhawk is capable of performing all autopilot functions for quadrotor, fixed-wing, and rover-style robots. Moreover, Autopilots cannot independently perform complex autonomous operations. Therefore, the systems must be equipped with three distinct onboard computers. In the experiments conducted for the thesis, a Raspberry Pi was used as an onboard computer for unmanned aerial vehicles.

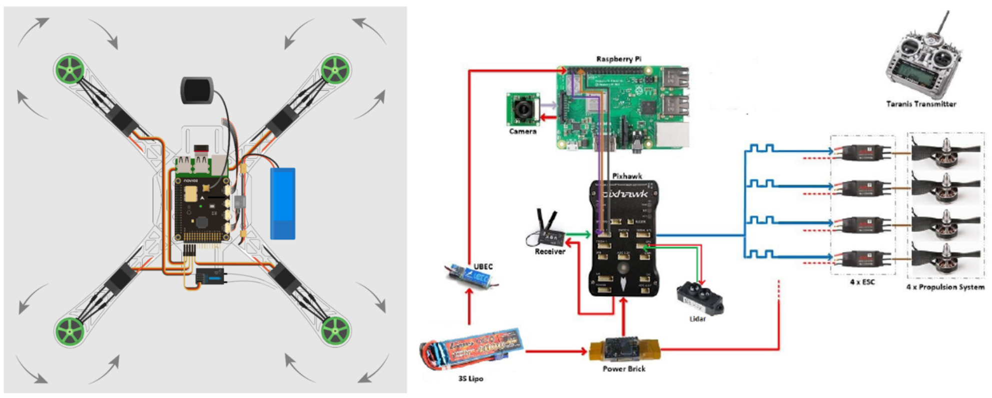


FIGURE 5.2: Setups of Navio 2 and Pixhawk Autopilots

A motion tracking system that follows the UAV-UGV collaboration process from the outside was not utilized in the experiments. A motion tracking system is a system that performs relative pose detection with high precision in indoor environments, but is unsuitable for outdoor environments. Therefore, the process of estimating the relative positions of the UAV and UGV was conducted using onboard cameras and the Aruco marker method. The UAV is equipped with a stereo camera system consisting of two 5-megapixel, 160-degree fish-eye raspberry cameras connected to the stereo pi development card. Using these cameras to capture 1080p images, terrain depth reconstruction was performed. Concurrently, the marker on the UGV was identified using one of the captured images, and the autonomous landing and tracking processes were carried out accordingly.

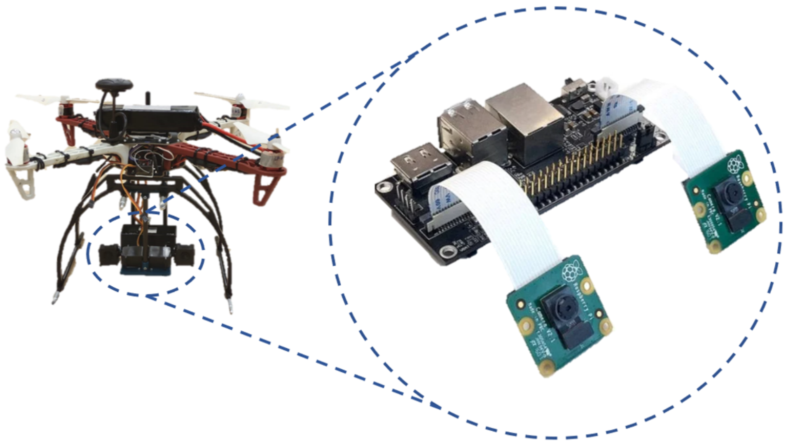


FIGURE 5.3: Stereo Onboard Camera

When performing depth analysis with a stereo camera, the relationship between the baseline distance and the flight altitude imposes certain restrictions. Due to this, depth analysis cannot be conducted safely with a stereo camera above a certain height. It has also been used with a 4k monocular camera to capture images from greater heights and to calculate depth using the structure from motion method.

The built-in quadrotor can transmit images in real-time up to 400 meters to a Jeson nano card or computer serving as a ground station. In the interim, real-time wifi image transfer over the network in an indoor environment made for Thu

24-30 frames per second (fbs) or the resolution of an image with the transfer speed may be decreased based on the distance. In such instances, the quadrotor stores the images in its internal memory, and jetson nano processes the images after landing.

b) Indoor Quadrotor: DJI Tello

DJI Tello is a commercial drone designed for indoor use with a high level of stabilization. Its SDK (software development kit) makes it possible for users to design a new controller and adapt it to Matlab and Python. However, Tello does not permit access to and manipulation of its primary flight controller, which is responsible for low-level control. It connects to the computer through Wi-Fi and has three ports with the same IP address. The first port provides communication between the drone and the user, allowing the user to send commands to the UAV. The second port shares sensor information and the current state of the UAV, including barometer values, ToF (Time of Flight) sensor values, battery percentage, and Wi-Fi signal strength. During computer vision-based controller processes, all of them function simultaneously.

The Tello, on the other hand, is an easily controlled quadrotor. It has a 100-meter range and a flight time of only 13 minutes. The maximum permitted height is 30 meters. Due to the ToF camera and integrated IMU, it is therefore better suited for use in indoor environments, but can struggle to operate in outdoor environments with minimal disturbances. The ToF camera, also known as the flash lidar sensor, is one of the quadrotor's most sophisticated components. Object scanning, measuring distance, indoor navigation, obstacle avoidance, gesture recognition, tracking objects, measuring volumes, reactive altimeters, 3D photography, and augmented reality are a few of the possible applications. DJI Tello uses it primarily for stabilization, but it can also be used for other missions, such as autonomous landing and object tracking.

The front-facing onboard camera of the DJI Tello is 5 megapixels and has a 720p resolution. In order to conduct indoor tests involving autonomous tracking, landing, and pose estimation with a marker and onboard camera, a 3D component was designed and attached to the quadrotor in order to reflect the camera's view onto the ground using a mirror.

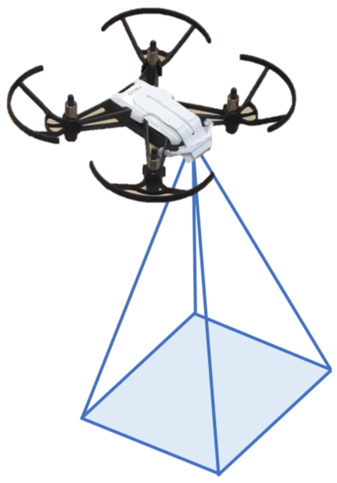


FIGURE 5.4: DJI Tello Quadrotor with the 3D Printed Mirror Accessory

5.1.2 Unmanned Ground Vehicle Layout

A rover with a gear ratio of 75:1 between each skid steering 4 wheels and the engine shaft, whose control algorithms are very similar to differential drive, was favored as a land vehicle in the experiments. Jetson Nano development card was utilized to capture and process UGV images captured by UAV. Jetson nano is one of the most powerful onboard development cards available, with 2GB and 4GB capacities and a 5V input pin designed to run AI algorithms for image processing operations such as image segmentation and classification, and object detection. The Raspberry pi UAV on the UAV transmits captured images to the Jetson Nano on the UGV via the wifi-network to which it is connected. Using the images it receives, Jetson Nano calculates the UGV's global path and transfers this information to the raspberry card that controls the engines.

Jetson nano was assisted by a Raspberry Pi for UGV motion control. This component is connected to the raspberry pi and controls the Xbox 360 Kinect stereo camera motion used on the UGV. For depth detection, the Kinect camera and UGV view are used for dynamic obstacle avoidance. For this procedure, a simpler controller and stereo cameras may also be preferred.

The UGV was equipped with 7.2V brushed motors. When the wheels are in contact with a flat surface, these motors require 6.6A of current. This current requirement can increase to 14A when there is a load on them or when moving in a rough terrain. RoboClaw motor driver is used to drive such high-current motors for this reason. RoboClaw motor driver is a motor driver that can drive two 15A DC motors simultaneously and has receiver connection pins. This driver is also compatible with onboard controllers including Raspberry Pi, Arduino, and Jetson.

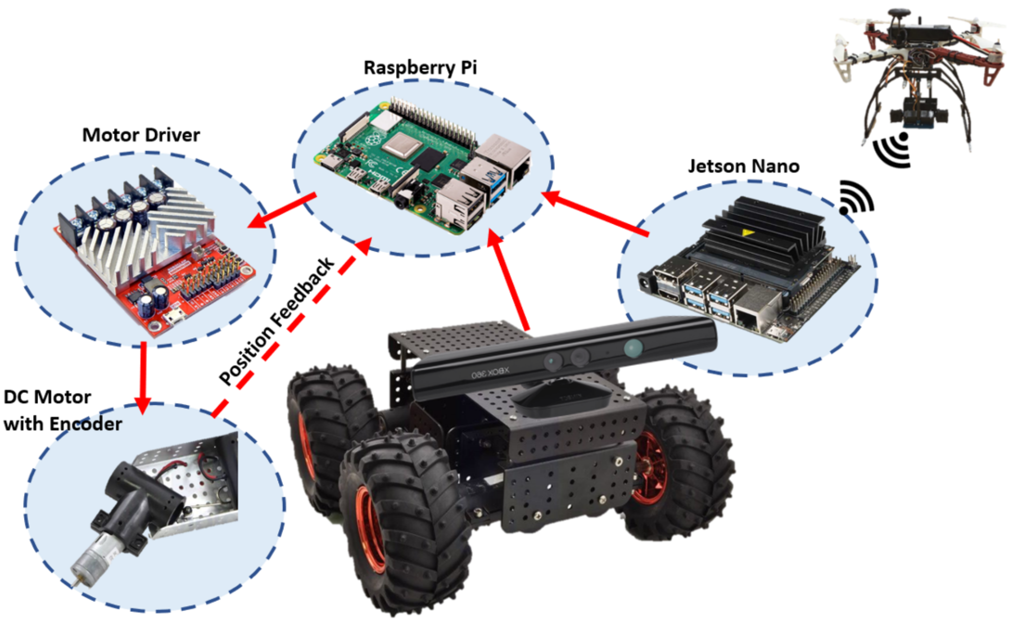


Figure 5.5: UGV Layout

5.1.3 Test Environments

Throughout the experiments, tests were conducted in both simulated and real environments. Both matlab and vrep simulation programs were utilized in simulation-based tests. When designing the Quadrotor's low-level controller, MATLAB was used.

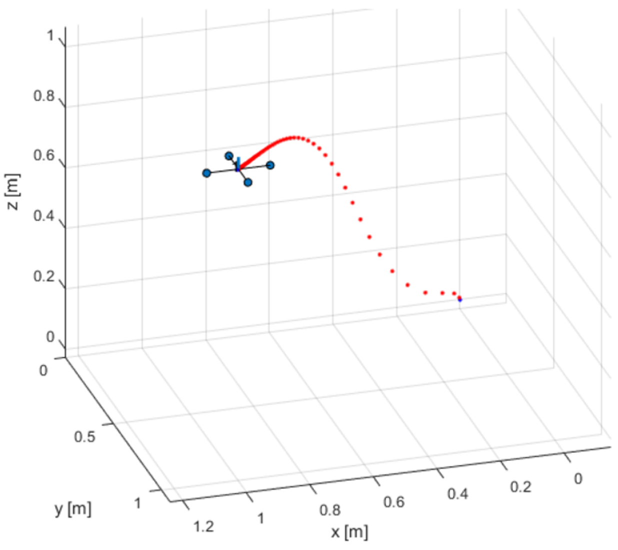


FIGURE 5.6: Matlab Capture during a Test

Virtual Robot Experimentation Platform, also known as V-Rep, is a three-dimensional simulation program that includes a variety of mobile and stationary robot platforms, in addition to numerous assistant objects that can be used to construct test scenes.This program is one of the most popular simulations due to its compatibility with Python, Matlab, and C++, which are frequently used for robot programming. On the other hand, the fact that 3D solid models designed in programs such as Solidworks Catia can be implemented in simulation with high resolution is one of the reasons why this simulation program is preferred.

The 3D model of the ground vehicle and the sensors used in the actual setup was applied to the simulation in the v-rep experiments conducted for the thesis. After a quadrotor scanned the unknown environment, the ground vehicle was allowed to travel from the starting point to the destination. The simulation shows the quadrotor taking pictures of the surface, scanning the test area, and then autonomously landing on the marker on the ground vehicle.

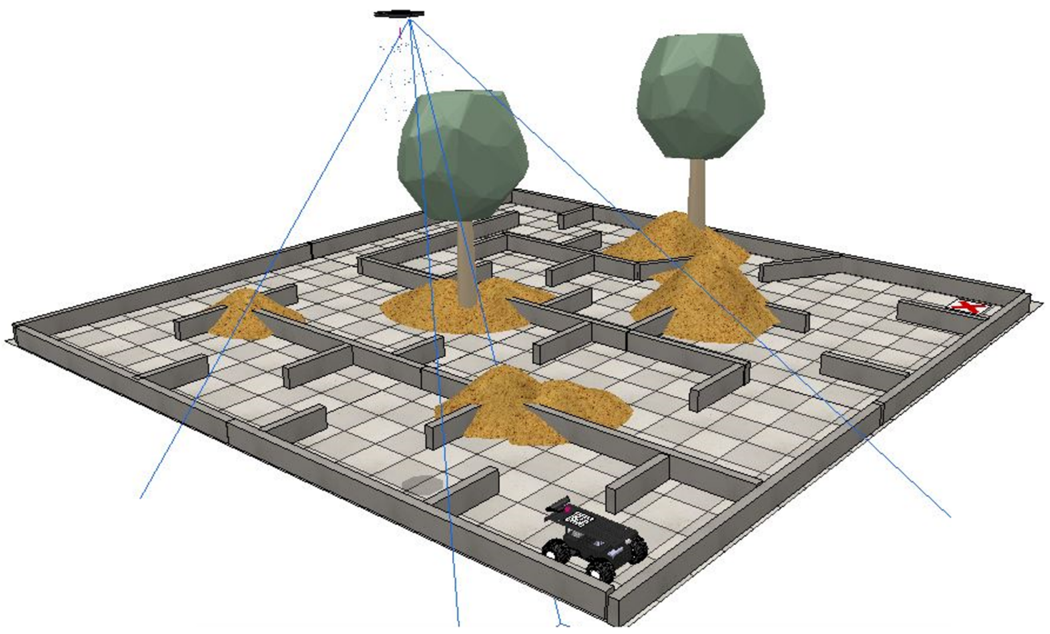


FIGURE 5.7: V-rep Test Scene View 1

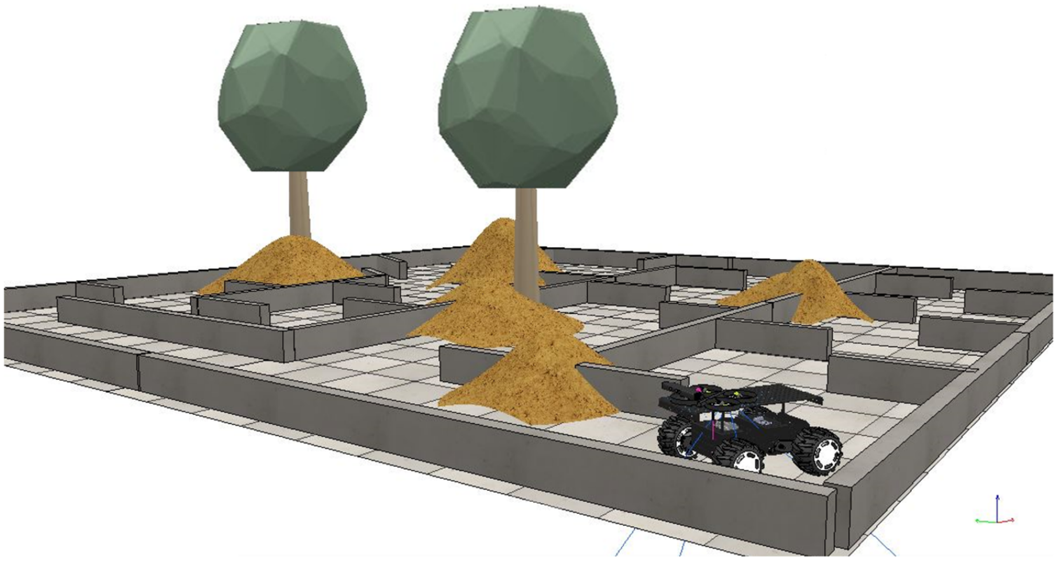


FIGURE 5.8: V-rep Test Scene View 2

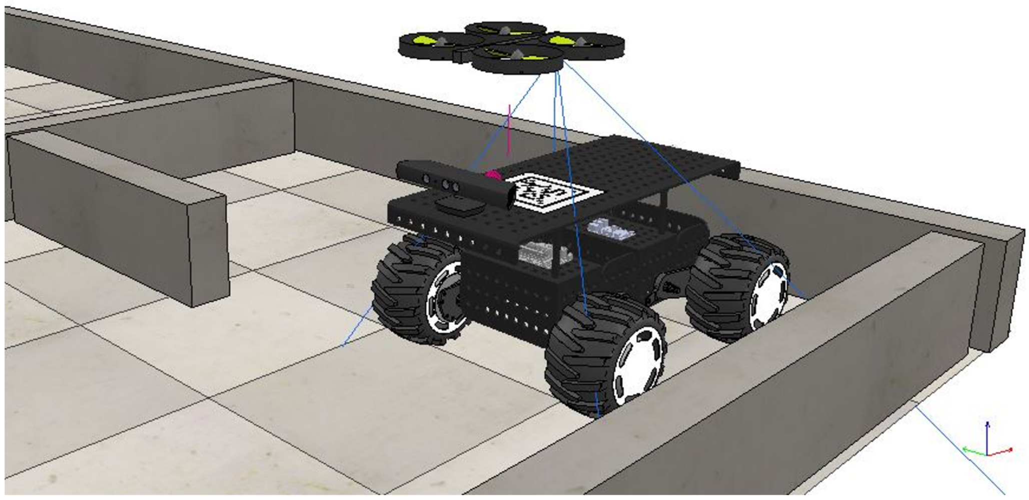


FIGURE 5.9: Utilized UAV and UGV in V-rep Test Scene

During the experiments, tests were also conducted in authentic indoor and outdoor settings. During these tests, outdoor environment flights were performed by scanning a $9000m^2$ area at an altitude of 6 to 10 meters. At an altitude of 1.5 to 2 meters, a $6m^2$ area was scanned in an indoor environment.

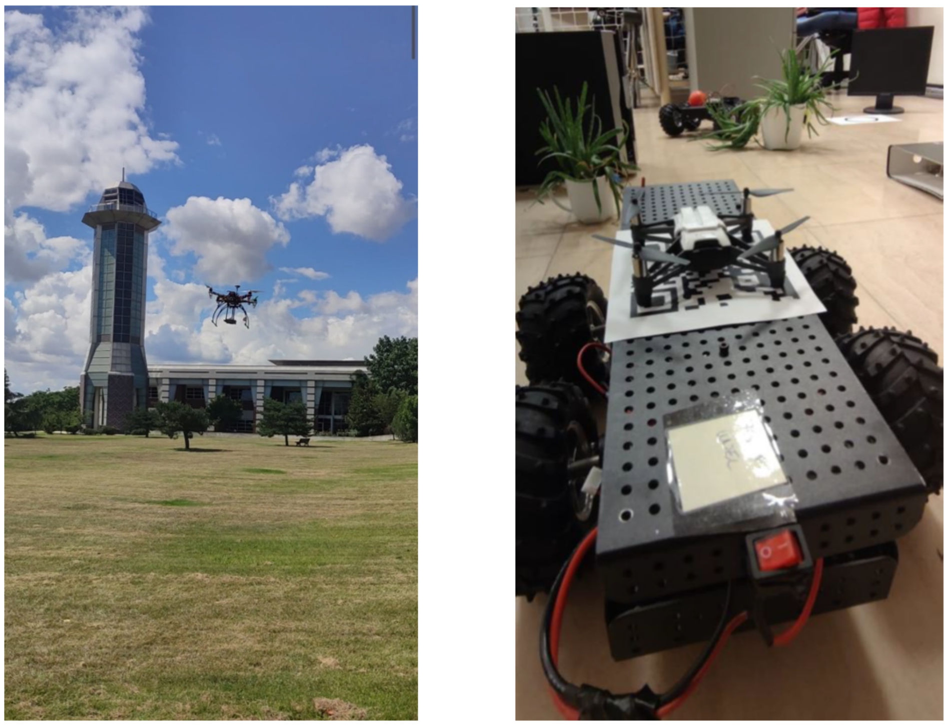


FIGURE 5.10: Outdoor and Indoor Actual Test Environments

5.2 Results and Discussion

5.2.1 Results of Image-based Path Planning Algorithms on Topographic Map

The proposed path planning algorithm and the well-known probabilistic and deterministic algorithms which are PRM and A*, are implemented to the V-rep based simulation environment and the actual environment established in laboratory. During the experiment the hyperparameters of stereo depth reconstruction method are never changed and selected as $\omega = 50$, $WindowSize = 10$. Similarly, the sensitivity value of the circle detection algorithm has not been changed as well as the thresholds of the radius; and the same marker used top of the UGV for pose estimation. Throughout all of these experiments, a Lenovo E490 ThinkPad with 16GB of RAM is used as a work-station. The initial and end points of the drawn lines by Hough line detection method, are indicated with red cross signs. Similarly, the corners detected on disparity map are represented with red asterisks, and the corners found from the original image are showed with green plus signs on the figures below, after made the elimination according to pixel values as explained in the section 4.1.1. The acquired results indicate that the A* approach requires either longer processing time or stronger computational capacity, but the PRM algorithm does not always provide the optimal path since it employs random points, despite the fact that it locates the paths in a shorter amount of time. The computing time and energy demand of the PRM algorithm are proportional to the number of thrown way-points. In addition, neither of these two well-known path planning algorithms take the depth information of the ground into account. They exclusively utilize information derived from the occupancy grid form of terrain image. Therefore, it is not possible to identify hills or pits that are not visible in a single top view of image.

5.2.1.1 Scene 1 (A*, PRM, Dynamic Prog. Method, Proposed Method)

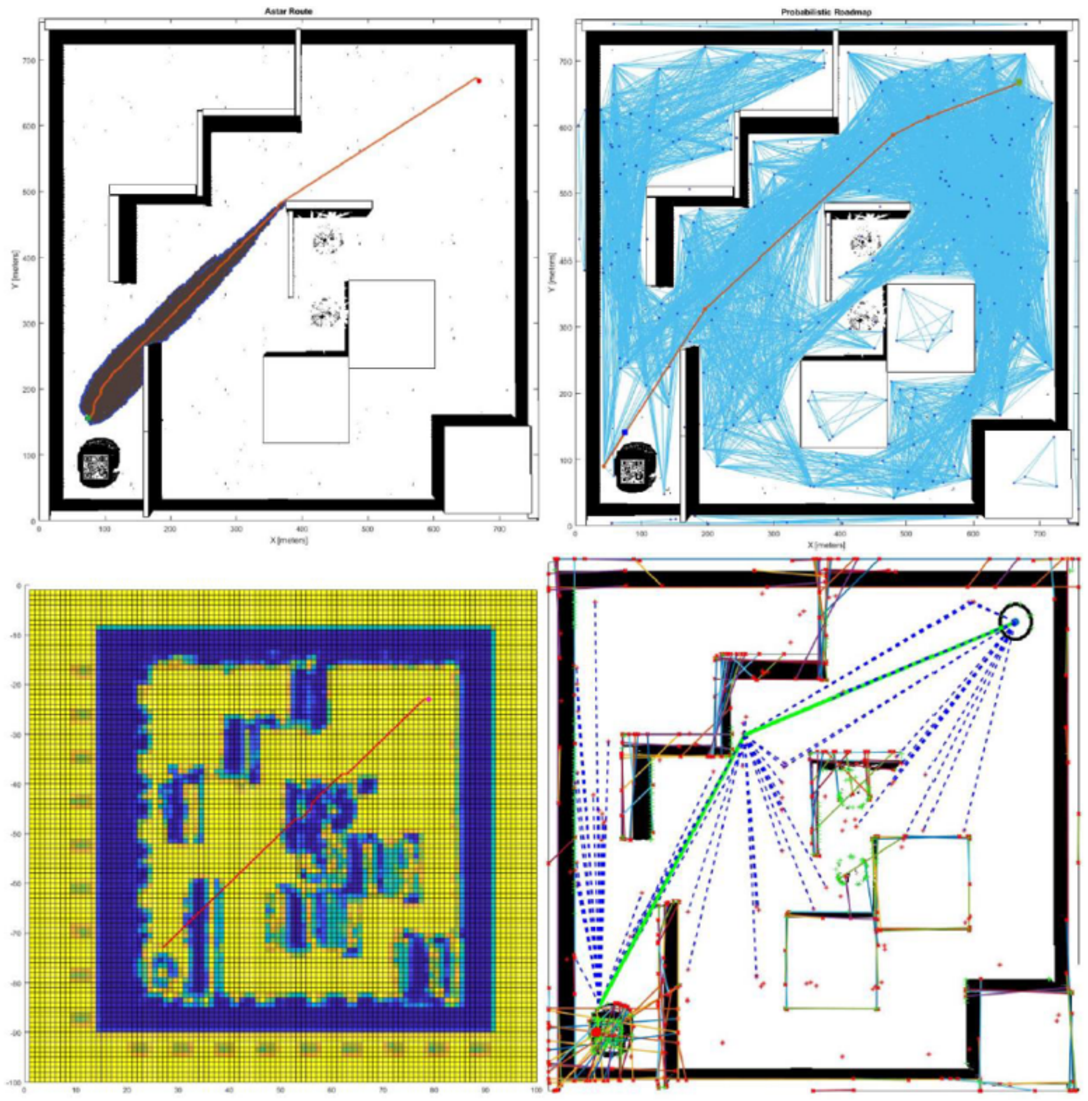


FIGURE 5.11: Found Paths in the Scene-1 (V-rep: 759x763) Left Top Image:A*, Right Top Image: PRM, Left Down Image: Dynamic Programming-based Algorithm, Right Down Image: Proposed Algorithm

Algorithm Name	Computation Time (s)
PRM	2.520640
A*	123.036743
Dynamic Programming Method	23.806219
Proposed Algorithm	1.720869

FIGURE 5.12: Computation Time for Paths in Scene 1

5.2.1.2 Scene 2 (A*, PRM, Dynamic Prog. Method, Proposed Method)

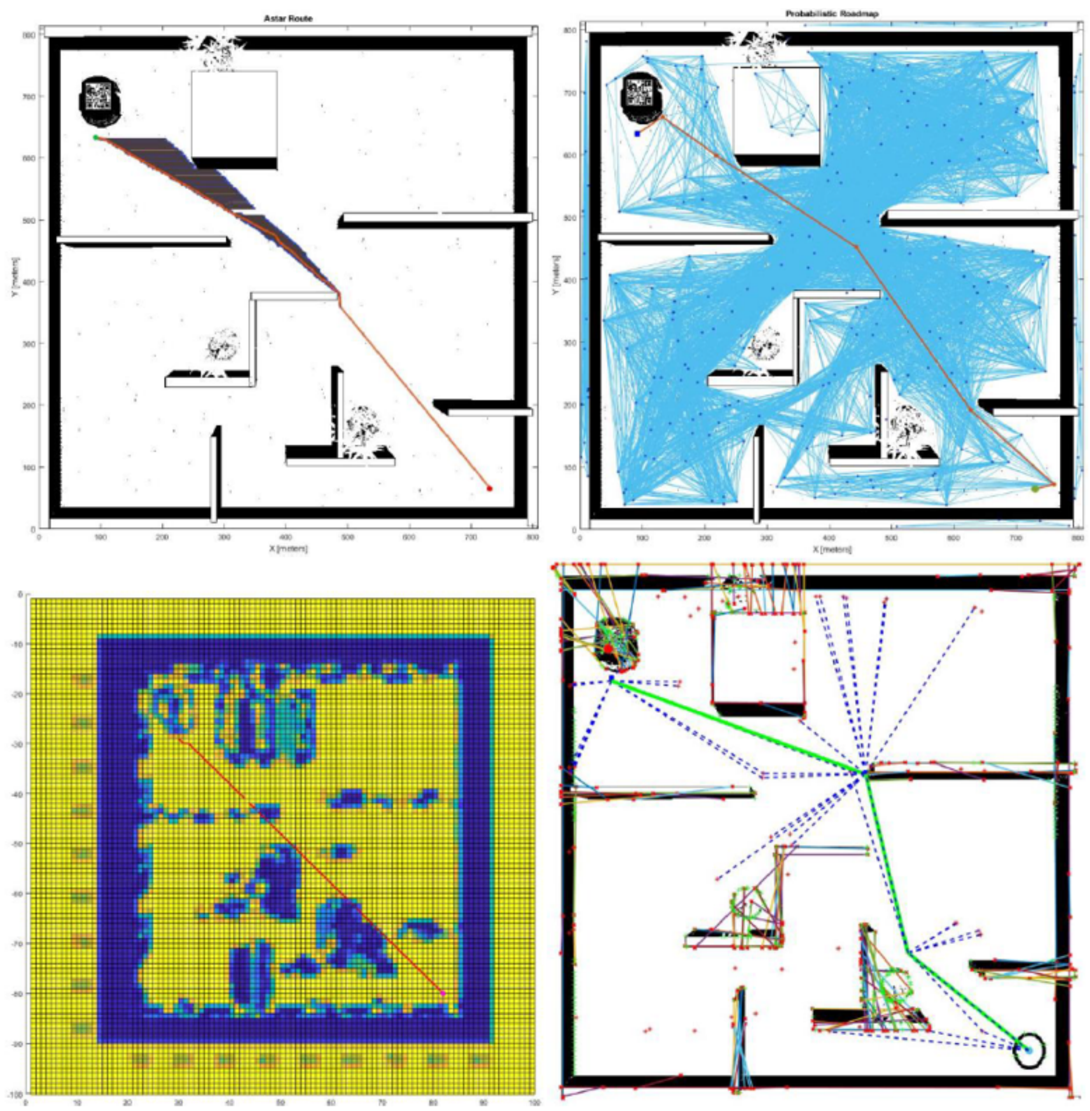


FIGURE 5.13: Found Paths in the Scene-2 (V-rep: 808x814) Left Top Image:A*, Right Top Image: PRM, Left Down Image: Dynamic Programming-based Algorithm, Right Down Image: Proposed Algorithm

Algorithm Name	Computation Time (s)
PRM	2.786833
A*	124.769130
Dynamic Programming Method	23.444926
Proposed Algorithm	1.613129

FIGURE 5.14: Computation Time for Paths in Scene 2

5.2.1.3 Scene 3 (A*, PRM, Dynamic Prog. Method, Proposed Method)

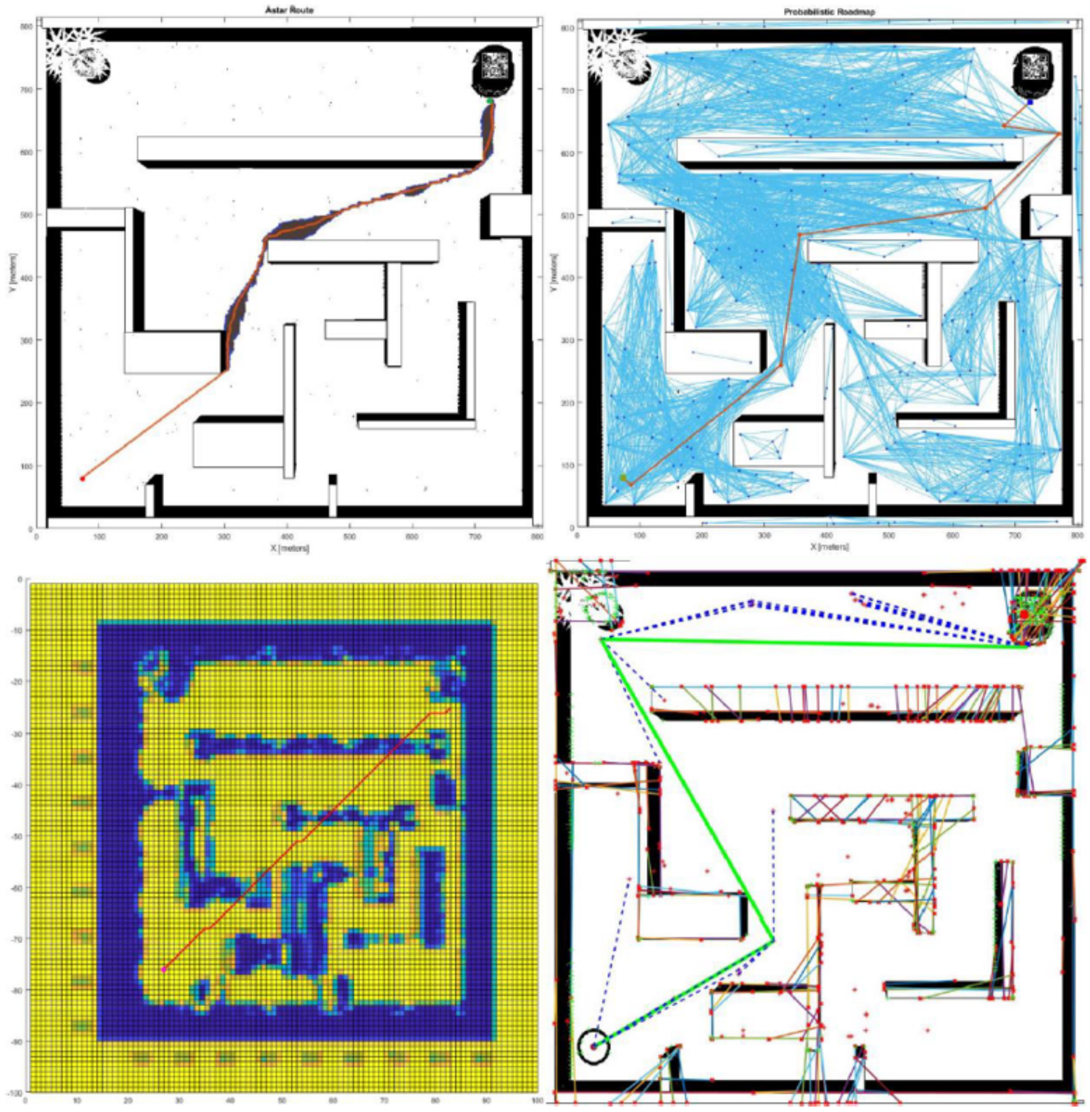


FIGURE 5.15: Found Paths in the Scene-3 ((V-rep: 808x814) Left Top Image:A*, Right Top Image: PRM, Left Down Image: Dynamic Programming-based Algorithm, Right Down Image: Proposed Algorithm

Algorithm Name	Computation Time (s)
PRM	2.695357
A*	157.781910
Dynamic Programming Method	25.237965
Proposed Algorithm	1.873277

FIGURE 5.16: Computation Time for Paths in Scene 3

5.2.1.4 Scene 4 (A*, PRM, Dynamic Prog. Method, Proposed Method)

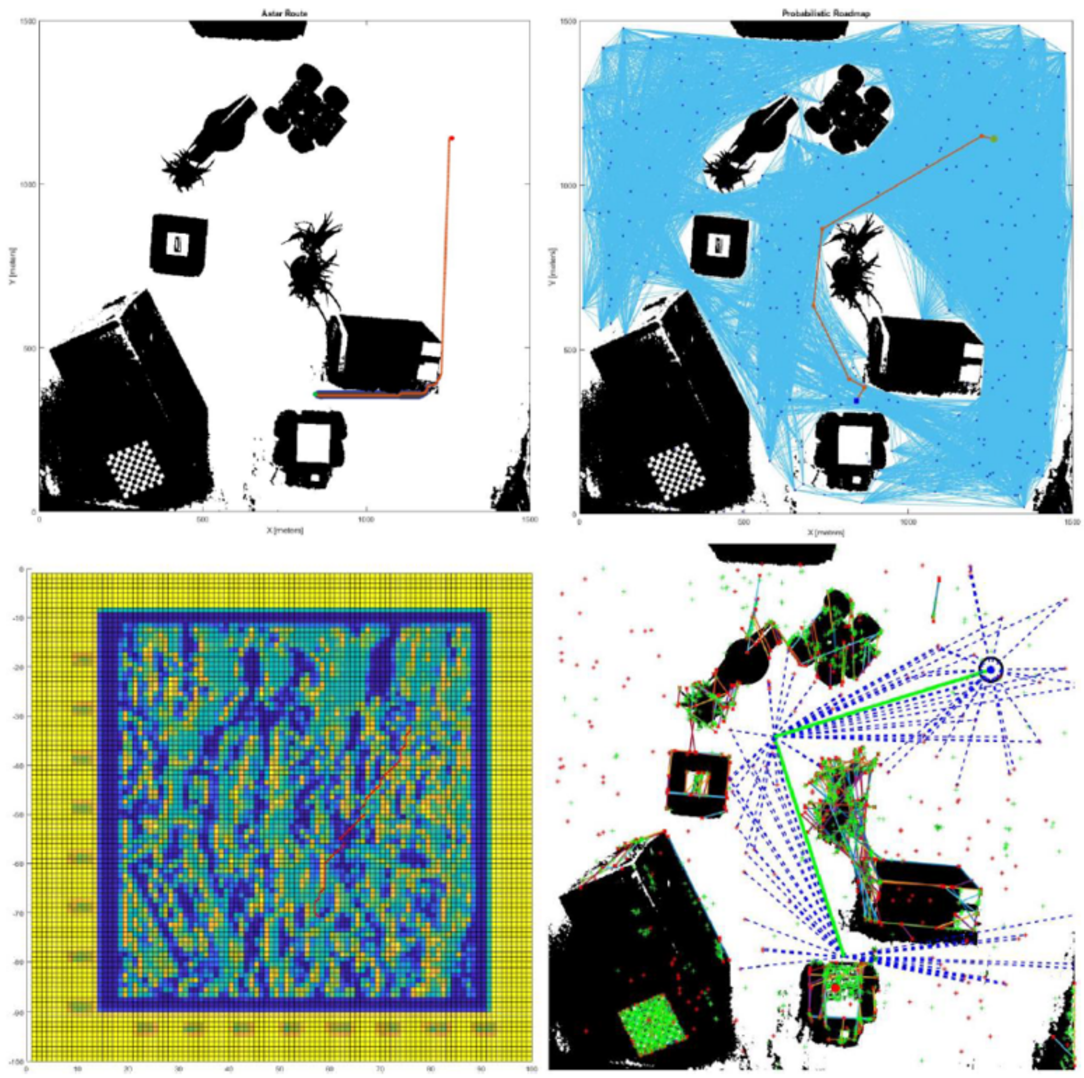


FIGURE 5.17: Found Paths in the Scene-4 (Real scene: 1500x1500) Left Top Image:A*, Right Top Image: PRM, Left Down Image: Dynamic Programming-based Algorithm, Right Down Image: Proposed Algorithm

Algorithm Name	Computation Time (s)
PRM	4.889430
A*	109.420019
Dynamic Programming Method	14.585697
Proposed Algorithm	3.379621

FIGURE 5.18: Computation Time for Paths in Scene 4

Chapter 6

Conclusion and Future Works

6.1 Summary and Contributions

In this thesis, a novel path-planning algorithm is designed using only computer vision techniques, and its performance is compared to that of well-known probabilistic and deterministic path-planning algorithms. Since path planning is one of the most important steps for autonomous mobile robot applications, over four decades have been spent developing various techniques in this area. However, as a result of recent advancements in computer vision and machine learning techniques, it is now possible to improve existing path planning algorithms or create new generation techniques that function more efficiently. In this context, an image-based path planning algorithm employing a stereo-vision-obtained terrain depth map is developed. The developed algorithm is also based on other fundamental computer vision techniques, including edge, line, corner, and circle detection algorithms, as well as pose estimation techniques using the Aruco Marker Method. After explaining the mathematical and vision-based background, the developed algorithm is compared to well-established algorithms on different virtual scenes generated by the V-rep simulation program and real physical setups generated in a laboratory setting. Then the outcomes are discussed and the contributions of the developed algorithm are highlighted.

On the other hand, because the image stitching algorithms found in the literature are based on pixel intensity or feature-based methods, their outputs are ineffective in some instances. In image pairs where the background is exactly the same color, where there is insufficient overlap, or where key feature detection is not possible, direct and feature-based stitching algorithms cannot produce positive results. IMU data are utilized by the image stitching algorithm developed and presented in the scope of the thesis. Even though the example in the case scenario utilizes images captured by a UAV, the proposed technique is applicable to any camera equipped with an IMU sensor. The results obtained with the proposed technique demonstrated that the Decoupling between the stitched image pairs is of higher quality, as is the stitching accuracy.

Experiments conducted within the scope of this thesis place a significant emphasis on in-depth analysis. This analysis was deciphered using both the stereo depth reconstruction method and the structure from motion technique, and their respective performances were compared. The stereo depth reconstruction method is a technique for analyzing depth using two cameras from a fixed location. Different machine learning classifiers have been tested and their results have been compared so that this technique can be implemented with the highest level of efficiency. Despite all of this, in the stereo depth reconstruction method, the distance between the cameras, known as the baseline, and the distance between the target object and the camera is limited, and this method does not produce a healthy result beyond a certain distance. Decoupling. Decoupling. After the features were extracted and a three-dimensional point cloud was obtained, the images captured with the moving monocular camera using the structure from motion technique were then combined to create a three-dimensional point cloud.

Vision-based motion control is another significant aspect of the research conducted for the thesis. The question of how healthy UAV control can be achieved with the aid of a simple onboard camera and a marker placed on the UAV was investigated. During this process, the position data obtained from system identification techniques system vision Blackbox was subsequently obtained with this model by utilizing one of the same specific quadrotor indoor model characteristics as a white box to compare the model's performance. Due to the consistency of the results, it has been demonstrated that the tested onboard vision system can be used for UAV motion control.

In addition, only static obstacles or an instantaneous view of dynamic obstacles on the terrain are considered by all of the analyzed path planning algorithms. Consequently, none of these algorithms provide a sufficient result for avoiding dynamic obstacles. This problem can be categorized as a local path planning problem, and various types of distance measurement sensors are typically used to make decisions in local path planning problems. However, optical flow, one of the computer vision techniques, can also be used to solve this issue.

6.2 Future Work

The research conducted under the headings vision-based motion control and terrain analysis continue with segmentation and dynamic obstacle avoidance algorithms. Using the Mask-RCNN and U-Net methods, the bumps in images captured from the actual test environment are separated into their respective classes, and a dataset is generated. It is hoped that the algorithm trained with this dataset will be able to label bumps in images captured by UAVs in rough environments and that the generated path will be based on this information.

On the other hand, estimating the motion of objects moving on the terrain using the optical flow algorithm and UAV-captured images is also among the most important future work.

Bibliography

[1] T. Miki, P. Khrapchenkov, and K. Hori, "Uav/ugv autonomous cooperation: Uav assists ugv to climb a cliff by attaching a tether," in *2019 International Conference on Robotics and Automation (ICRA)*, pp. 8041–8047, IEEE, 2019.

[2] K. Asadi, A. K. Suresh, A. Ender, S. Gotad, S. Maniyar, S. Anand, M. Noghabaei, K. Han, E. Lobaton, and T. Wu, "An integrated ugv-uav system for construction site data collection," *Automation in Construction*, vol. 112, p. 103068, 2020.

[3] X. a'Xing, Z. Jin, F. Haolong, Z. Tao, and L. Dongjie, "Vision-based map building and path planning method in unmanned air/ground vehicle cooperative systems," *The Journal of Engineering*, vol. 2020, no. 13, pp. 520–525, 2020.

[4] M. J. Schuster, M. G. Müller, S. G. Brunner, H. Lehner, P. Lehner, R. Sakagami, A. Dömel, L. Meyer, B. Vodermayer, R. Giubilato, *et al.*, "The arches space-analogue demonstration mission: Towards heterogeneous teams of autonomous robots for collaborative scientific sampling in planetary exploration," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 5315–5322, 2020.

[5] T. Miki, P. Khrapchenkov, and K. Hori, "Uav/ugv autonomous cooperation: Uav assists ugv to climb a cliff by attaching a tether," in *2019 International Conference on Robotics and Automation (ICRA)*, pp. 8041–8047, IEEE, 2019.

[6] S. Hood, K. Benson, P. Hamod, D. Madison, J. M. O'Kane, and I. Rekleitis, "Bird's eye view: Cooperative exploration by ugv and uav," in *2017 International Conference on Unmanned Aircraft Systems (ICUAS)*, pp. 247–255, IEEE, 2017.

[7] J. H. Kim, J.-W. Kwon, and J. Seo, "Multi-uav-based stereo vision system without gps for ground obstacle mapping to assist path planning of ugv," *Electronics Letters*, vol. 50, no. 20, pp. 1431–1432, 2014.

[8] Y. Zhou, S. Lai, H. Cheng, A. H. M. Redhwan, P. Wang, J. Zhu, Z. Gao, Z. Ma, Y. Bi, F. Lin, *et al.*, "Toward autonomy of micro aerial vehicles in unknown and global positioning system denied environments," *IEEE Transactions on Industrial Electronics*, vol. 68, no. 8, pp. 7642–7651, 2020.

[9] A. Niewola and L. Podsedkowski, "Psd–probabilistic algorithm for mobile robot 6d localization without natural and artificial landmarks based on 2.5 d map and a new type of laser scanner in gps-denied scenarios," *Mechatronics*, vol. 65, p. 102308, 2020.

[10] P. Kim, L. C. Price, J. Park, and Y. K. Cho, "Uav-ugv cooperative 3d environmental mapping," in *Proceedings of the ASCE International Conference on Computing in Civil Engineering*, 2019.

[11] H. A. Lauterbach and A. Nüchter, "Preliminary results on instantaneous uav-based 3d mapping for rescue applications," in *2018 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, pp. 1–2, IEEE, 2018.

[12] Y. Yue, C. Zhao, R. Li, C. Yang, J. Zhang, M. Wen, Y. Wang, and D. Wang, "A hierarchical framework for collaborative probabilistic semantic mapping," in *2020 IEEE international conference on robotics and automation (ICRA)*, pp. 9659–9665, IEEE, 2020.

[13] M. T. Ort, *MapLite: autonomous navigation in rural environments without detailed prior maps.* PhD thesis, Massachusetts Institute of Technology, 2020.

[14] Z. Zhan, W. Jian, Y. Li, and Y. Yue, "A slam map restoration algorithm based on submaps and an undirected connected graph," *IEEE Access*, vol. 9, pp. 12657–12674, 2021.

[15] D. Ferguson, M. Likhachev, and A. Stentz, "A guide to heuristic-based path planning," in *Proceedings of the international workshop on planning under uncertainty for autonomous systems, international conference on automated planning and scheduling (ICAPS)*, pp. 9–18, 2005.

[16] H. Noborio and Y. Nishino, "Image-based path-planning algorithm on the joint space," in *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No. 01CH37164)*, vol. 2, pp. 1180–1187, IEEE, 2001.

[17] R. Miranda-Colorado and L. T. Aguilar, "Robust pid control of quadrotors with power reduction analysis," *ISA transactions*, vol. 98, pp. 47–62, 2020.

[18] A. Hamza, A. H. Mohamed, and A. El-Badawy, "Robust h-infinity control for a quadrotor uav," in *AIAA SCITECH 2022 Forum*, p. 2033, 2022.

[19] K. Pan, Y. Chen, Z. Wang, H. Wu, and L. Cheng, "Quadrotor control based on self-tuning lqr," in *2018 37th Chinese Control Conference (CCC)*, pp. 9974–9979, IEEE, 2018.

[20] L. Martins, C. Cardeira, and P. Oliveira, "Feedback linearization with zero dynamics stabilization for quadrotor control," *Journal of Intelligent & Robotic Systems*, vol. 101, no. 1, pp. 1–17, 2021.

[21] M. Labbadi and M. Cherkaoui, "Robust adaptive backstepping fast terminal sliding mode controller for uncertain quadrotor uav," *Aerospace Science and Technology*, vol. 93, p. 105306, 2019.

[22] M. Labbadi and M. Cherkaoui, "Robust adaptive nonsingular fast terminal sliding-mode tracking control for an uncertain quadrotor uav subjected to disturbances," *ISA transactions*, vol. 99, pp. 290–304, 2020.

[23] H. Mo and G. Farid, "Nonlinear and adaptive intelligent control techniques for quadrotor uav–a survey," *Asian Journal of Control*, vol. 21, no. 2, pp. 989–1008, 2019.

[24] P. Shukla and A. Kumar, "Comparative performance analysis of various control algorithms for quadrotor unmanned aerial vehicle system and future direction," in *2020 International Conference on Electrical and Electronics Engineering (ICE3)*, pp. 498–501, IEEE, 2020.

[25] G. Farid, M. Hongwei, S. M. Ali, and Q. Liwei, "A review on linear and nonlinear control techniques for position and attitude control of a quadrotor," *Control and Intelligent Systems*, vol. 45, no. 1, pp. 43–57, 2017.

[26] S. Sun, A. Romero, P. Foehn, E. Kaufmann, and D. Scaramuzza, "A comparative study of nonlinear mpc and differential-flatness-based control for quadrotor agile flight," *IEEE Transactions on Robotics*, 2022.

[27] H. Housny, H. El Fadil, *et al.*, "Fuzzy pid control tuning design using particle swarm optimization algorithm for a quadrotor," in *2019 5th International Conference on Optimization and Applications (ICOA)*, pp. 1–6, IEEE, 2019.

[28] J. Gómez-Avila, C. López-Franco, A. Y. Alanis, and N. Arana-Daniel, "Control of quadrotor using a neural network based pid," in *2018 IEEE Latin American Conference on Computational Intelligence (LA-CCI)*, pp. 1–6, IEEE, 2018.

[29] N. E. Gmili, M. Mjahed, A. E. Kari, and H. Ayad, "Intelligent pso-based pds/pids controllers for an unmanned quadrotor," *International Journal of Intelligent Engineering Informatics*, vol. 6, no. 6, pp. 548–568, 2018.

[30] D. Park, H. Yu, N. Xuan-Mung, J. Lee, and S. K. Hong, "Multicopter pid attitude controller gain auto-tuning through reinforcement learning neural networks," in *Proceedings of the 2019 2nd International Conference on Control and Robot Technology*, pp. 80–84, 2019.

[31] S. Yang and B. Xian, “Energy-based nonlinear adaptive control design for the quadrotor uav system with a suspended payload,” *IEEE Transactions on Industrial Electronics*, vol. 67, no. 3, pp. 2054–2064, 2019.

[32] H. Ghadiri, M. Emami, and H. Khodadadi, “Adaptive super-twisting non-singular terminal sliding mode control for tracking of quadrotor with bounded disturbances,” *Aerospace Science and Technology*, vol. 112, p. 106616, 2021.

[33] E. Okyere, A. Bousbaine, G. T. Poyi, A. K. Joseph, and J. M. Andrade, “Lqr controller design for quad-rotor helicopters,” *The Journal of Engineering*, vol. 2019, no. 17, pp. 4003–4007, 2019.

[34] E. Kuantama, I. Tarca, and R. Tarca, “Feedback linearization lqr control for quadcopter position tracking,” in *2018 5th International Conference on Control, Decision and Information Technologies (CoDIT)*, pp. 204–209, IEEE, 2018.

[35] C. MASSÉ, O. GOUGEON, D.-T. NGUYEN, and D. SAUSSIÉ, “Modeling and control of a quadcopter flying in a wind field: A comparison between lqr and structured control techniques,” in *2018 International Conference on Unmanned Aircraft Systems (ICUAS)*, pp. 1408–1417, IEEE, 2018.

[36] E. Adel, M. Elmogy, and H. Elbakry, “Image stitching based on feature extraction techniques: a survey,” *International Journal of Computer Applications*, vol. 99, no. 6, pp. 1–8, 2014.

[37] T. Tao, J. C. Koo, and H. R. Choi, “A fast block matching algorthim for stereo correspondence,” in *2008 IEEE Conference on Cybernetics and Intelligent Systems*, pp. 38–41, IEEE, 2008.

[38] R. A. Hamzah, R. Abd Rahim, and Z. M. Noh, “Sum of absolute differences algorithm in stereo correspondence problem for stereo matching in computer vision application,” in *2010 3rd International Conference on Computer Science and Information Technology*, vol. 1, pp. 652–657, IEEE, 2010.

[39] J. Zbontar and Y. LeCun, “Computing the stereo matching cost with a convolutional neural network,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1592–1599, 2015.

[40] D. Hirner and F. Fraundorfer, “Fc-dcnn: A densely connected neural network for stereo estimation,” in *2020 25th International Conference on Pattern Recognition (ICPR)*, pp. 2482–2489, IEEE, 2021.

[41] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, “Densely connected convolutional networks,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4700–4708, 2017.

[42] C. Bailer, K. Varanasi, and D. Stricker, “Cnn-based patch matching for optical flow with thresholded hinge embedding loss,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3250–3259, 2017.

[43] S. A. Iz and M. Unel, “An image-based path planning algorithm using a uav equipped with stereo vision,” in *IECON 2022 – 48th Annual Conference of the IEEE Industrial Electronics Society*, pp. 1–6, 2022.

[44] T. Jiřinec, “Stabilization and control of unmanned quadcopter,” 2011.

[45] Y. M. Al-Younes, M. A. Al-Jarrah, and A. A. Jhemi, “Linear vs. nonlinear control techniques for a quadrotor vehicle,” in *7th International Symposium on Mechatronics and its Applications*, pp. 1–10, IEEE, 2010.

[46] A. Nemati and M. Kumar, “Non-linear control of tilting-quadcopter using feedback linearization based motion control,” in *Dynamic Systems and Control Conference*, vol. 46209, p. V003T48A005, American Society of Mechanical Engineers, 2014.