UNIVERSITÉ DU LUXEMBOURG

PhD-FSTC-2018-69
The faculty of Sciences, Technology and Communication

DISSERTATION

Presented on 26/09/2018 in Luxembourg

to obtain the degree of

DOCTEUR DE L'UNIVERSITÉ DU LUXEMBOURG
EN SCIENCES DE L'INGÉNIEUR

by

Arun ANNAIYAN
Born on 30 November 1988 in Cuddalore (India)

Collision-Free Navigation of Small UAVs in Complex Urban Environment

Dissertation Defence Commitee

Dr. Ing Holger Voos, Dissertation Supervisor
Professor, Université du Luxembourg

Dr. Ing Francesco Viti, Chairman
Associate Professor, Université du Luxembourg

Dr. Ing Mohamed Darouach, Vice Chairman
Professor, Université de Lorraine, IUT de Longwy, France

Dr. Harouna Souley Ali,
Associate Professor, Université de Lorraine, IUT de Longwy, France

Dr. Ing Miguel Olivares Mendez
Research scientist, Université du Luxembourg

UNIVERSITY OF LUXEMBOURG

Abstract

Faculty Name
INTERDISCIPLINARY CENTRE FOR SECURITY,RELIABILITY AND TRUST

Doctor of Philosophy

by

Small unmanned aerial vehicles (UAVs) are expected to become highly innovative solutions for all kind of tasks such as transport, surveillance, inspection or guidance, and many commercial ideas already exist. Herein, small multi rotor UAVs are preferred since they are easy to construct and to fly, at least in wide open spaces. However, many UAV business cases are foreseen in complex urban environments which are very challenging from the perspective of UAV flight. Our work focuses on the autonomous flight and collision-free navigation in an urban environment, where GPS is still considered for localization but where variations in the accuracy or temporary unavailability of GPS position data is explicitly considered. Herein, urban environments are challenging because they require flight nearby large structures and also nearby moving obstacles such as humans and other moving objects, at low altitudes or in very narrow spaces and thus also in areas where GPS (global positioning system) position data might temporarily be very inaccurate or even not available. Therefore we designed a custom stereo camera with adjustable base length for the perception of the possible potential obstacles in the unknown outdoor environment. In this context the optimal design and sensitivity parameters are investigated in outdoor experiments. Using the stereo images, graph based SLAM approach is used for online three dimensional mapping of the static and dynamic environment. For the memory efficiency incremental online loop closure detection using bag of words method is implemented here. By having the three dimensional map, the cost of the cell and its transition calculated in real time by the modified D* lite which will search and generate three dimensional collision free path planning. Experiments of the 3D mapping and collision free path planning are conducted using small UAV in outdoor scenario. The combined experimental results of real time mapping and path planning demonstrated that the three dimensional collision free path planning is able to handle the real time computational constraints while maintaining safety distance.

Acknowledgements

First of all, I would like to thank Professor Holger Voos for having offered me the possibility of doing a PhD under his supervision. He provided all the necessary resources to achieve the present work. Being part of Automation and Robotics research group was a highly appreciated experience and led to many happy moments. My sincere thanks to my committee member Prof.Francesco Viti for giving me the constructive feedback during the discussions.

Special thanks for Dr. Somasundar Kannan for sharing his knowledge on various topics of science and motivating me to do research in all the situations. He guided me from the beginning of this work and provided me the opportunity for many fruitful discussions. I have learned numerous scientific concepts through his explanations. Big thanks to Jan Dentler for his wonderful nature of helping me in all circumstances. His help during my outdoor experiments was really tremendous. Moreover in office his positive energy and down to earth attitude was supportive all over the time.

A very special gratitude goes out to FNR - Fonds national de la Recherche (Luxembourg) for funding this work through AFR-PhD grant CoNAV (10156266). This grant helps me a lot to continue my research work very smooth. Also I'm thanking luxembourg people I have met during my stay in the country. They are so welcoming as well as encouraging and introduce me to new culture in a nice way.

Finally I am grateful to all my colleagues in SnT and FSTC for their encouraging spirit and smile at everyday work. I thank all my technicians who supported me to complete the hardware related work for all my experiments in the lab.

Contents

List of Figures

List of Tables

Abbreviations

2D	**T**wo **D**imension
3D	**T**hree **D**imension
BRIEF	**B**inary **R**obust **I**ndependent **E**lementary **F**eatures
BRISK	**B**inary **R**obust **I**nvariant **S**calable **K**eypoints
BOW	**B**ag **O**f **W**ords
COTS	**C**ommerically **O**ff **T**he **S**helf
DOG	**D**ifference **O**f **G**aussian
FAST	**F**eatures from **A**ccelerated **S**egment **T**est
GFTT	**G**ood **F**eatures **T**o **T**rack
GPS	**G**lobal **P**ositioning **S**ystem
IMU	**I**nertial **M**easurement **U**nit
INS	**I**nertial **N**avigation **S**ystem
NNS	**N**earest **N**eighbour **S**earch
ORB	**O**riented **F**AST and **R**otated BRIEF
RANSAC	**RA**ndom **SA**mple **C**onsensus
ROS	**R**obotic **O**perating **S**ystem
SAD	**S**um of **A**bsolute **D**ifferences

SIFT	**S**cale **I**nvariant **F**eature **T**ransform
SLAM	**S**imultaneous **M**apping **A**nd **L**ocalization
SSD	**S**um of **S**quare **D**ifferences
SURF	**S**peed **U**p **R**obust **F**eatures
SVO	**S**tereo **V**isual **O**dometry
UAV	**U**nmanned **A**erial **V**ehicle
USB	**U**niversal **S**erial **B**us

To my family Amutha, Annaiyan, Lakshmipathy.

Chapter 1

Introduction

1.1 Unmanned Aerial Vehicles

Small Unmanned Aerial Vehicles (UAV) have gained considerable importance in the defense and commercial sectors in the last 20 years. According to the report [1], the UAV industry business will be valued up to 127.3 billion USD and it opens up a lot of potential possible applications in the future. More specifically small UAVs flying in lower altitudes have been seen to be of great potential in many applications. Similarly a large focus in current research is placed on the idea to increase the autonomy of the existing UAV solutions and systems. As shown in Figure 1.1 different types of small UAVs with different configurations are available on the market as commercially off the shelf (COTS) products. Depending on the application and scenario, the configuration can be custom made or tuned accordingly. Small UAVs can provide cost effective solutions. For example, solution to get aerial videos and images by small UAVs are cheaper compared to the images provided by satellites which also often have low resolution and are weather dependent. Some of the applications include monitoring the crop growth in the agricultural sector, surveys of the channelized pipelines in the energy sector for maintenance as well as damage assessment. Since small UAVs are capable of flying in confined space, they have the ability to monitor the situation during emergency and disaster. Nowadays security companies are extensively using them to acquire the out of range details where a static camera cannot be used from a fixed location. Damage assessment or structural inspection using small UAVs is performed by logistic companies in order to identify the cracks in the outer shell of ships, bridge monitoring, road damages and collapse inspection during accidents. Media and entertainment industry uses small UAVs for photography and filming, while the communication sector uses them for creating temporary networks where there is not enough ground towers to provide

signal. In transportation and e-commerce sectors it is helping in delivering goods. This clearly shows that the usage of small low altitude UAVs has dramatically increased in civil applications. From the perspective of manufacturing and design configuration, the

FIGURE 1.1: Different types of small UAVs
Left to right : Octorotor, hexarotor, gimbal drone, delta-wing, fixed-wing, Quad-rotor

UAV sector is also evolving according to the feedback coming from the existing solutions. From Figure 1.1 we can infer that the size, shape, payload capabilities of UAVs are getting different. This is due to the fact that every sector has its own specific requirements to increase the usage of small UAVs like more payload capability, flight speed, accurate sensing and perception, flexible maneuvering, aerial manipulation, cooperative flight, more endurance etc. For example, a rotorcraft has hovering capabilities and it can maneuver while the fixed wing can navigate fast but cannot hover. In infrastructure inspection the requirement is to acquire high resolution stable video in a vast area. However, the existing solution needs atleast one pilot with additional people depending on the application to conduct the mission without accidents. If the man power is reduced through the autonomy of the small UAV, then it is considered to be a major advantage for the user. While seeing the future of small UAVs with all the above mentioned successful applications, there are still open questions in the case of mission safety and autonomy in operations involving confined spaces. The unsolved open research questions among other are:

- Do the small UAVs have sufficient autonomy to complete the mission without trained pilot?

- How does autonomous operation works in a practical scenario where both static and dynamic objects are present?
- What are currently available capabilities in autonomous operations? Can a UAV perform maneuvering in close environments involving people, in an "everyday" scenario?

Since the word "autonomous" is often used in the robotics community, we would like to provide atleast a working definition. According to International Organization of standardization[2] the robot autonomy is defined as,

Definition 1.1. (Autonomy) A robot which has the ability to perform intended task based on the current state and sensing without human intervention.

So the small UAVs which can perform an intended task without human intervention will be considered as an autonomous small UAV. One of the ways to achieve it, is by considering a more realistic and practical approach. The practical requirements are,

- Real time onboard computation.
- Real time decision making.
- Safety aspects.

This thesis concentrates on the autonomy of Vertical Take Off and Landing (VTOL) UAVs. The concerned open research questions are listed in section 1.1. The autonomous module includes submodules such as mapping and localization, collision avoidance, path planning as shown in Figure 1.2. With these submodules the small UAVs are capable of flying in lower altitudes autonomously. But there are challenges when it comes to reality and practicality. Because in the urban environment complex manoeuvring in real time is still difficult due to the uncertainties associated with it.

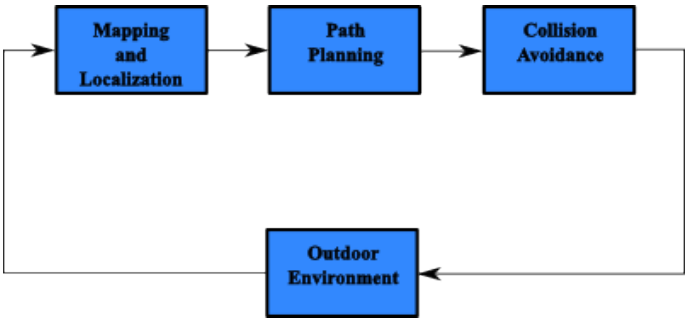


FIGURE 1.2: Primary modules for autonomous navigation

1.2 Problem statement

Small UAVs navigating in low altitudes require a high level of autonomy and sophisticated sensing capabilities. When a UAV is hovering above the ground at just a few meters the GPS information is not reliable and accurate. It is possible to combine the GPS information with the already measured map to estimate the collision free path, but this solution might not provide new object updates with three dimensional details. So this means the altitude information of the structures like buildings, trees, power lines and poles cannot be directly obtained up to date. The practical urban environment consists of buildings, moving humans, automobiles as shown in Figure 1.3. The only known information here is the initial position and goal position of the UAV, while the rest of the details are partially unknown. Then navigating in this environment as seen

FIGURE 1.3: Example of the practical urban environment

in Figure 1.3 which is highly dynamic and unstructured requires the acquisition of real time data. Through the acquired real-time data we have to retrieve the possible threats for crashing, for example the size, shape, position of the obstacles with its relative distance. In order to consider the dynamic objects, frequent update and data refreshment is needed to capture the modification happening at each time instance. With the use of real-time information, a new collision-free path should be planned and re-planned online if necessary. Therefore, the small UAVs in low altitudes that needs to be used in the urban environment should have a high level of autonomy to achieve the mission safely with efficiency.

1.3 Objectives

The objectives of this thesis are briefly summarized below,

- The primary objective is to develop a real-time collision avoidance module for a small UAV.
- Exploring the environment by mapping and estimating the possible obstacles including moving objects.
- Collision free path planning and re-planning in the case of dynamic objects.
- Implementation and execution in the on-board computer with the available memory mounted on the small UAV
- Validating the developed module/package through real time experiments in unknown as well as dynamic outdoor environments.

1.4 Solution approach

In this thesis our proposed solution approach is to endow certain level of autonomy which can be applied in a real-time outdoor environment. To meet existing challenges small UAVs require additional exteroceptive sensors to sense and perceive the environment. Since vision-based sensors are considered to be promising for small UAVs performing real-time processing, a stereo camera is chosen here. Due to its reliable performance as observed in the literature with low cost, a forward looking custom made stereo camera is used here. Using the calibrated stereo camera, point clouds are generated online where the distance to objects are obtained. That is not enough for the collision avoidance, because the trajectory of the vehicle is not yet estimated.

Therefore Simultaneous Localization and Mapping (SLAM) is applied here. Online SLAM with memory efficiency and accuracy are the requirements here. Recent advancements in graph based SLAM approaches are evidently powerful and suitable for real time processing. For memory constraints and computational power requirements, a loop closure detector using the bag of words methods is designed to reject the previously estimated locations which then directly influences the memory of the on-board computer. The graph SLAM forms a set of nodes to store the poses where the g20 optimization algorithm reduces the error. The camera motion is estimated by the 3D-3D correspondence which forms the stereo odometry. The map which has started to generate will be represented through octomap which is proven to be memory efficient

using the probabilistic occupancy estimation with multi resolution. This octomap gives the occupancy grid where the position of the obstacle and free space will be obtained. To identify the collision free path from the octomap three dimensional path planning and replanning (PRP) algorithm is developed based on the D* lite algorithm. This will calculate the path every iteration through the graph search, and if the cell cost changes during the process, it will dynamically recalculate from the current position of the vehicle. This will calculate the shortest path from the current position to the goal position. The designed algorithm will also dynamically change based on the cost of each cell from the octomap.

1.5 Contributions

The key contributions of this work are summarized below,

- *A first key contribution includes the design and development of a custom made stereo camera.*
- *Development and implementation of a graph SLAM with SURF detector extended by a loop closure detection method.*
  This contribution consists of development of a graph based SLAM with SURF detector for feature generation along with Loop Closure Detection using the online bag-of-words approach. The whole package is tested and analyzed in static as well as dynamic environments and dense forest like environment to make it robust and stable.
- *Developed D* lite based three dimensional Planning and re-planning method(PRP)*
  This contribution includes the development of efficient three dimensional path planning and re-planning technique based on D*lite algorithm in three dimensional environment. The developed package is then implemented in numerous experiments.
- *Combined the Graph SLAM and Dynamic path planning to check the computational power*
  This contribution considers the combination of the perception and planner package in the onboard computer and executing it in parallel to check the computational power. The combined package is launched and tested through the different experiments. The outcome of the experiments satisfies the memory constraints as required.

- *The efficiency and accuracy of this combined solution is analyzed from the real time experimental data and its comparison is highlighted.*

1.6 Structure of this document

Following the introduction chapter the remaining of the thesis is structured as below:

- **Chapter 2** explains the existing methods and state of the art with respect to the small UAV where we discuss the techniques used with sensors, Simultaneous Localization and Mapping(SLAM) algorithms, visual odometry for the localization, path planning and its various scenario-based techniques and its advantages and disadvantages.
- **Chapter 3** describes the design and development of the custom made stereo camera. Futhermore it gives the brief theoretical background about the stereo vision.
- **Chapter 4** shows the graph based mapping and localization algorithm. Besides it also explains the bag of words approach to create dictionary for the loop closure detection method. The tested results with four datasets are presented and discussed in the context of real time processing.
- **Chapter 5** includes the three dimensional path planning and replanning (PRP) algorithm from the octomap. The details include the graph search in the occupancy grid with cost changes during the online process and dynamic planning.
- **Chapter 6** shows the experimental results and analysis of the combined online mapping and localization along with path planning framework in a real time three dimensional environment. This gives the details of the computational power, memory efficiency of the whole framework through flight experiments.
- **Chapter 7** concludes the thesis by highlighting the contributions, introduced approach and the future perspectives of this research.
- **Appendix A** shows the master-slave configuration, specifications of the camera model, schematics of the base plates used for the stereo camera.
- **Appendix B** gives the specifications of the onboard computer.
- **Appendix C** describes the software and hardware used in this work.
- **Appendix D** explains the different co-ordinate systems and its transformations.

Chapter 2

Literature Review

This chapter presents the state-of-the art methods provided in the literature for small UAVs with respect to sensing and estimation, visual mapping and localization, and finally path planning. The concepts are briefly described and discussed with its advantages and disadvantages for different scenarios.

2.1 Sensors

Autonomous small UAVs navigating in a physical world require sensing capabilities to explore and model the environment. Sensing is done with the help of a suitable sensor which depends on the mission or task. When a sensor is used with small UAVs, it is often sensitive to air flow during hovering and other flight operations. This inturn leads to problems in data acquisition coming from the onboard sensors such as data ambiguity, statistical hardware noise, computational constraints, systematic errors etc. These are some of the key factors to be considered while choosing the suitable sensors.

2.1.1 Sensors for navigation of UAV

The selection of a sensor suitable for a specific mission depends on the factors such as payload capacity of the vehicle , size, environment (indoor/outdoor), reliability of available power [3], on-board computational power, endurance, amount of flight operation time. Nowadays, most of the commercially available small UAVs have an Inertial Measurement Unit (IMU) for estimation of relative position and velocity. Basically an IMU consists of three axis gyroscope and accelerometer for the measurement of rotation and linear acceleration. But the vibration and drifting of the hardware system leads to erroneous position and velocity estimation over the time. So instead of using an IMU

alone, it can be fused with other sensors based on the task [4]. Furthermore, the Global Position System (GPS) is also used successfully in outdoor missions for waypoint navigation where map of the environment is provided. However, the reliability of satellite signals in low altitude cluttered environments are still dubious. To overcome this problem, there is a need for another kind of sensor for more accuracy and efficiency. For example sound navigation and ranging (sonar) (Figure 2.1c) is also deployed in [5], [6] to find the obstacle distance by emitting the acoustic signal and measuring the feedback of its echoes. The advantage of using sonar is that it is not affected by illumination or color of the objects and even works in dark environment with dust and high moisture. While on the other hand, it normally works well with rigid materials and has a limited detection range. Also the accuracy gets low if the temperature of sonar sensor varies from 5 to 10 degrees. For example if the sonar is deployed in complex outdoor environment for autonomous navigation, the measurement accuracy is not enough to complete the task.

Figure 2.1b shows the kinect sensor [7] which is used in [8] and it is known for its

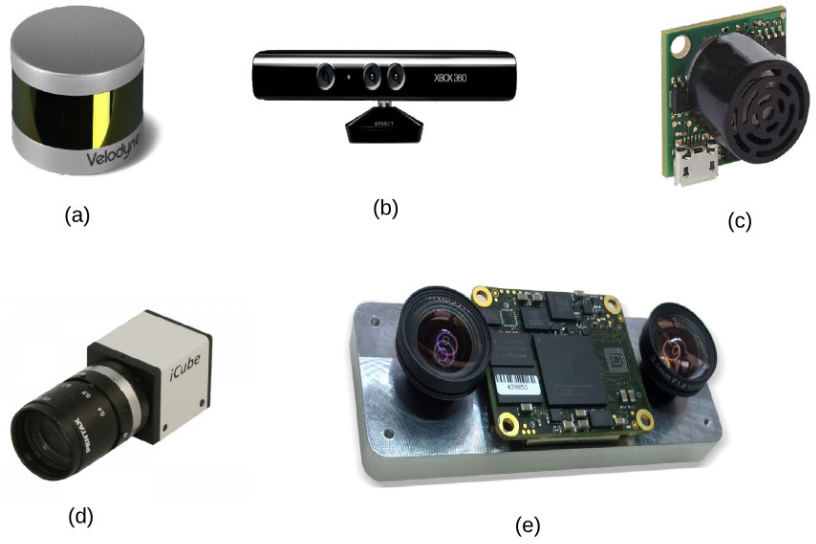


FIGURE 2.1: Different kinds of sensor used in UAV. From left to right (a)Lidar, (b)kinect, (c)ultrasonic, (d)monocular camera, (e)stereo camera

accurate 3D measurements in short range of distance. In case of long distance the data is ambiguous and not adequate enough to process. Furthermore, the usage of Lidar sensors (Figure 2.1a) in small UAVs [9] is also increasing because it provides precise 3D measurements. But the usage of several beams leads to more power consumption. However, Lidar sensors consumes a higher amount of power which will reduce the total flight time of the UAV. Vision sensors are widely used in small UAVs which is proven by the

availability of large number of computer vision techniques for various applications [10] targeting UAV systems. The visual sensors can also provide the range, color, structure which are considered to be essential information in autonomous navigation. Besides, it is light weight and has less power consumption than e.g. a Lidar. Vision sensors can be divided into monocular camera (Figure 2.1d) and stereo camera (Figure 2.1e). The advantage of the monocular camera is that it gives impressive results with low power consumption and light weight. The drawback is that the scale of the environment cannot be obtained directly which leads to complexity in finding the three dimensional size and shape of the object in the environment. So in unknown outdoor environment it needs additional sensors to compute the 3D information. On the other hand stereo camera will directly give the depth information. The advantages include high resistance to vibration and shock due to absence of sensitive mirrors when compared to Lidar. Along with that it has large field of view which then helps in computing the 3D measurement for longer range [7].

2.1.2 Selection of the sensors

The Table 2.1 categorizes the list of sensors used in small UAVs with its advantages and disadvantages. In this thesis, as a first step the challenges in the outdoor navigation of the small UAV in unknown and dynamic environment can be handled by perceiving robust data through a suitable sensor. For that reason the small UAV should have adequate spatial information which can be obtained through a suitable sensor. This information is then later integrated in the form of a map for planning. From the Table 2.1 we can evidently see the reliability of depth information from the stereo camera. Besides that, the stereo camera provides dense information of the depth in the natural light with low noise if it is calibrated accurately. This will enable us to detect even the smaller objects in narrowed space. Here the primary information that needs to be extracted from the objects are the length, width and height. Since the stereo vision depth estimate technique is well established it can give more details about the environment. More specifically, also dynamic objects can be extracted with robustness [11].

According to [12] the sensor along with algorithm and its environment [13] decides the performance. For effective perception and planning we need dense information with low noise for fast maneuvering. Due to the above mentioned reasons along with the performance in real-time navigation, stereo camera is chosen in this thesis as the primary sensor. We designed our custom made stereo vision system in order to cope with all the objectives which is elaborately explained in Chapter 3.

Sensor	**Pros**	**Cons**
Inertial Navigation System(INS)	Estimate the 6 dof position(x, y, z) and orientation(roll, pitch, yaw). It is low cost and light weight as well as easy to mount.	Small bias leads to large error due to its integration over the time.
Global Positioning System(GPS)	Suitable for outdoor and works perfect if more than 5 satellite signals are available.	Difficult in low altitude and cluttered environment.
Sonar	Good in indoor with limited range.	Not suitable for large outdoor.
Lidar	Accurate 3d measurements of the environmental model.	Large number of beams lead to more power consumption.
Kinect	Works for slow and short range of measurements.	Not for long range measurements.
Monocular camera	Light weight and low power consumption.	Scale ambiguity, unknown and unobservable scale leads to complex error.
Stereo camera	There are no sensitive mirrors which makes it robust to vibration and shock. It has large field of view, simple to compute range.	Relies on adequate textures and lighting.

TABLE 2.1: Summary of the sensors used in the small UAVs

2.2 Feature Detection

Detection of the features in the image frame is the preliminary low level processing step in computer vision. The basic process involves the identification of edges, corners and the regions with rich information. Extracting the features which are stable and robust in real time applications is still challenging. The challenges are due to the different scale, viewpoint, illumination variation. Normally the raw image contains noises. After filtering the noises, the identification of the position of the pixels can be done through an efficient feature detection and description method. Based on the information available in the image, selected parts are extracted to check whether it is a feature or not. The requirement here is that it should be processed in real time with invariance to rotation and translational movements. In feature based techniques, textures, color, edges and corners gives primitive information. The extracted features should be resistant enough to rotation, translation and illumination invariance in the consecutive images. Tracking the identified features from one frame of the image to the other can be done through matching the corresponding features. A detailed survey about visual feature detection can be found in [14].

In Section 2.2.1 more details about the feature detection methods are explained and the definition of visual features is introduced first.

Definition 2.1. (Visual features) The primitive pixels in each frame of the images after the noises are filtered is defined as a visual feature. Generally in image the rich information can be corner, edges and region with more textures.

2.2.1 Feature detection methods

Followed by the general feature detection method in section 2.2. Different feature detection methods are briefly discussed here. Scale Invariant Feature Transform (SIFT) [15] process includes the construction of scale space. As shown in (2.1) scale space is given as,

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y) \tag{2.1}$$

where,
L(x,y,σ) is scale space function.
G(x,y,σ) is the variable scale Gaussian.$*$ is convolution operator
I is an image and x, y are the address of the pixels in the camera coordinates.
σ is the scale parameter and

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} \exp^{-(x^2+y^2)2\sigma^2}$$

It is localized by taking maxima and minima. In SIFT algorithm the orientation is calculated so that it will be rotation invariant, and based on the local image gradient the feature is identified which is invariant to illumination and distortion changes.

To increase the computational capability and faster detection Speed Up Robust Features (SURF) [16] was introduced. SURF uses the hessian matrix and the keypoint correspondence between two successive images can be calculated by computing Euclidean distance between their feature point descriptor. Later, Binary Robust Independent Elementary Features (BRIEF) [17] was developed using intensity difference between the pixels. However for real time performance the speed of BRIEF algorithm is not reliable. Followed by that Features from Accelerated Segment Test (FAST) corner method [18] was implemented by taking 16 pixels around a point and checking whether it is a corner or not. It uses the intensity difference only around the contiguous circle. It has the advantage of more reliable matching and resistant to illumination change. But according to [19], the number of points are not stable which gives less precision measurement. To increase the precision FAST detector method is fused with BRIEF descriptor with modifications. Oriented FAST and Rotated BRIEF (ORB) method was developed by using the FAST corner for scale search in image pyramid and the rotated version of BRIEF [20]. The modifications in BRIEF descriptor includes rotation invariance by computing intensity weighted centroid of the corner region. As explained in [20] the moments of a patch can be defined as,

$$m_{pq} = \Sigma x^q y^q I(x, y) \tag{2.2}$$

By using these moments the centroid can be found by,

$$C = \left(\frac{m_{10}}{m_{00}}, \frac{m_{01}}{m_{00}}\right) \tag{2.3}$$

Then orientation of the patch region θ will be written as,

$$\theta = atan2(m_{01}, m_{10}) \tag{2.4}$$

The robustness of ORB is tested in real time [19] in comparison to SURF algorithm. ORB produces less robustness with more memory consumption when compared to SURF. SURF is also more robust for scale changes.

Next to that Binary Robust Invariant Scalable Keypoints (BRISK) [21] was developed with focus on computational reliability by applying the speed in the key point scale space in the continuous domain. This method works on the binary features of the corner detection. In this method the scale space is based of n octaves with n value 4 typically. The octaves are formed by half sampling the input image. IN the sampling process Gaussian sampling σ_i is applied to avoid blur effects.

The estimation of the local gradient is done by,

$$g(P_i, P_j) = (P_j - P_i).\frac{I(P_j, \sigma_i) - I(P_i, \sigma_i)}{|P_j - P_i|^2} \tag{2.5}$$

where,
$I(P_j, \sigma_i)$ and $I(P_i, \sigma_i)$ are the points where the intensity values are smoothed using gaussian.
Similarly the Fast Retina Keypoint (FREAK) [22] method uses binary features inspired from the pattern that is used by the retina of the eye. However the FREAK method uses different kernel size while smoothing the intensities unlike to BRISK and ORB. According to [22] exponential change in size and overlapping receptive field is comparatively different from BRISK. Fro example the overlap receptive field is achieved by using intensities I_i measured at receptive fields like shown in the equation below,

$$I_A > I_B, I_B > I_C, I_A > I_C \tag{2.6}$$

where I_A, I_B, I_C are the intensities measured at receptive fields A, B and C respectively. Based on the overlap between fields new information will be added. Kaze [23] uses non linear diffusion filtering to obtain accurate localization of the features like shown in equation below,

$$\frac{\partial L}{\partial t} = div(c(x, y, t). \bigtriangledown L) \tag{2.7}$$

In equation 2.7 div, $\bigtriangledown$ and c are divergence, gradient operators and conductivity function (c) respectively where t is the scale parameter. In contrast to gaussian scale space this can be adaptive to natural boundaries of the features present in the environment. The drawback is that kaze requires high computational power. Thus in turn affects the real time performance. It becomes common to combine one feature detector method with another descriptor. For example Good Features To Track (GFTT) [24] can be combined with ORB, FREAK, BRIEF. Besides FAST detector can also be combined with FREAK and BRIEF.

2.2.2 Comparison of feature detectors through real time experiment

To choose the suitable feature detector that can give optimal performance and robustness, we compared the SIFT, SURF, BRISK, KAZE, ORB, GFTT-ORB, GFTT-FREAK, GFTT-BRIEF, FAST-FREAK and FAST-BRIEF methods. Here 3676 stereo images are taken at frame rate of 30 fps. The total duration of the sequence of images are 122 s. A sample raw stereo image that is used during the feature extraction process is shown in Figure 2.2.

(a) Left image (b) Right image

FIGURE 2.2: One stereo image out of 3676 stereo images taken during the feature detection process

Figure 2.3 shows the extracted features which are new from the previous image and its matching between the left and right images using SIFT, SURF, ORB, KAZE algorithms. As we can see the results of SURF in Figure 2.3a and ORB in Figure 2.3c, there are 35180 (SURF) and 33920 (ORB) number of features which are represented in yellow circles respectively. The feature that are present in both the left and right images are around the trees and leaves. The detection regions are in the form of large circles in Figure 2.3a, 2.3b, 2.3c whereas in Figure 2.3d the detection regions are small. ORB method has more matching correspondence whereas the SIFT has less matching.

(a) SURF (b) SIFT

(c) ORB (d) KAZE

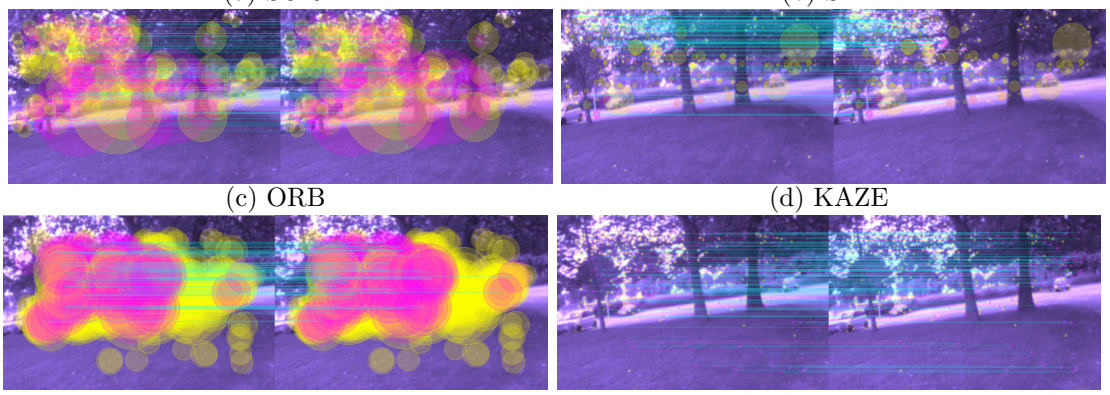


FIGURE 2.3: Features extracted by using algorithms SURF, SIFT, ORB, KAZE. The color red denotes the features found in the previous. Color yellow represents the new features and pink color represents the features matching between the left image and right image.

Figure 2.4 shows the extracted features which are new from the previous image and its matching between the left and right images using GFTT-ORB, GFTT-BRIEF, BRISK, GFTT-BRIEF algorithms. As seen in Figure 2.5b BRISK algorithm shows more new features (yellow color) and GFTT-ORB has less new features but it has more matching as shown in Figure 2.4a. From Figure 2.4b and Figure 2.4c we can see that the detection

of the features are small as points.

Figure 2.5 shows the FAST-BRIEF and BRISK feature detector output. Figure 2.5a

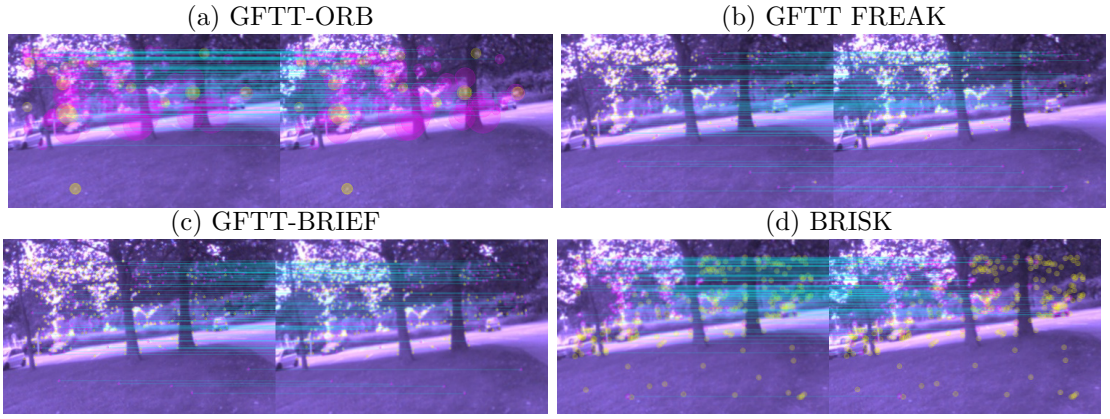


FIGURE 2.4: Features extracted by using algorithms GFTT-ORB, GFTT-BRIEF, BRISK, GFTT-BRIEF. The color red denotes the features found in the previous. Color yellow represents the new features and pink color represents the features matching between the left image and right image

has more number of features extracted in trees and ground but the same features in left and right images in less. But in Figure 2.5b the features present in both the images are more whereas the extracted new features are less.

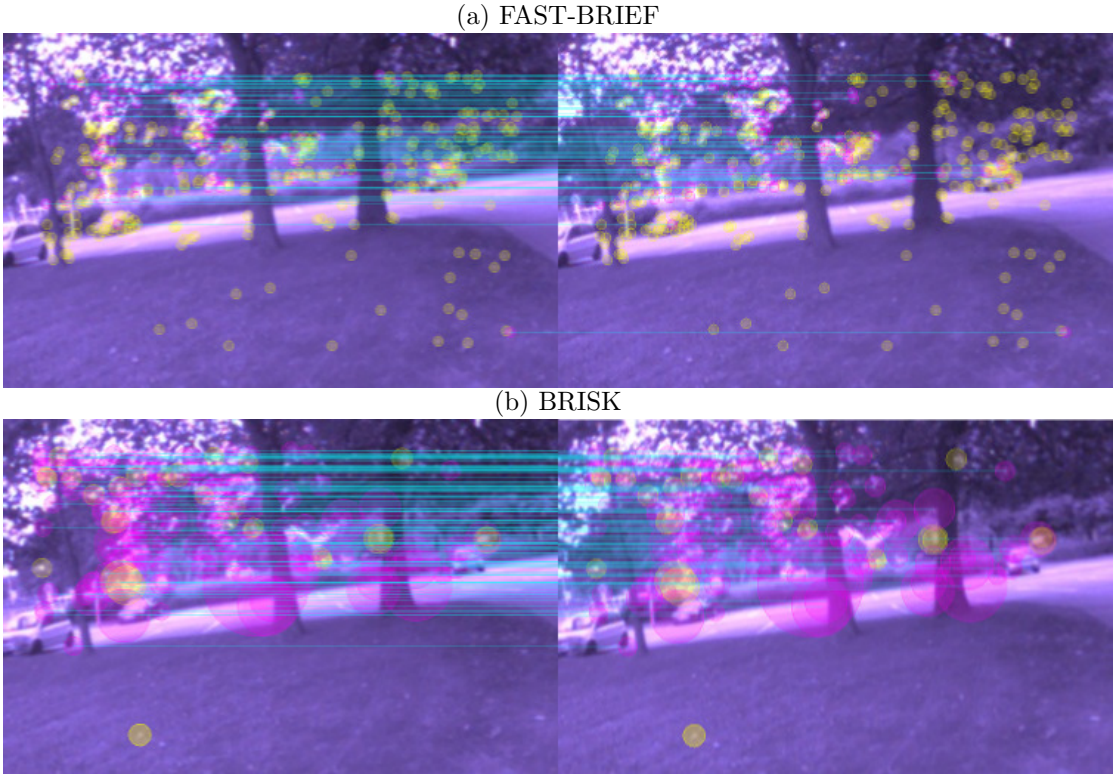


FIGURE 2.5: Features extracted by using algorithms FAST-BRIEF and BRISK. The color red denotes the features found in the previous. Color yellow represents the new features and pink color represents the features matching between the left image and right image

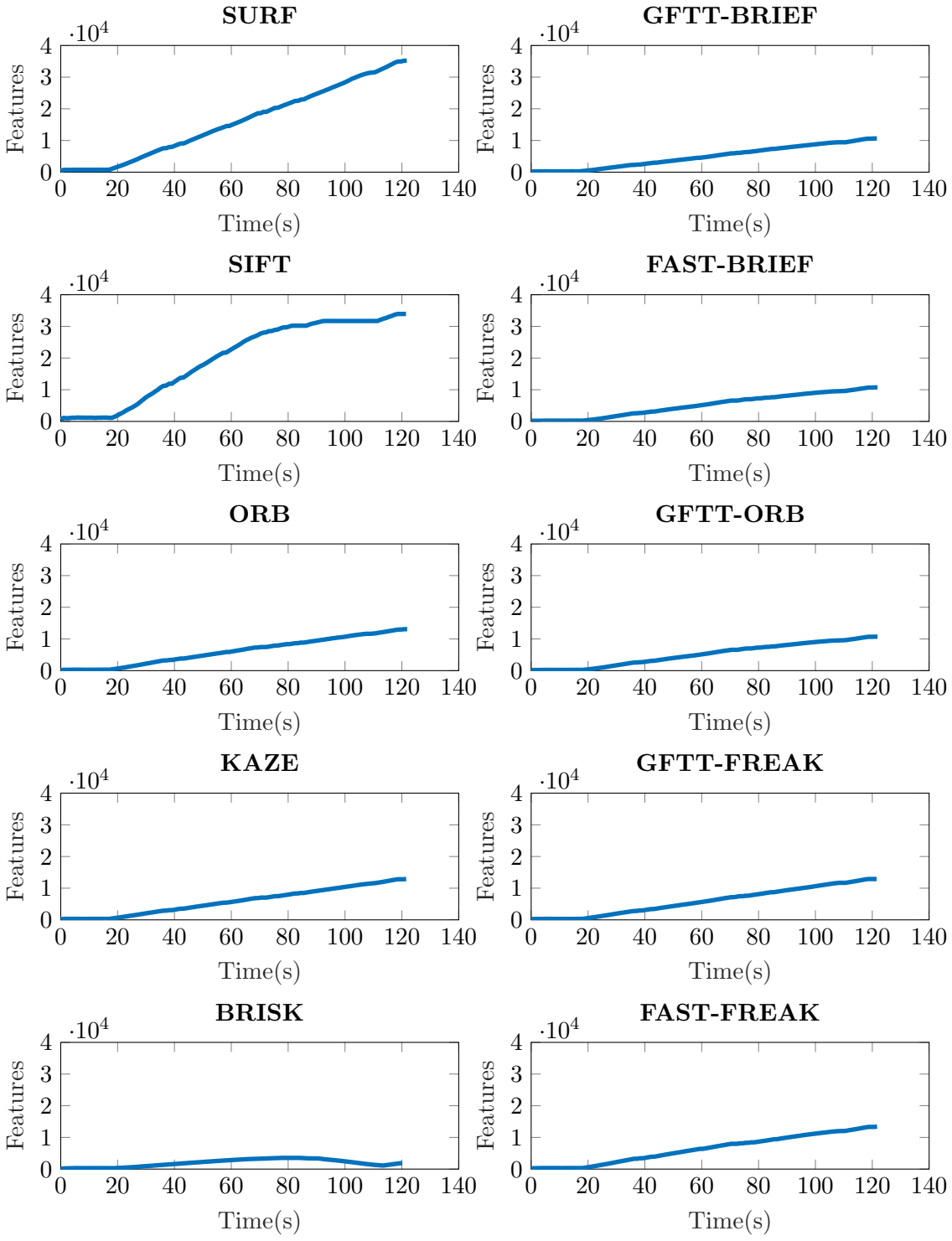


FIGURE 2.6: Total number of features vs Time (s)

Figure 2.6 shows the total number of features extracted in the time duration of 122 s. SURF algorithm extracts as much as 35180 features, which is the highest among the algorithms compared here. At the same time BRISK algorithm extracts the lowest

number of features which is equal to 3535. Table 2.2 summarizes the total number of features extracted by different algorithms.

Algorithm	**Features**
SURF	35180
SIFT	33920
ORB	13090
FAST-FREAK	13420
FAST-BRIEF	10720
GFTT-FREAK	12900
GFTT-ORB	10720
BRISK	3535
KAZE	12920
GFTT-BRIEF	10640

TABLE 2.2: Summary of the total number of features extracted.

From the Table 2.2 we can observe that the SURF algorithm detected the maximum number of features compared against other algorithms implemented here. Even though the SIFT algorithm has 33920 features which are nearly close to the SURF method, the SIFT algorithm took maximum time of 562.9 ms, contrary to SURF which took 434 ms. During the analysis, we observed that the SURF required 43.88 ms of keypoint detection time on an average. And the total time of 278.315 ms for the detection in all the images. For real time feature detection with a higher number of features, the SURF algorithm proves to be efficient as well as robust in different illumination condition. Due to these advantages and testing with different feature detectors, SURF algorithm is taken for the feature detection. We used the features as input for the Stereo Visual Odometry as explained in Section 2.3 and also in loop-closure detection, which is discussed in Section 2.4.1.

2.3 Stereo Visual Odometry (SVO)

The process of estimating the position and orientation of the vehicle using the stereo images is called stereo visual odometry. Here position and orientation combine together can be called as pose. Based on the sensor input, either monocular or a stereo image has to find the 3D motion of the camera. The advantage is that it can be useful in the case of GPS denied environment. Moreover the stereo image based visual odometry doesn't have the scale ambiguity problem because it can get scale of an object directly. However the challenges include the tracking of the motion affected by factors like sensitive image conditions or fast maneuvering of the vehicle. Due to the reliability of the features in the outdoor environments, most of the stereo visual odometry algorithms are feature-based approaches.

Figurer 2.7 shows the basic process of stereo visual odometry.

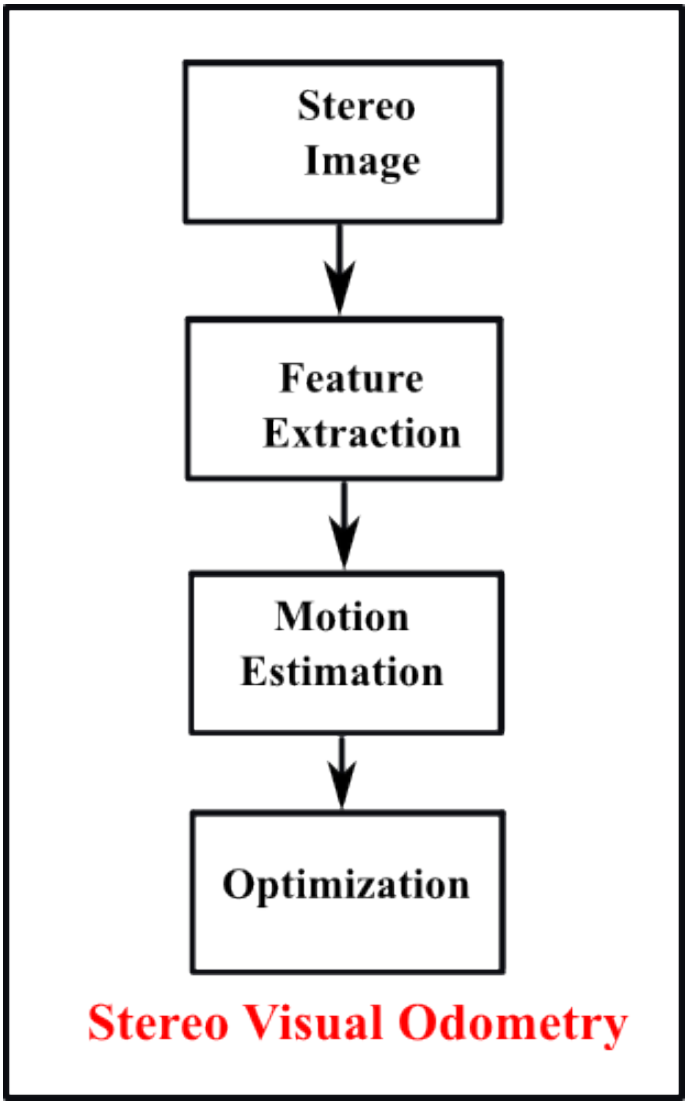


FIGURE 2.7: General stereo visual odmetry framework

The basic stereo visual odometry process can be explained by considering a vehicle with stereo camera rigidly attached to it. when vehicle started to move the stereo camera provides left and right images like in equation 2.8.

$$I_{(l,0;n)} = I_0, ..., I_n I_{(r,0;n)} = I_0, ..., I_n \tag{2.8}$$

Let's assume stereo camera coordinate system is same as vehicle coordinate system. In stereo camera system left camera is considered to be the origin. The stereo camera position at each time instants $k-1$ and k can be related using rigid body transformation $T_{k,k-1} \epsilon \mathbb{R}^{4x4}$ like shown below

$$T_{k,k-1} = \begin{bmatrix} R_{k,k-1} & t_{k,k-1} \\ 0 & 1 \end{bmatrix} \tag{2.9}$$

In equation 2.9 $R_{k,k-1} \epsilon SO(3)$ is the rotation matrix where as $t_{k,k-1} \epsilon \mathbb{R}^{3x1}$ represents translation vector. The subsequent motion of stereo camera will be in set $T_{1:n}$

$$T_{1:n} = T_{1,0}, ...T_{n,n-1} \tag{2.10}$$

The set of camera poses shown in equation 2.11 has the camera transformations at k=0 with respect to initial coordinate frame

$$C_{0:n} = C_0, ...C_n \tag{2.11}$$

The current pose of the camera C_n can be found by using all the transformations $T_{k,k-1}$ where (k = 1 ...n). Here full trajectory of the camera $C_{0:n}$ can be obtained by including relative transformation of Image I_k and I_{k-1}. The pose of the camera is incremental and it can be recovered by computing transformation between all images with respect to initial position.

According to the [25] stereo based visual Odometery (SVO) has been a popular research. In [26] iterative closest point matching based SVO is presented. More detailed experimental study of SVO with respect to the UAV can be found in [27]. This study shows that the images captured at an altitude of 200 meters are sufficient to compute the Odometry of the vehicle. Similarly [27] also presents SVO based on high altitude images with a focus on the fixed wing based UAV. In the same direction [28] explains the performance in high altitude with respect to the viewing angle of stereo camera. Image gradient based approach is used in [29] where it resists for illumination changes. Even

though these methods are real time it is not concentrating on the urban environment in low altitudes. In [30] the authors use Iterative Closest Multiple Line method adapted from Iterative Closest Point (ICP) algorithm for matching the features efficiently in the consecutive images. In Mars Exploration Rover (MER) project the visual odometry [31] is performed by using Harris corner for the feature detection and matching which is done by the pseudo normalized cross correlation. Later this is improved computationally in [32] which is developed for Mars Science Laboratory (MSL). To reduce the projection errors, [33] proposes points and line segment features which are then minimized by non-linear methods. The pose estimation through the feature matching correspondences can be categorized into 2D-2D, 3D-2D, 3D-3D methods. The reference [34] uses essential matrix for estimating the pose using 2D-2D correspondence. While 3D-2D correspondence is done in [35] where extrinsic matrix is used for the estimation. Next to that 3D-3D correspondence can be obtained from the point clouds alignment [36].

Figure 2.8 shows the stereo visual odometry of a small UAV in a static environment. The arrows shown in blue color denotes orientation of the small UAV.

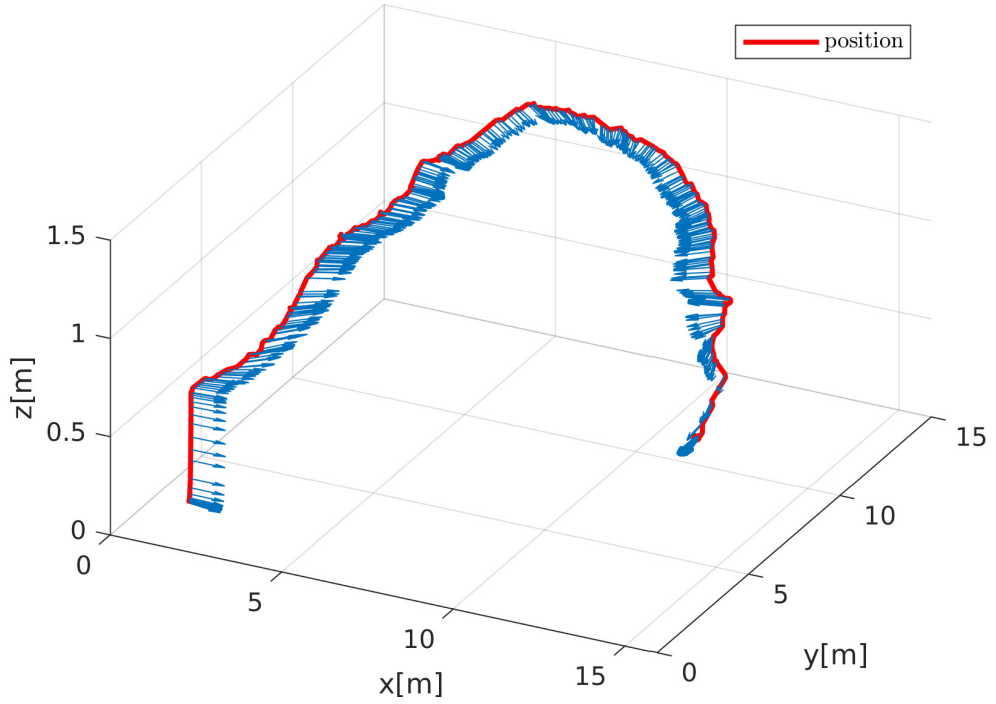


FIGURE 2.8: Stereo Visual Odometry of the small UAV traveled in the outdoor environment

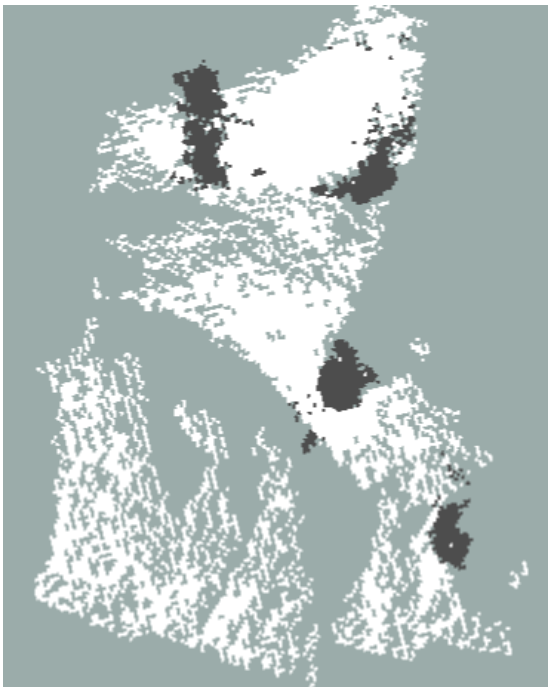
But all these methods don't addresses the localization problem in dynamic environments. In dynamic environments the localization can be affected by the dynamic features. As long as the moving objects are in the field of view, features can be detected and tracked

in the consecutive frames. For example in this work[11] used optical flow to extract the moving objects. After the objects moved away from the view, it is not possible to track. Later in [37] the authors use Iterated Sigma Point Kalman Filter with RANSAC for feature classification from the moving objects. But this technique is not often considered for small UAV applications, because matching the features in plane surface as an assumption does not work for the flying vehicle. To overcome the challenges we proposed the solution by setting the threshold for the features classification. For minimizing the re-projection error we used the gauss newton optimization along with RANSAC.

2.4 Simultaneous Localization and Mapping (SLAM) methods

Localization of the small UAV relative to the map of the environment can be achieved by using the well known technique called Simultaneous Localization and Mapping (SLAM). SLAM has been extensively used in robotics research community where the history and its development in the early stage can be found in [38]. Furthermore the implementation challenges like data association, computation and accuracy in SLAM for the last 20 years are explained elaborately in [39]. Recent developments and advancements include the real time problems in terms of robustness, scalability, map models, sensors and new tools are explained in the survey paper [40]. The fundamental idea of SLAM involves the construction of map using the data perceived by the on-board sensor while localizing the UAV in that map at the same time. In the early stage it starts with 2d mapping and localization as shown in Figure 2.9. The 2D mapping techniques made considerable progress and have available industrial solutions like [41]. They used vision sensor [42] for slow moving robots [43], which are commercial products now. In the case of 3D mapping and localization for small UAV there are still challenges which are yet to be solved. Particularly for unknown outdoor environments there are fundamental challenges that need to be addressed. So it requires fundamental research with the combination of sensors with fast sensing of the environment, fast maneuvering and computational power. These capabilities in exchange reduce the failure rate, increase the robustness and performance which is application oriented. One of the key difficulties with these 3D sensing technologies is the limitations and reliability of the computational power. But recent advancements in the computing technology gives the scope for the three dimensional mapping and localization to be easily applicable to common robotic systems. It is also more realistic and practical approach for many UAV applications.

(a) Image showing the scenario



(b) Two dimensional map

FIGURE 2.9: Image and its two dimensional map where the black , white, grey color represents the obstacle space, free space and unknown respectively.

In [44] the standard formulation of SLAM as maximum posteriori estimation problem was discussed. Later it was improved with an efficient optimization process. Let us take an example to find the trajectory of the small UAV and the pose of the points in the environment which is represented as unknown variable κ and the set of measurements Z like in equation

$$Z = \{z_n : n = 1, ...m\}. \tag{2.12}$$

Here measurements can be expressed as a function of κ like

$$z_n = h_n(\kappa_n) + \epsilon_n. \tag{2.13}$$

where h_n is a known function and ϵ_n is random measurment noise.
Similar to SLAM method, the Parallel Tracing and Mapping (PTAM) presented in [45] is often used with the monocular camera rather than the stereo camera [46].

Some of the key SLAM methods include Extended Kalman filter based SLAM (EKF-SLAM), Particle filter and Graphical model based approaches. In [47] it is shown that the classical EKF-SLAM is inconsistent, however further improvement was done in [48]. But still the EKF-SLAM lacks the accuracy for the large scale mapping [49].

The particle filter method [50] are well implemented in landmark-based and grid representations [51]. According to [52] the particle filter has some basic limitations due to its bounded number of particles required to estimate the pose and trajectory. This problem will deteriorate depending on the noise of the sensor and its parameters. In dynamic environments, the SLAM techniques which depends on the iterative closest point algorithm does not update the moving objects. This in turn the makes the map inconsistent and it will not help in planning the safe navigation. So here the algorithm requires a preprocessing step from the point cloud to map. The aforementioned shortcomings are overcome by graph based SLAM techniques. For example the self correction in online mapping are done in [53] and [54] [55]

As mentioned in Section 2.1, for the perception system, stereo camera is the primary sensor. The contribution of computer vision algorithms in the field of UAV is well known and the recent survey of the vision based navigation techniques are found in [10]. In this thesis the focus is on the vision based SLAM algorithms to attain accurate localization and mapping for collision avoidance. An extensive survey on visual SLAM has been noted in [56]. It has also been addressed in the literature that the accuracy and robustness depends on the data association (short term: feature tracking, long term: loop closure) from the vision sensor. The frame update rate should be able to track the features in the consecutive frames. In this setup feature based SLAM gives high precision. In the context of SLAM research Visual Inertial odometry algorithms are presented in [57] [58]. In these algorithms the loop closure detection is not implemented which means the exploration of the environment is infinite. So by using loop closure it is possible to get the more accurate topology and also be capable of finding the shortcuts between locations. Estimating the precise topology is the key factor to achieve robustness.

2.4.1 Loop Closure Detection

Loop closure detection combined with mapping and localization plays a signification role in attaining the consistency, accuracy of the information and computational power. The basic idea of loop closure detection is to identify the data that is already in previous measurements taken from the same environment. This is challenging due to inherent noises in the sensor and ambiguities in the environment. Image retrieval from the feature detector and descriptors act as the main source for identifying the same place in the environment. Recognizing a place that is already captured using the visual data online is a complex task due to the variations happening in the real world environment from time to time. According to the literature, this method can be called “visual place

recognition" or "loop closure detection". Further the key questions in the topic of Loop Closure Detection are as follows:

- How to recognize a place with respect to the visual data generated by the sensor?
- If identified and recognized, whether it is the same one or the wrong identification due to perceptual aliasing?
- How this will influence the collision avoidance in real time?

The computer vision community have tried to enhance the computational power and precision in the estimation for large scale outdoor environment by using the loop closure detection methods [59] [60]. Further detailed state of the art can be found in [61]. From the literature common approaches for loop closure detection includes voting [62] and bag of words [63] [64] methods. In voting methods maximum likelihood estimation is used for matching the current image features with previous one in the database. According to [62] the maximum likelihood estimator depends on the offline construction of the dictionary i.e a database. In contrast bag-of-words uses classified image. The bag of words technique further includes online and offline methods. We focus on the online methods which can incrementally update the database. This can be categorized into two. One of them is to retrieve the interest points in the image as explained in Section 2.2 [15][16] and other method uses the whole image instead of using the detection process inside the image. The advantage of online loop closure detection is that it can be used without prior information.

2.4.2 Three Dimensional Map: Octomap

Modeling the environment in the form of 3D map will be a useful information for navigation of the small UAV. The main reason to get a 3D map is to retrieve the position of the obstacles and the information about occupied and free space available. These two informations are mandatory to make a decision for planning a collision free path. To model the environment as a 3D map, methods like voxels where the grid is arranged as cubes [65] are commonly used and developed even for small environments with millimeter precision [66]. The drawback in grid volume representation methods are that it needs large memory to store and compute. Other algorithms include the stored point cloud [67] methods. The problem with point cloud method is that it does not differentiate between free space, occupied space and unknown space. In [68], [69] the authors used octrees for mapping and navigation. Octrees stores information in the form of cubic volumes and it has hierarchical structure which can be subdivided into eight sub-volumes. The minimum size gives the resolution of the octree. Based on this octree, popular approach

called octomap [70] is developed. The octomap uses the probabilistic sensor fusion to integrate the sensor readings. As provided in [70] the probability $P(n|z_{(z:t)})$ of node n occupied can be estimated provided sensors measurements $z_{1:t}$ like in equation below,

$$P(n|z_{(z:t)}) = \left[\frac{1 - P(n|z_t)}{P(n|z_t)} \frac{1 - P(n|z_{1:t-1})}{P(n|z_{1:t-1})} \frac{P(n)}{1 - P(n)}\right]^{-1}. \tag{2.14}$$

The above equation is rewritten as in [70],

$$L(n|z_{1:t}) = L(n|z_{1:t-1}) + L(n|z_t). \tag{2.15}$$

with,

$$L(n) = log\frac{P(n)}{1 - P(n)}$$

In addition to that, octomap uses multiresolution and also a clamping policy for map compression. One of the advantage is that the octomap can update the occupied space through raycasting [70]. This technique helps in handling of dynamic objects with implicit cell update. The Figure 2.10 shows the octomap and its respective image. From the Figure 2.10b we can see the unstructured trees which are located on the ground. The same can be observed in the Figure 2.10a. For example the shape of the trees originating from the ground and the distance between each trees. In Figure 2.10b the octomap colors red, green and blue represents the x, y and z axis respectively.

(a) Image

(b) Octomap

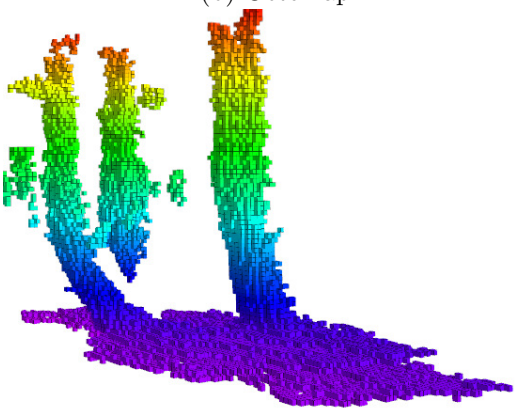


FIGURE 2.10: Image and its respective Octomap where the green , red, blue color represents the x, y and z axis respectively.

From the literature it is evident that graph based SLAM with loop closure detection and octomap will perform real time under the memory constrains.

2.5 Path Planning

Path planning for UAV systems [71] [72] is a wide research area with many approaches and algorithms. The path planning strategies for UAV system focuses on the performance optimization, collision avoidance, real- time planning, risk minimization. Most of the current research focuses on two dimensional path planning [73] as shown in Figure 2.11 [74] ,[75] but this technique works with static obstacles with terrain navigation or in the environment which is already known. Classic methods include probabilistic roadmaps and potential field methods.

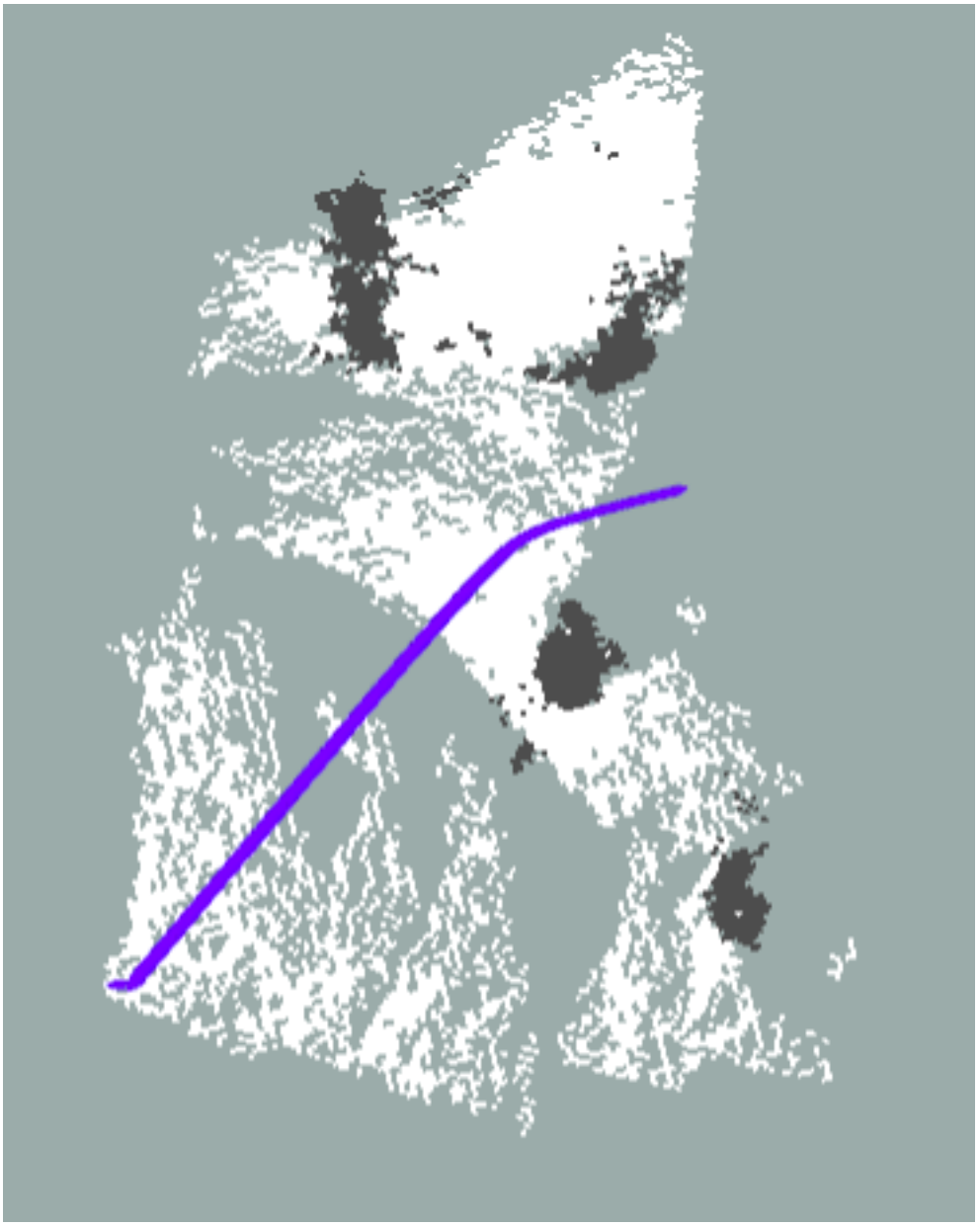


FIGURE 2.11: Two dimensional path planning in the two dimensional map where black and white color represents the occupied and free space respectively.

There are three cases when it comes to path planning,

- **Case 1**: Two dimensional planning in two dimensional map.
- **Case 2**: Two dimensional planning in three dimensional map.
- **Case 3**: Three dimensional planning in three dimensional map.

Figure 2.11 shows the 2D path planning in 2D map. Here the white and black color spaces are considered to be free and occupied space. So planner will search the collision free path in x and y direction only. In extension to that the same 2D path planning done in 3D octomap is shown in Figure 2.12. In this example we have two trees in the form of octomap

The planner doesn't consider height information for collision free path. So it has the risk of colliding the obstacles present with different height. This kind of 2D planning will be implemented in real time application upto certain limit.

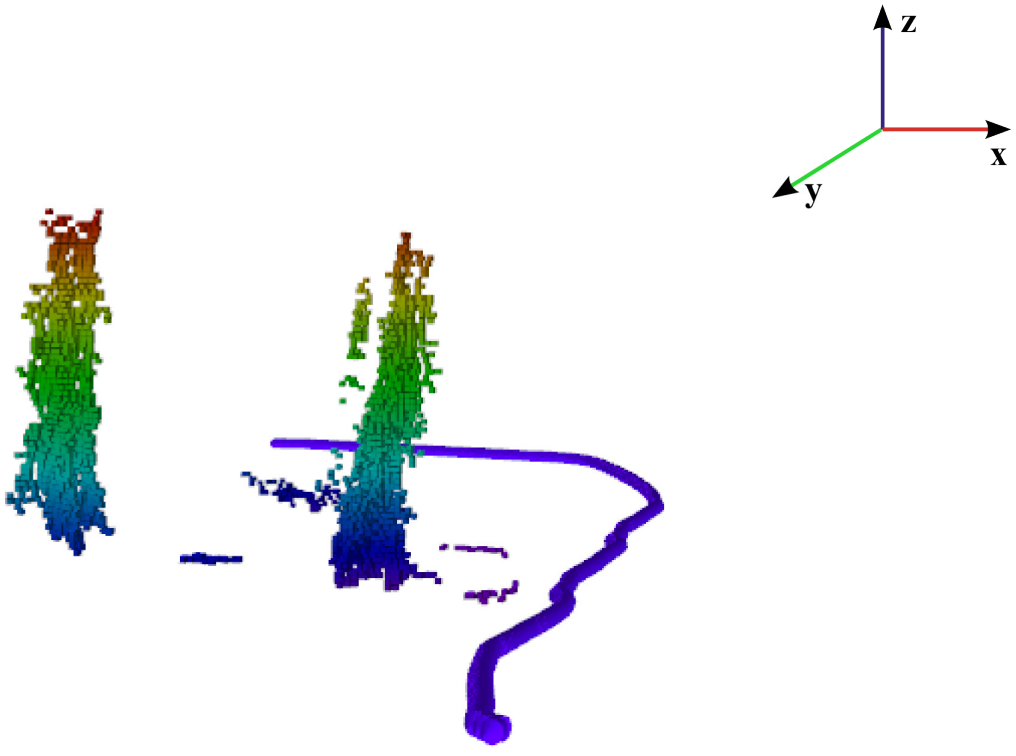


FIGURE 2.12: Two dimensional path planned in three dimensional map.

Typically the real world includes cluttered unstructured environment with static as well as dynamic objects which introduces lot of uncertainty. Due to the above factors, optimal three dimensional planning becomes mandatory for the UAV systems in outdoor

unknown environment. In contrast to the 2D planning, the complexity in 3D planning is more due to the kinematic constraints. 3D path planning algorithms specifically for small UAV systems have been investigated in [76]. Recent literature review [77] classifies the 3D path planning algorithms into five categories such as sampling based, Node based optimal algorithms, Mathematical model based, Bio-inspired, Multi fusion algorithms.

Sampling based algorithm works with known environment which will be sampled as cells or nodes. Based on the search technique in the nodes or cells it will find the feasible path. Probabilistic road Maps [78], Rapidly exploring random trees [79] and Potential field algorithms [80] are the examples of the sampling based algorithms.

Node based algorithms are based on the decomposed graph [81] and the search will be based on set of nodes or cells to find the optimal path. Algorithms include Dijkstra[82], A^*, Lifelong planning A^*[83] and D^* [84] [85] comes under node based algorithms. Mathematical model based algorithms work with kinematic and dynamic constraints to achieve the optimal solution. Mixed-Integer Linear Programming [86], Binary Linear programming [87], differential flatness [88] are mathematical model based algorithms. Bio-inspired algorithms use the stochastic approaches to search the near optimal path. Genetic algorithms [89], Memetic algorithm [90], Ant Colony Optimization[91] are some of the bio-inspired algorithms.

The Node based optimal algorithm which also includes graph based search method is a well known technique for finding the shortest path between nodes. While Optimizing the path as shorts as possible, it often produces a suboptimal solution, but it can still be deployed in complex mission due to its simplicity. It uses nodes, arc and its weight information to calculate cost of path. For 2D environments it has less complexity in finding the cost of path when compare to 3D environments. However when it comes to re-planning in 3D environments it requires strong computational power.

Potential field algorithm works on the attractive and repulsive forces on the goal point and the obstacles respectively. But the tendency of generating the local minima is difficult in complex scenarios. Evolutionary algorithms use optimal solutions but the computational cost is high. On the contrary graph search algorithms are proven to be robust and efficient especially when it comes to long term planning in large complex outdoor environment with dynamic objects. Graph search algorithms along with efficient optimization can be also used for real-time path planning in 3D environments.

To tailor path planning for the small UAV in urban environment we require fast re-planning. For example if there is a new obstacle detected in previous valid path, it needs to re-plan repeatedly from the current position to target position. This re-planning module opens up different applications.

To focus on dynamic replanning we modified the classic D* Lite algorithm to do a 3D search and planning. Since D* Lite is an incremental heuristic search method based on LPA* used for 2D planning. LPA* uses valid path information that is calculated already using the previous information to re-plan new path. It repeatedly replan while exploring the search.

Generating 3D map and searching the collision free 3D path to plan on it would be the realistic approach. So in this thesis graph search is applied in the 3D grid and based on that 3D collision-free path planning is done online.

Chapter 3

Design and development of custom stereo camera

This chapter discusses the design and development aspects of the custom made stereo camera used in this thesis. Besides that, it describes the challenges specifically related to outdoor environments. This includes the sensitivity, illumination variation and computational power. First, we provide the overview of the design and development of the stereo camera that has the potential to be implemented in a small UAV for real time navigation. Second the fundamental concepts of the stereo geometry, stereo matching algorithms and its initial experimental results are presented here.

3.1 Motivation and Challenges

Stereo vision gives depth information by using the 2D image scene taken from different view points of two camera. Since the two cameras are displaced with each other, a point in the real world appears in each 2D image with displacement. By finding the corresponding points in the two images with known configuration of two cameras, depth of a point can be estimated. Moreover, the base length of the two cameras determines the maximum range at which the stereo camera can compute the depth information. The range decides the accuracy, robustness of the data. Here we focus on the outdoor environment sensing from the small UAV, we need base lengths that can be adjustable according to the scenario. Because the outdoor environment has different kinds of scenarios like narrowed buildings and streets or vast open space with few objects or more cluttered with objects that are placed near to each other. To handle those scenarios we need a stereo camera setup that can have adjustable base length with easy adaptability. The available COTS stereo camera products have fixed baseline, we cannot change their

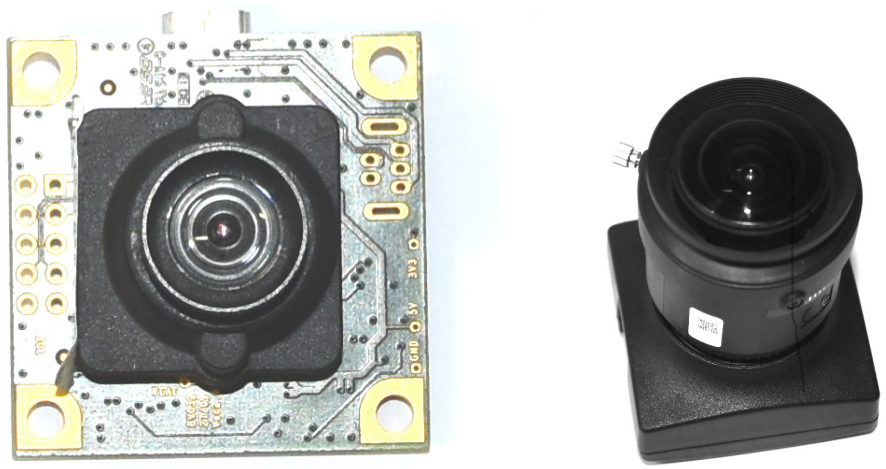
distance between each of the cameras. So we designed custom made stereo camera with varying base length and adaptable sensitivity. Eventhough the camera setup is custom made, still there are challenges like inherent ambiguities associated with stereo camera. This will affect the accuracy and fast computation of the depth information. This can be tackled by the stereo matching algorithms but the complexity in the processing time does not guarantee the real time implementation. For example the noise in the image causes blur which impacts the robustness. So the parameters of stereo camera had chosen appropriately in this thesis.

3.2 Stereo camera setup

Low cost CMOS cameras are widely used in research and the technology is well established. It can provide depth information with color data in contrast to the time-of-flight, structure light cameras [92]. We used two ueye CMOS cameras to develop the stereo camera setup. First we designed and developed a stereo setup with board level camera model UI-1221-LE-C-HQ which is shown in Figure 3.1a. Then later we designed another setup using the housed camera model UI-1220-LE-C-HQ with Tamron lens which is shown in Figure 3.1b.

3.2.1 Camera specifications

In computer vision, the first and foremost work is to decide the type of images required to achieve the mission. Then it is followed by the design and model that can be helpful to attain the objective of the work. In our case we need color images which has 24 bits per pixel where as the grayscale images provide 8 bits per pixel. So color images have more information to extract the depth data. Moreover, we need global shutter camera because the camera will be always in motion and it has to handle dynamic objects. Figure 3.1a and 3.1b show the two camera models which provides color images and global shutter. Initially, the stereo camera setup is developed using two identical single board level cameras as shown in Figure 3.1a. Later we developed the second one using the two identically housed camera model like in Figure 3.1b. The weights of the board level and the housing camera without lens are 16 grams and 44 grams respectively. The specifications of both the cameras can be found in Table 3.1. Both of them have the same specification except the outer housing of the camera and the lens model. The advantage of the board level camera is that it is light weight and which will be useful while mounting it on the small UAV. Though the housed camera we have is heavier than the board level camera, it can provide longer range and its lens has easy adaptability option for focusing and exposure settings.



(a) Board level ueye camera- UI-1221LE-C-HQ (b) Housed ueye camera - UI-1220LE-C-HQ

FIGURE 3.1: Camera models.

Parameters	**Values**
Image width	640
Image height	480
Frame rate	30 fps
Red gain	0
Blue gain	0
Green gain	0
Exposure	1ms
Flash delay	1000 μ s
Flash duration	30000 μ s
Pixel clock	30 MHz
Color mode	rgb8
Shutter	global shutter

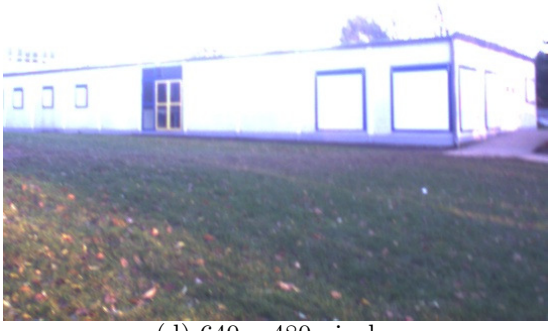
TABLE 3.1: Specification of the ueye camera model UI1221-LE-C-HQ and UI1220-LE-C-HQ

3.2.2 Images and discussion

We have checked different resolutions starting from 376 × 240, 540 × 480, 640 × 480 pixels along with frame rates ranging from 30 fps, 20 fps, 15 fps. We focused on the outdoor environments, the exposure always need to be adapted according to the lighting condition. For example the images shown in Figure 3.2a and 3.2b have the resolution of 376 × 240 and 540 × 480 respectively. These two images taken using the board level camera 3.1a. Later the images shown in Figure 3.2c and 3.2d are taken using the housed camera 3.1b. We can notice the difference in terms of contrast and brightness. The images produced by both the camera models have advantages and disadvantages. For instance low image resolution can increase the speed of the computation but it cannot give more information from longer range. In contrast, higher the image resolution it will take more computational power, but it will take more measurements to perform computation. So depending on the environment and lighting condition, we choose the parameters like image resolution, frame rate and exposure rate.

(a) 376 x 240 pixels

(b) 540 x 480 pixels



(c) 640 x 480 pixels

(d) 640 x 480 pixels

FIGURE 3.2: Images with different resolution taken in outdoor environment.

3.2.3 Sensitivity

The camera can be sensitive to different operating conditions such as lighting, vibrations, reliability of the power. During the initial testing we used auto exposure to adapt to the light available. But it is not constant enough to produce the clear images. So we adjusted the exposure time according to the environment condition. Figure 3.3a shows more shadows in the cluttered outdoor environment where the camera was set in the auto exposure mode. In contrast to that we can observe more light in Figure 3.3b and it is difficult to find the objects present in that area. The same kind of issue is observed while the environment is filled with more snow on the ground as shown in Figure 3.3c. Another scenario where shadows and sunlight are mixed and it is not adapted by the auto exposure like shown in Figure 3.3d. So we used the constant exposure time varying from 1 ms to 6 ms. To avoid blurring in both the cameras, the lens can be adjusted. Furthermore in the housed camera seen in Figure 3.1b the lens has more options to adjust and lock it manually.

(a) 376 x 240 pixels

(b) 540 x 480 pixels

(c) 640 x 480 pixels

(d) 640 x 480 pixels

FIGURE 3.3: Images with sensitivity taken in different outdoor environment.

3.3 Mechanical assembly of the stereo camera

The two identical board level cameras are mechanically assembled as a stereo camera like shown in Figure 3.4. We used carbon rod to fix it with maximum length of 299.5 mm with outside and inside width of 10.12 mm and 8 mm respectively. The cameras are initially fixed with a carbon base plate of width 1.5 mm. The dimensions of the plates can be found in Appendix A. This stereo setup was initially tested and used for the experiments. Figure 3.5 shows the stereo setup mounted on the small UAV DJI- Matrice 100.

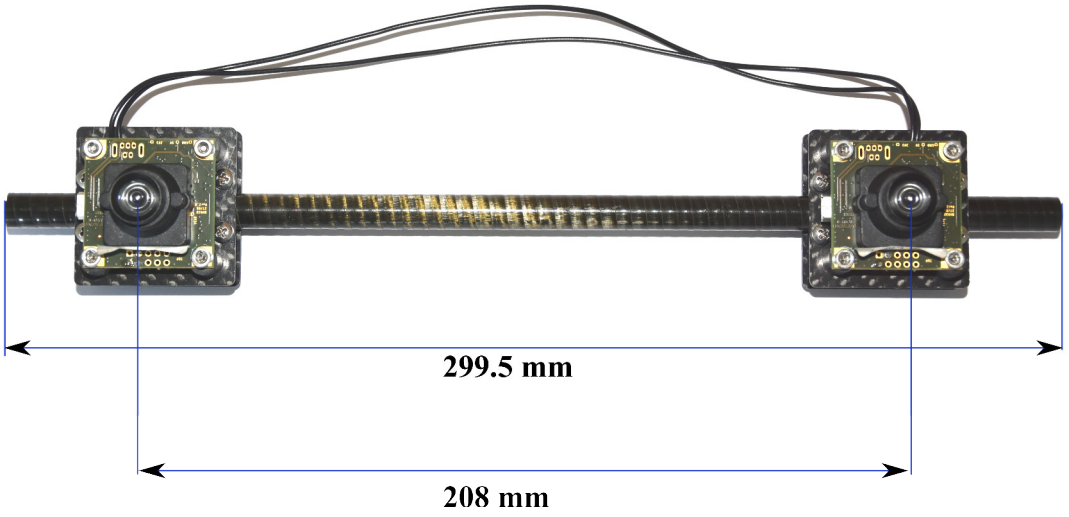


FIGURE 3.4: Board level stereo camera setup using UI- 1221-LE-C-HQ camera model.

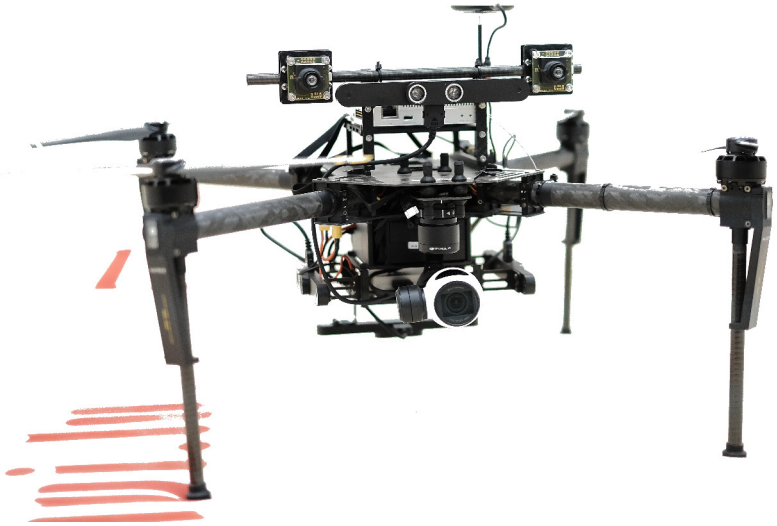


FIGURE 3.5: Board level stereo camera setup using UI-1221-LE-C-HQ fixed on the DJI-Matrice 100 UAV

Later we designed another stereo setup with same specification but with different lens (CS-type) like shown in Figure 3.6. This gives larger field of view which is beneficial in the large scale outdoor scenario. The focus and exposure can be adjusted manually in the Tamron lens. In the outdoor environment the daylight often changes, in that case this lens set up will be beneficial. This setup is assembled with the hollow carbon rod of maximum length of 300 mm. Each of the camera is clamped using a dedicated plastic holder.

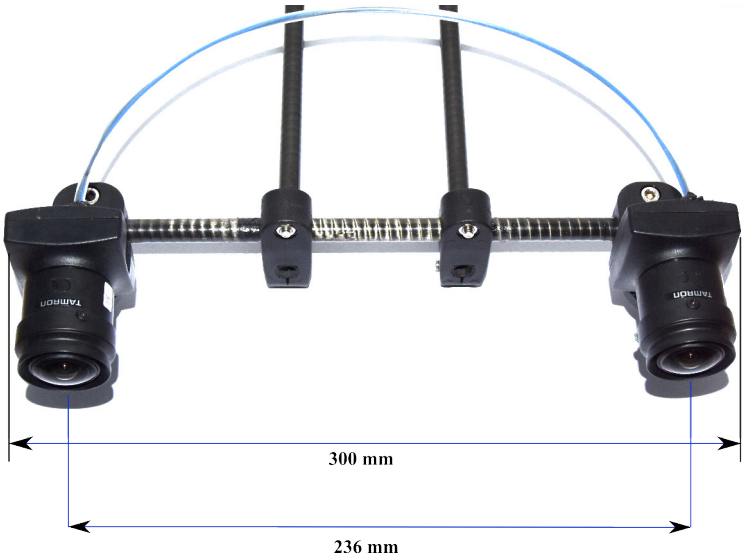


FIGURE 3.6: House model stereo camera setup with Tamron lens using UI- 1220-LE-C-HQ

FIGURE 3.7: Housed stereo camera using UI- 1220-LE-C-HQ with Tamron lens setup fixed on the DJI-Matrice 100 UAV

Table 3.2 shows the weight of both the stereo camera setup with all the materials included.

Camera	**Weight(g)**
UI-1221LE-C-HQ - single camera	14.7
UI-1221LE-C-HQ - stereo camera setup	91.7
UI-1220LE-C-HQ - single camera	77.2
UI-1220LE-C-HQ - stereo camera setup	304.4

TABLE 3.2: Weight of the single camera as well as stereo camera setup

3.4 Synchronization

Since both the stereo camera setup is made up of two independent cameras. The time stamps of the two camera are often not identical. This affects the complete process of calculating the 3D points. To avoid the timestamps varying from each other, the clocks of the two identical cameras should be synchronized. Otherwise it will lead to erroneous estimation while extracting the depth information. So we synchronized the two cameras using master slave configuration. We used cable to form the connection between the two cameras. The master slave configuration is formed by connecting the trigger input and flash output of the master camera to the trigger input of the slave camera. The configuration of the detailed circuit diagram can be found in Appendix A. This synchronization setup is tested upto 25 fps for identical time stamps.

3.4.1 Tuning

Since the stereo camera setup is connected with the on-board computer during the experiments it requires tuning of the usb bandwidth transfer rate. We performed software tuning to achieve identical time stamps for both the cameras. We modified the default parameters of bulk transfer size, Nice value, USBFS memory which are shown in Table 3.3. These parameters are from ueye-camera drivers.

Parameters	**Values**
Bulk transfer size	512 kb
Usbfs memory	1024 mb
Nice value	43

TABLE 3.3: Modified parameters to handle usb transfer failures

Using bulk transfer size the behavior of the usb- subsytem can be adjusted whereas the Usbfs sets the buffer memory. Furthermore Nice value is used in master slave configuration for bandwidth. During synchronization the master slave camera configuration has a polling mechanism that the master camera needs to be triggered before the slave camera. During the implementation we observed that subscribing both the cameras at the same time creates failure rate of half of the frame rate.

To avoid this issue we initiate the master camera first upto 10 fps and then the slave camera will be started to capture the images. In this way we received identical stereo images constantly during experiments.

3.5 Stereo Camera Fundamentals

This section explains the theoretical aspects of the stereo image geometry and its parameters. First we discuss the concepts of the pinhole camera model, epipolar geometry and the perspective projection which are required to understand the stereo camera model and its geometric relationship between the sensor and the captured images.

In general pinhole camera is a simple camera which use a mathematical model to relate the 3D world coordinate system and its 2D projection in image plane. Figure 3.8 shows the central projective imaging model. In the origin of camera frame the rays converge to form a non-inverted image which is then projected to image plane at the location of z=f. Here X_w, Y_w, Z_w represents 3D points in world coordinate system whereas (x_c, y_c, z_c) represents camera coordinate system in the image plane. The geometric relation between 3D coordinates to the 2D using the pinhole camera model is called perspective projection. The distance from the image plane to the optical center is referred as focal length f. From Figure 3.8 the point in world coordinates $P(X_W, Y_W, Z_W)$ projected on to a image

plane p =(x,y) can be obtained using similar triangles like in equation below,

$$x = f\frac{X_W}{Z_W}, y = f\frac{Y_W}{Z_W} \tag{3.1}$$

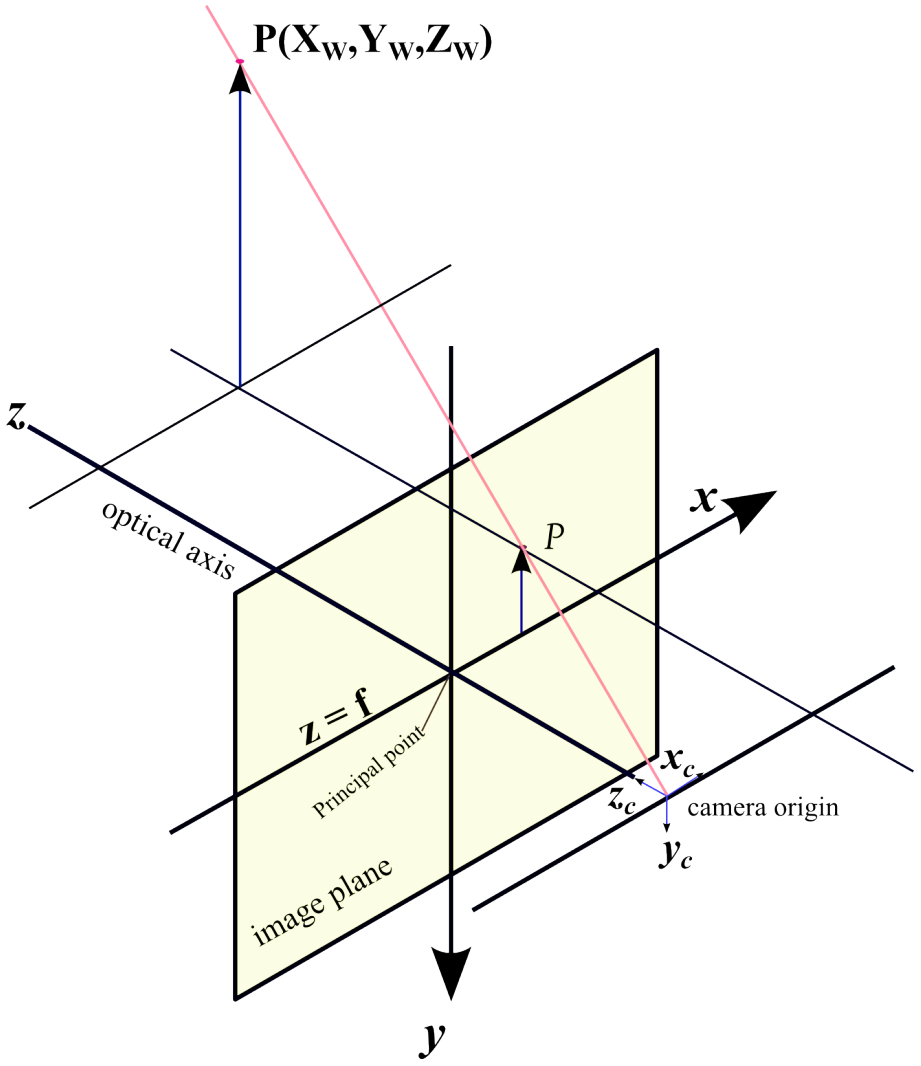


FIGURE 3.8: Principle of pinhole camera model

For stereo geometry we need to used two pinhole cameras which basically acquire 2D images of an object from different positions inorder to construct 3D images. Figure 3.9 shows the two pinhole cameras to form the stereo geometry with camera or optical centers O_L and O_R of the left and the right camera. P is a point in the 3D space where the p_l and p_r denotes the projection of the point $P(x, y, z)$ in the left and the right images respectively. Here e_l and e_r represents the epipoles which is defined by the intersection of the point with the line across the optical centers. The epipolar plane is formed by the points P and the optical centers O_l and O_r. The line connecting $P - O_l$

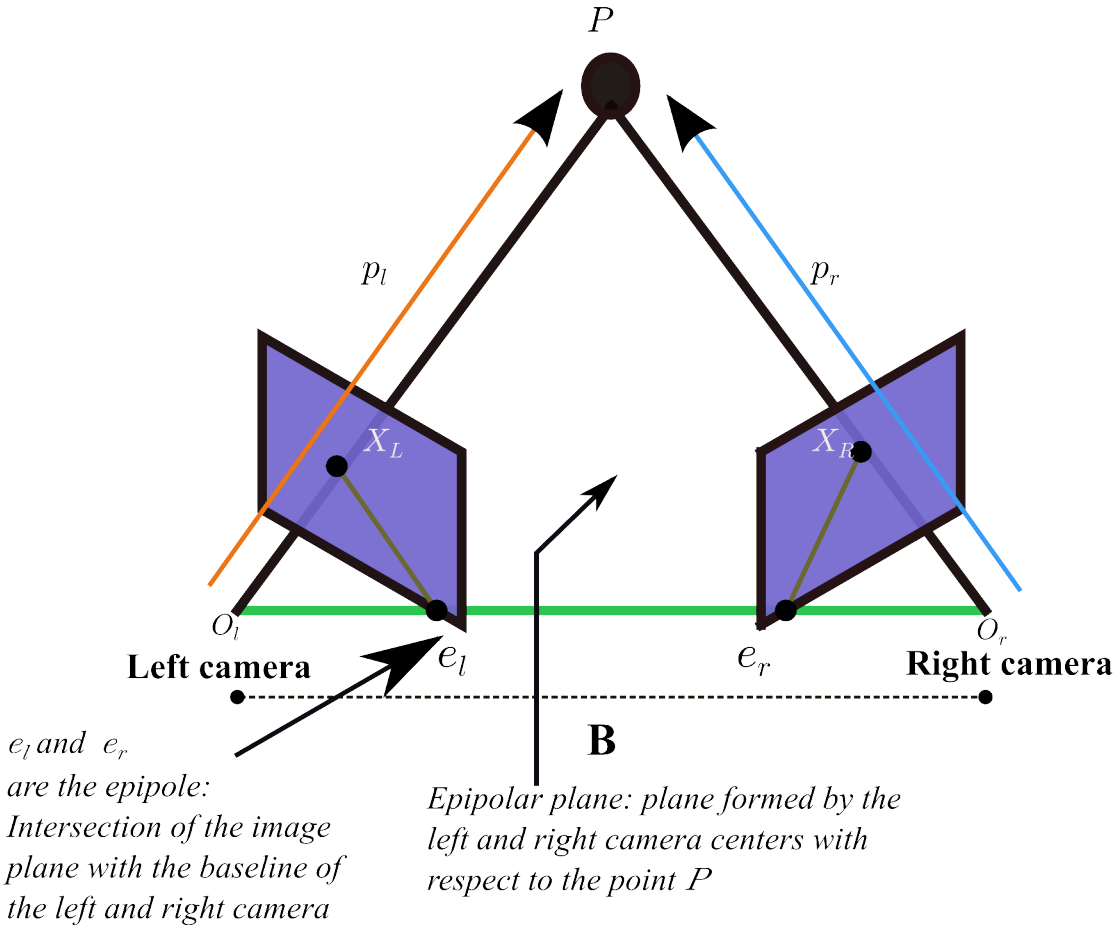


FIGURE 3.9: Epipolar geometry

in the left camera is denoted as p_l image. The point in the left image which also needs to be mapped in the epipolar line of the right image is called epipolar constraint.

With the pixel coordinates of left image $p_l(x_l, y_l)$ and the right image pixel coordinates $p_r(x_r, y_r)$ the 3D point $P(x, y, z)$ can be calculated with the baseline B formed by the distance between the optical center O_l and O_r. Z is the depth of the object which is the distance between P and the baseline B.

Essential matrix

The vectors p_l and p_r lie in the epipolar plane. Co-planarity constraint between the vectors $(P_l - B)$, B, P_l is given by,

$$(P_l - B)^T B P_l = 0$$

$$P_r^T R B P_l = 0 \tag{3.2}$$

$$P_r^T RSP_l = 0 \tag{3.3}$$

where, R is the rotation matrix S is the translational matrix

$$S = \begin{pmatrix} 0 & -T_z & T_y \\ T_z & 0 & -T_x \\ -T_y & T_x & 0 \end{pmatrix}$$

Here the epipolar geometric constraints are expressed interms of normalized image co-ordinates,

$$P_r^T EP_l = 0 \tag{3.4}$$

$E = RS$ is the essential matrix. The conjugate points P_r^T and P_l are in homogeneous normalized image coordinates. This essential matrix has 5 degrees of freedom which has 3 rotation and 2 translation parameters. The rank of essential matrix is of two and it contains two non-zero singular values.

Fundamental matrix

Epipolar line formed in right image can be defined using the fundamental matrix and a point in the left image. The points in the left image and their corresponding points in the right image can be related using the fundamental matrix. It is formed by applying the camera model and the essential matrix as shown below,

$$X_R^T F X_l = 0 \tag{3.5}$$

where X_R^T and X_l are image points in homogeneous form and F is 3x3 matrix which is called fundamental matrix.

$$X_R^T \begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix} X_l = 0 \tag{3.6}$$

The relationship between two corresponding views can be obtained by fundamental matrix. Like in equation 3.6 fundamental matrix is a 3x3 matrix i.e 9 components with rank 2 and it has seven degrees of freedom.

Homography matrix

As shown in Figure 3.9 the set of points in the world captured by the stereo camera are then projected onto a common plane. Since it is viewed by two cameras, the relationship between those two points can be defined as 3×3 matrix as shown below,

$$p_l = H p_r \tag{3.7}$$

where, H is the homography matrix with rank 3. Using the homography matrices the stereo images are rectified. Stereo image rectification is an important step for real time processing. The underlying process is to generate the matrix conversion based on the calibration parameters i.e intrinsic and extrinsic parameters applied to the images along with homograph matrix H. This will align the stereo pair of images in a common plane which in turns align the epipolar lines in common image plane axis. More details about the calibration procedure is explained in Section 3.6.

The 3D point coordinates $P(x, y, z)$ is computed by the pixel coordinates of the left image $P_l(x_l, y_l)$ and the right image $P_R(x_R, y_r)$ along with baseline of the cameras given by the distance between the center of the left camera O_l and right camera O_r. The distance between the corresponding points in the left and right image defines the disparity d like shown in equation below,

$$d = x_l - x_r \tag{3.8}$$

From the disparity map, depth map can be obtained by,

$$x = \frac{\mathrm{B}x_l}{d}, y = \frac{\mathrm{B}y_l}{d}, z = \frac{\mathrm{B}f}{d} \tag{3.9}$$

where B is the baseline of the stereo camera.
Equation 3.10 ,3.11 and 3.12, 3.13 shows the transformation of original image coordinates (x,y) to rectified images. H_l and H_r denotes homography matrices for left and right images respectively. In this rectification step it involves multiplication of inverse 3x3 matrices. This directly related to real time performance which includes software and hardware. Moreover when left and right images are row aligned it will be easier for tracking the epipolar line and also it is possible to scan each row.

$$\begin{bmatrix} x'_l \\ y'_l \\ z'_l \end{bmatrix} = {H_l}^{-1} \begin{bmatrix} x''_l \\ y''_l \\ z''_l \end{bmatrix} \tag{3.10}$$

$$\begin{bmatrix} x'_r \\ y'_r \\ z'_r \end{bmatrix} = {H_r}^{-1} \begin{bmatrix} x''_r \\ y''_r \\ z''_r \end{bmatrix} \tag{3.11}$$

$$\begin{bmatrix} x_l \\ \\ y_l \end{bmatrix} = \begin{bmatrix} \frac{x'_l}{z'_l} \\ \\ \frac{y'_l}{z'_l} \end{bmatrix} \tag{3.12}$$

$$\begin{bmatrix} x_r \\ \\ y_r \end{bmatrix} = \begin{bmatrix} \frac{x'_r}{z'_r} \\ \\ \frac{y'_r}{z'_r} \end{bmatrix} \tag{3.13}$$

Figure 3.10 shows the disparity map formed by the stereo image. On right side of the Figure 3.10 we can see the pointcloud of the same stereo image.

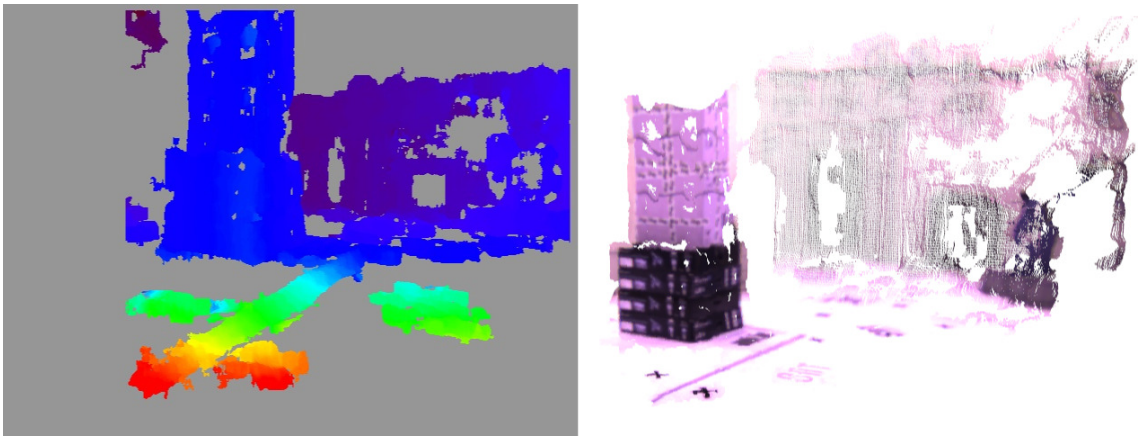


FIGURE 3.10: Disparity map (left) and 3D point clouds (right)

3.6 Calibration

Camera calibration gives the relation between the measurements in pixel and real world three dimensional points. It is used to find the intrinsic and extrinsic parameters of the camera.

- Extrinsic parameters are 3D position and orientation of the camera.
- Intrinsic parameters are focal length, image center and distortion parameters of the lens. size of the pixels.

According to [93] normally focal length of a lens has only 4 percentage of accuracy. During lens detachment and attachment, the intrinsic parameters will change.

Let's take a simple pinhole camera model for the basic understanding as shown in Figure 3.8. Normally the camera projects the three dimensional point $P(X_w, Y_w, Z_w)$ into the

two dimensional image plane as image (x_c, y_c) with O as the origin. This is called projective transformation and it can be written as

$$C = GP \tag{3.14}$$

where,

$$C = \begin{bmatrix} x_c \\ y_c \\ w_c \end{bmatrix}$$

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

$$P = \begin{bmatrix} X_w \\ Y_w \\ Z_w \end{bmatrix}$$

Typically the calibration process is done by a known pattern like chessboard. So that it can be identifed via different angles and position. Using this information, it is possible to compute the extrinsic and intrinsic parameters of the camera. Here the projection of the three dimensional homogeneous world coordinates on the image plane can be written as,

$$C_p = AW_p \tag{3.15}$$

where,

$$\begin{pmatrix} C_{p_1} \\ C_{p_2} \\ C_{p_3} \\ C_{p_4} \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{pmatrix} \begin{pmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{pmatrix}$$

By using known 3D structure, the calibration will determine the 12 elements in the matrix A. Since the image is only two dimensional there is no need to find the C_{p3}. The

Cartesian image coordinates (x_c, y_c) are,

$$x_c = \frac{C_{p_1}}{C_{p_4}} \tag{3.16}$$

$$y_c = \frac{C_{p_2}}{C_{p_4}} \tag{3.17}$$

$$C_{p_1} = a_{11}X_w + a_{12}Y_w + a_{13}Z_w + a_{14} = C_{p_4}x_c \tag{3.18}$$

$$C_{p_2} = a_{21}X_w + a_{22}Y_w + a_{23}Z_w + a_{24} = C_{p_4}y_c \tag{3.19}$$

$$C_{p_4} = a_{41}X_w + a_{42}Y_w + a_{43}Z_w + a_{44} \tag{3.20}$$

After simplification, the equation can be formed,

$$\begin{bmatrix} X_{w_1} & Y_{w_1} & Z_{w_1} & 1 & 0 & 0 & 0 & 0 & -X_{w_1}x_{c_1} & -x_{c_1}Y_{w_1} & -x_{c_1}Z_{w_1} \\ X_{w_2} & Y_{w_2} & Z_{w_2} & 1 & 0 & 0 & 0 & 0 & -X_{w_2}x_{c_2} & -x_{c_2}Y_{w_2} & -x_{c_2}Z_{w_2} \\ \vdots & & & & & & & & & & \\ X_{w_n} & Y_{w_n} & Z_{w_n} & 1 & 0 & 0 & 0 & 0 & -X_{w_n}x_{c_n} & -x_{c_n}Y_{w_n} & -x_{c_n}Z_{w_n} \\ 0 & 0 & 0 & 0 & X_{w_1} & Y_{w_1} & Z_{w_1} & 1 & -X_{w_1}y_{c_1} & -y_{c1}Y_{w_1} & -y_{c1}Z_{w_1} \\ 0 & 0 & 0 & 0 & X_{w_2} & Y_{w_2} & Z_{w_2} & 1 & -X_{w_2}y_{c_2} & -y_{c2}Y_{w_2} & -y_{c2}Z_{w_2} \\ \vdots & & & & & & & & & & \\ 0 & 0 & 0 & 0 & X_{w_n} & Y_{w_n} & Z_{w_n} & 1 & -X_{w_n}y_{c_n} & -y_{c_n}Y_{w_n} & -y_{c_n}Z_{w_n} \end{bmatrix} \begin{bmatrix} a_{11} \\ a_{12} \\ a_{13} \\ a_{14} \\ a_{21} \\ a_{22} \\ a_{23} \\ a_{24} \\ a_{31} \\ a_{32} \\ a_{33} \\ a_{34} \\ a_{41} \\ a_{42} \\ a_{43} \\ a_{44} \end{bmatrix} = \begin{bmatrix} x_{c_1} \\ y_{c_2} \\ \vdots \\ x_{c_n} \\ y_{c_1} \\ y_{c_2} \\ \vdots \\ y_{c_n} \end{bmatrix}$$

The above equation can be written as

$$DQ = R. \tag{3.21}$$

By adding transpose on D^T on both the sides,

$$D^T DQ = D^T R$$

$$Q = (D^T D)^{-1} D^T R$$

Using the above equation Q can be found. Now the matrix A can be determined using Q and all the unknowns can be estimated in this camera calibration process. Through the camera calibration the values of focal length, rotation matrix, translation matrix, distortion coefficients are estimated. Figure 3.11 shows the chessboard pattern of 9×7 squares with a scale of 0.0325 m set of images used for calibration of our stereo camera.

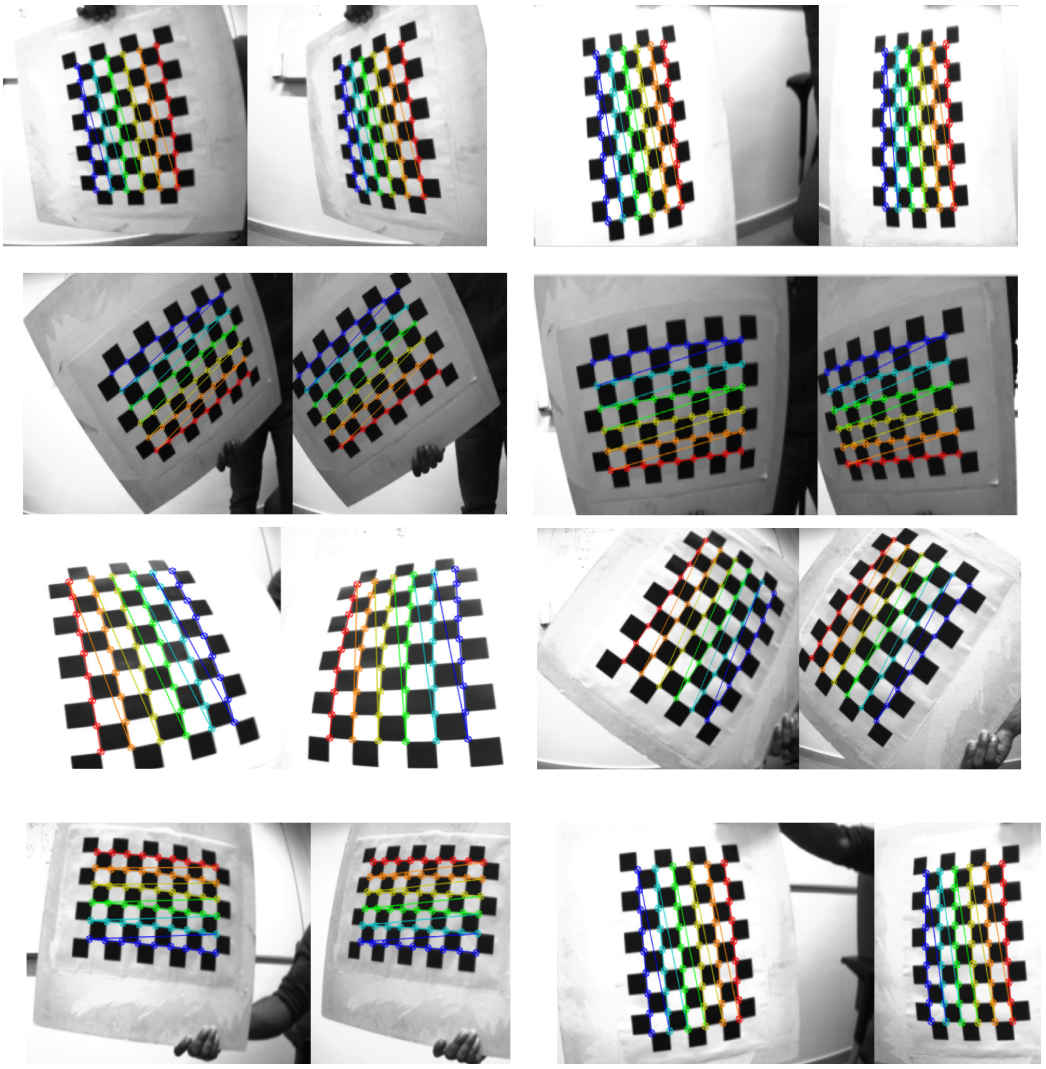


FIGURE 3.11: Calibration image samples

Calibration result

The custom made stereo camera setup is tested with different base length to check the accuracy and robustness. The general form of the projection matrix consists of the focal lengths (f_x, f_y) and the principal points (c_x, c_y) as shown below,

$$Projectionmatrix = \begin{bmatrix} f_x & 0 & c_x & T_x \\ 0 & f_y & c_y & T_y \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{3.22}$$

We showed the project matrix of one of the calibration result. Equation (3.23) and (3.24) shows the left and right camera projection matrix. In this particular case the epipolar error is 0.12.

Here the T_x, T_y gives the optical center of the right camera with respect to the left camera. It can also be explained as the distance between the optical center of left camera and right camera. For example in equation 3.24 the T_y value is -9.78 which is the exact distance between the left and right optical center of this stereo camera setup.

$$Projectionmatrix_{(leftcamera)} = \begin{bmatrix} 367.476 & 0 & 208.620 & 0 \\ 0 & 367.476 & 231.578 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{3.23}$$

$$Projectionmatrix_{(rightcamera)} = \begin{bmatrix} 367.476 & 0 & 208.620 & -9.738 \\ 0 & 367.476 & 231.578 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{3.24}$$

The accuracy of the camera calibration plays a significant role in the robustness of the depth calculation. In this thesis we used opencv approach [94] for calibrating the stereo camera.

3.7 Depth estimation using stereo image processing

The basic steps involved in stereo image processing is shown in Figure 3.12. With the input of calibrated stereo images, we need to find the correspondence of a point in the

left to the same point in the right image. This can be done by stereo matching algorithms like global matching, semi global matching methods. Correspondence matching involves search and matching which will gain computational complexity. Classic approach includes correlation window to search the points. The search can be constrained by a limit called disparity range. Basically it is formed by the horizontal scan line in the search window. Locating the common points in both the images determines the robustness and precision in the 3D reconstruction. Most of the stereo matching process involves the matching of the cost computation, cost aggregation from the initial cost, optimization and refinement. First it starts with cost computation for each pair of pixels and in the optimization process the best matching or correspondence for every pixel is determined.

The common cost aggregation methods are sum of square differences(SSD), sum of Absolute differences (SAD), Normalized cross correlation (NCC). We tested the sum of square differences (SSD) and sum of Absolute difference(SAD) methods to compute the disparity. Equation 3.25 shows the sum of square difference approach where the difference of reference pixels and target pixels and then squares and aggregates.

$$SSD = \sum_{u,v} (I_r(u,v) - I_t(u+d,v))^2 \tag{3.25}$$

Equation 3.26 shows the sum of absolute difference approach where it aggregates the pixel intensity difference between reference and target pixels.

$$SAD = \sum_{u,v} |I_r(u,v) - I_t(u+d,v)| \tag{3.26}$$

Equation 3.27 shows Normalized Cross Correlation. It uses mean values in block to normalize the cross correlation.

$$NCC = \frac{\sum_{u,v} I_r(u,v) - \bar{I}_r.(I_t(u+d,v) - \bar{I}_t)}{\sqrt{(I_r(u,v) - \bar{I}_r)^2.((I_t(u+d,v) - \bar{I}_t))^2}} \tag{3.27}$$

The primary goal is to get more accuracy as well as speed in real time. Once the pixel coordinates are located on both the images, it will be used to calculate the disparity map. This disparity map should refined because it may contain ambiguous and noise due to texture less regions or occlusions. Then by using the triangulation, 3D point clouds are generated like shown in Figure 3.10. Using the parameters such as base length of the cameras, focal length obtained from the calibration and disparity map from the stereo matching, we can estimate the 3D point in the real world using triangulation.

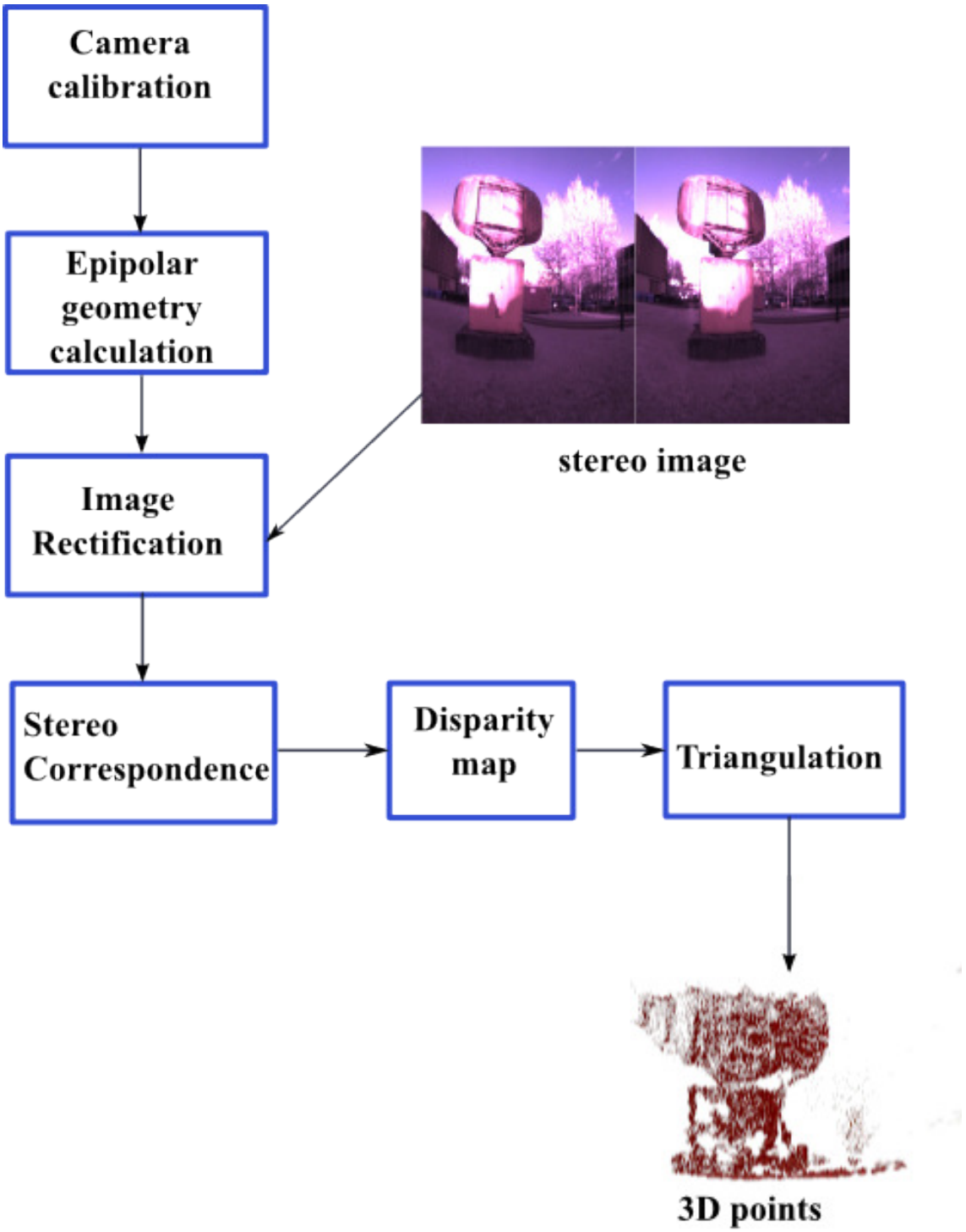


FIGURE 3.12: Overview of stereo image processing steps.

3.8 Concluding Remarks

The custom made stereo camera development and its efficiency in acquiring the depth of an objects in outdoor environment is explained in detail here. We discussed the design aspects and hardware implementation of the two custom made stereo camera models. During the real time experiments the issues like sensitivity, synchronization are discussed. Then this chapter provided an overview of the stereo camera fundamentals such as mathematical aspects of the camera calibration, stereo matching and image rectification and depth estimation method. Further more the design choice suitable for the primary objective and its quality in calculating the depth estimation influence the choice of the algorithms in the pipeline is investigated.

Chapter 4

Mapping and Localization

This chapter presents the mapping of the outdoor environment and along with localization necessary for the UAV navigation in this work. First, graph based SLAM is explained in detail. Second, the Stereo Visual Odometry is described. Results of the initial experiments conducted in the practical outdoor scenario and its analysis are presented next.

4.1 Motivation

As explained in the Chapter 2 mapping and localization is one of the primary modules for autonomous navigation of the small UAV. The motivation is to build a topological map that will allow to support the mission. An accurate map will be helpful to find the precise details of the environment including the dynamic objects. In this thesis we concentrate on mapping of the unknown outdoor environment which means without any prior information like amount of light available, arrangements of the objects and its size, color. For example, practical outdoor environment consists of objects like unstructured trees, buildings and automobiles etc. To deploy a system that can give robust information and high level understanding is the motivation for our approach. Later it will act as a planning guide for the small UAV in real time like shown in Figure 4.1.

Definition 4.1. (Unknown Environment) The conditions and circumstances around the vehicle are not known except the starting point and goal point.

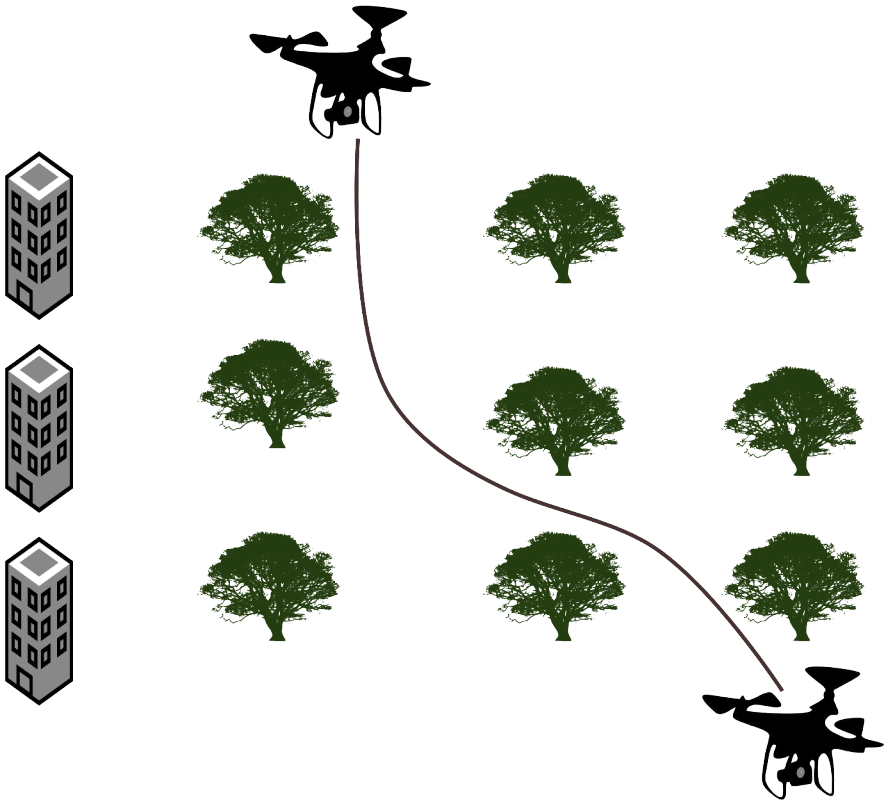


FIGURE 4.1: Illustration of the outdoor scenario.

4.1.1 Issues and Challenges

Here we will briefly address the issues and challenges in the mapping and localization problem for the UAV in outdoor environment. The challenges includes the factor where the UAV has to map and localize the outdoor environment in real time, while keeping the on board computing and memory capabilities reliable. Likewise it has also been an issue for the algorithm to map and discard the information about moving objects. For example if the environment consists of 3 to 4 moving objects at different speed, mapping all the moving objects and discard it while it is not present at that position anymore. If the UAV is deployed in these kind of scenarios, we need a systematic algorithm that can map and then update the moving object position in real time.

Other challenges include the ability to handle uncertainties and ambiguities in the environment and achieve robustness and efficiency in data association. Because in outdoor environment it is common to have identical objects like trees with same color and structure. But these objects are seperated with few meters to distinguish between themselves. This will lead to perceptual aliasing.

4.2 Graph based SLAM

As mentioned in Chapter 2 SLAM algorithm can be divided into online and full SLAM [40]. As shown in the equation (4.1) online SLAM estimates the posteriori distribution of current pose $\overline{\mathrm{p}}_t$ and map m_t provided the measurements $z_{1:t}$ and $u_{1:t}$ upto the time,

$$P(\overline{\mathrm{p}}_t, m|z_{1:t}, u_{1:t}). \tag{4.1}$$

The posterior distribution of the entire trajectory can be estimated in full SLAM as,

$$P(\overline{\mathrm{p}}_{0:t}, m|z_{1:t}, u_{1:t}). \tag{4.2}$$

where $\overline{\mathrm{p}}_{0:t}$ denotes all poses including current time.
Since we want to implement the algorithm in outdoor unknown environment, graph based SLAM can be used as it can handle the problem with accuracy and robustness. The framework of graph based SLAM consists of nodes and edges to represent the pose as shown in Figure 4.2. Most details about the graph based SLAM can be found in [95] and [96]

Generally, pose of the UAV is stored in the form of nodes as

$$\overline{\mathrm{p}} = (\mathrm{p}_1, \mathrm{p}_2, \mathrm{p}_3 \ldots . \mathrm{p}_n)^T \tag{4.3}$$

and the links between the nodes act as relative motion constraints between them. The main goal is to find the pose and map by maximizing the posterior of the full SLAM as given below

$$[\overline{\mathrm{p}}^*, m^*] = \arg\max_{\overline{\mathrm{p}}, m} P(\overline{\mathrm{p}}_{0:t}, m|z_{1:t}, u_{1:t}). \tag{4.4}$$

The maximum can be found by expanding the posterior in the recursive form [95]as,

$$P(\overline{\mathrm{p}}_{0:t}, m|z_{1:t}, u_{1:t}) = \nu P(z_t|\overline{\mathrm{p}}_t, m) P(\overline{\mathrm{p}}_t|\overline{\mathrm{p}}_{t-1}, u_t) P(\overline{\mathrm{p}}_{0:t}, m|z_{1:t}, u_{1:t-1}), \tag{4.5}$$

where ν is the normalizer. Considering it for all times t, we get the closed form as,

$$P(\overline{\mathrm{p}}_{0:t}, m|z_{1:t}, u_{1:t} = \nu P(\overline{\mathrm{p}}_0, m_0) \prod_t P(\overline{\mathrm{p}}_t|\overline{\mathrm{p}}_{t-1}, u_t) P(z_t|\overline{\mathrm{p}}_t, m) \tag{4.6}$$

$$P(\overline{\mathrm{p}}_{0:t}, m|z_{1:t}, u_{1:t} = \nu [P(\overline{\mathrm{p}}_0, m_0) \prod_t P(\overline{\mathrm{p}}_t|\overline{\mathrm{p}}_{t-1}, u_t) P(z_t^i|\overline{\mathrm{p}}_t, m)]. \tag{4.7}$$

In the above equation (4.6), $P(\overline{\mathrm{p}}_0, m_0)$ denotes the prior of the initial pose $\overline{\mathrm{p}}_0$ and map m. Initially there is no prior information about the map. Prior can be factorized into independent $P(\overline{\mathrm{p}}_0)$ and $P(m_0)$ and placed with normalizer ν.

With an assumption that the vehicle motion and measurements have a Normal (gaussian) distribution, we have the following expressions,

$$P(\overline{\mathrm{p}}_t|\overline{\mathrm{p}}_{t-1}, u_t) = \nu exp\left\{-\frac{1}{2}(\overline{\mathrm{p}}_t - c(\overline{\mathrm{p}}_{t-1}), u_t)^T Q_t^{-1}(\overline{\mathrm{p}}_t - c(\overline{\mathrm{p}}_{t-1}), u_t)\right\} \tag{4.8}$$

$$P(z_t^i|\overline{\mathrm{p}}_t, m) = \nu exp\left\{-\frac{1}{2}(z_t^i - l(\overline{\mathrm{p}}_{t-1}, m_t))^t R_t^i(z_t^i - l(\overline{\mathrm{p}}_{t-1}, m_t))\right\} \tag{4.9}$$

$$P(\overline{\mathrm{p}}_0) = \nu exp\left\{-\frac{1}{2}\overline{\mathrm{p}}_0^T \Sigma_0 \overline{\mathrm{p}}_0\right\} \tag{4.10}$$

where c and l are the motion model and measurement model respectively. We now compute the negative logarithm posterior as given below,

$$-\log P(\overline{\mathrm{p}}_{0:t}, m|z_{1:t}, u_{1:t}) = \text{const.} + \log P(\overline{\mathrm{p}}_0) + \sum_t \log P(z_t^i|\overline{\mathrm{p}}_t, m). \tag{4.11}$$

All the quadratic terms are summed up from equations (4.8) (4.9) (4.10) in

$$[\overline{\mathrm{p}}^*, m^*] = \underset{\overline{\mathrm{p}}, m}{\operatorname{argmax}} P(\overline{\mathrm{p}}_{0:t}, m|z_{1:t}, u_{1:t})$$

to the following maximization,

$$[\overline{\mathrm{p}}^*, m^*] = \underset{\overline{\mathrm{p}}, m}{\operatorname{argmax}} \sum_{ij} e_{ij}^T \Omega_{ij}^{-1} e_{ij}, \tag{4.12}$$

where e_{ij} is the error vector and Ω_{ij}^{-1} is the information matrix. The poses are calculated using the stereo data i.e extracting the interest points in the each image. Depending on the environment this can be trees, buildings, humans or objects present at that time. Typically, the constraints are the rigid body transformation and measurements of the features. For example the movement of the small UAV from one position to another requires translation and rotation transformation to calculate the pose. These nodes and edges altogether forms the graph which are referred as "front-end". The created graph has a configuration which need to be optimized. The front end forms the constraints by interpreting the sensor data. The back-end of the graph SLAM finds the configuration of the nodes by computing the maximum likelihood of the map and optimizes it.

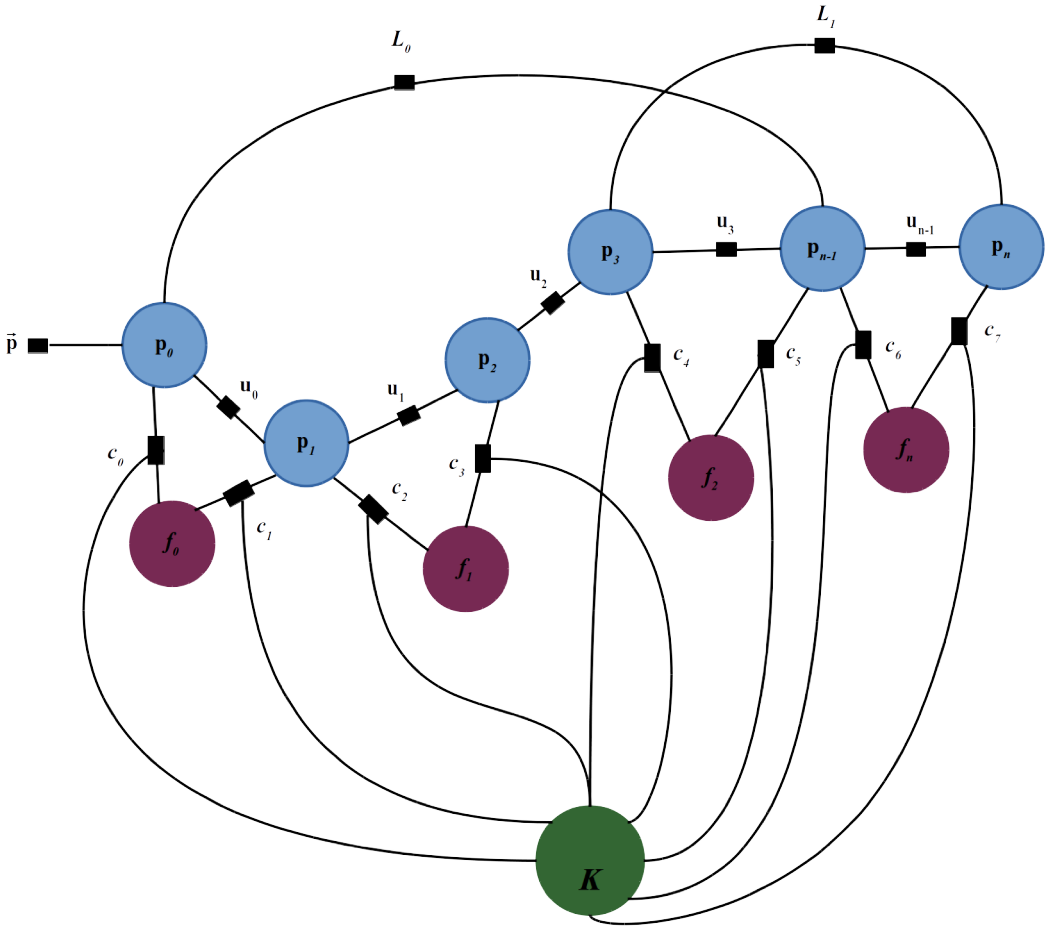


FIGURE 4.2: Graph representation as nodes and edges: p_1, p_2, p_3, p_4 ... p_n in Blue circle denotes the UAV pose, f_1, f_2, f_3, f_4 in color canonical aubergine denotes the relative features, L_0,L_1 shows the loop closures, K is the intrinsic calibration parameters, u_1, u_2...u_n denotes the corresponding Odometery constraints and p represents prior factors,c_1, c_2, c_3, c_4, c_5, c_6, c_7 are the variables associated with intrinsic calibration parameters.

4.2.1 Front-end SLAM

The Front-end of graph based SLAM constructs the nodes and edges from the stereo images. Besides that the key tasks that are undertaken in this phase includes the feature extraction and the Loop Closure Detection.

4.2.1.1 SURF: Feature detection and Generation

As mentioned in Chapter 2 in Section 2.2.2 SURF algorithm performs fast and efficient in outdoor environment. Due to that reason we implemented SURF algorithm for feature detection and description. We discuss the further details briefly in this section.

The Speed up robust features (SURF) algorithm process consists of two steps which are detection of the interest points in the image frame and specifying the description for the interest points. The descriptors can be matched sequentially at each time step. The initial step is to calculate the image convolution. After that the features are detected using the Hessian matrix. To increase the computational speed integral images are used. According to the SURF algorithm [16] for a chosen point $I(x, y)$ in an image I, the Hessian is calculated as,

$$H(X, \sigma) = \begin{vmatrix} L_{xx}(X, \sigma) & L_{yy}(X, \sigma) \\ L_{xy}(X, \sigma) & L_{yx}(X, \sigma) \end{vmatrix}. \tag{4.13}$$

where,

- $L_{xx}(X, \sigma)$ denotes the convolution of the gaussian second order derivative $\frac{\partial}{\partial x^2} g(\sigma)$ in point X. Likewise for $L_{yy}(X, \sigma)$, $L_{xy}(X, \sigma)$ and $L_{yx}(X, \sigma)$.
- Point $X = (x, y)$
- σ is the scale parameter (amount of blur)

The interest point descriptors are generated by the Haar wavelet response. The calculated interest points are matched in the successive images precisely. For example the correspondence of the interest points for two images are calculated by measuring euclidean distance between the respective feature point descriptor estimated in the previous image, if the measured distance is smaller than 0.6 times to the second nearest neighborhood. Since the SURF algorithm uses box filter and integral images the filter can be applied at any size directly.

To generate the feature, a keypoint in the image is internally selected by the algorithm. Around each keypoint a 16×16 window is chosen which will later divided into 4×4 size windows. Gradient and orientation are calculated for each point, which will be then moved to a 8 bin histogram in a 128-dimensional vector as shown in Figure 4.3.

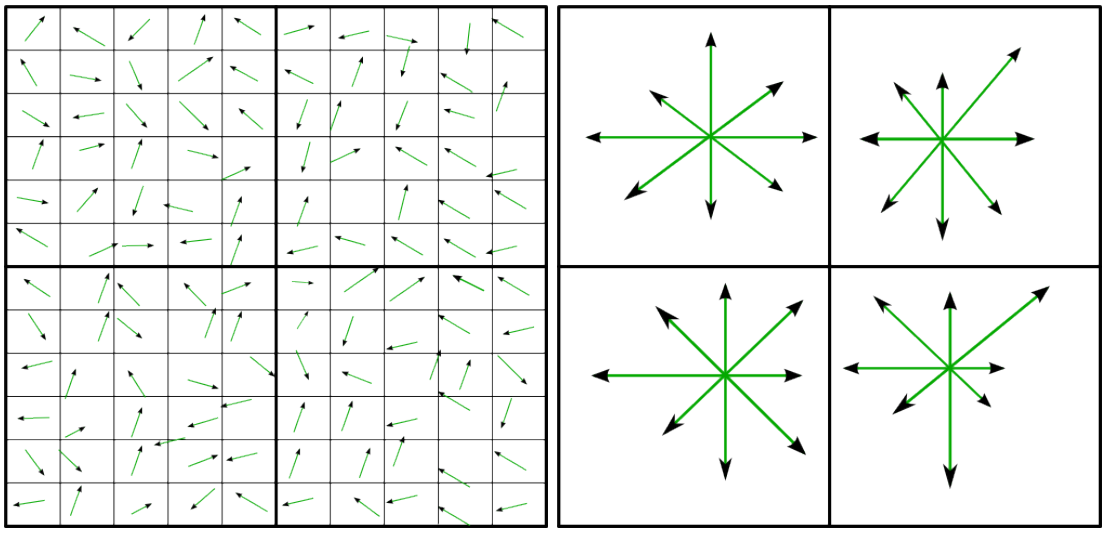


FIGURE 4.3: Image gradient and key point descriptor.

Depending on the range of the gradient and orientation, these values are added to the bin which is 128 dimensional vector. For example, the range of 0 to 44 degrees covers the first bin and the second bin covers the range of 45-89 degrees. The number of features included depends on the magnitude of the gradient. The rotation of the keypoints are subtracted from each orientation. At the end $4 \times 4 \times 8 = 128$ numbers are calculated and then normalized to generate the "feature vector". An example of the implemented SURF algorithm in stereo images is shown in Figure 4.4. In Figure 4.4 the color yellow circle represents the new feature and the pink shows corresponding features present in the left and right images respectively.

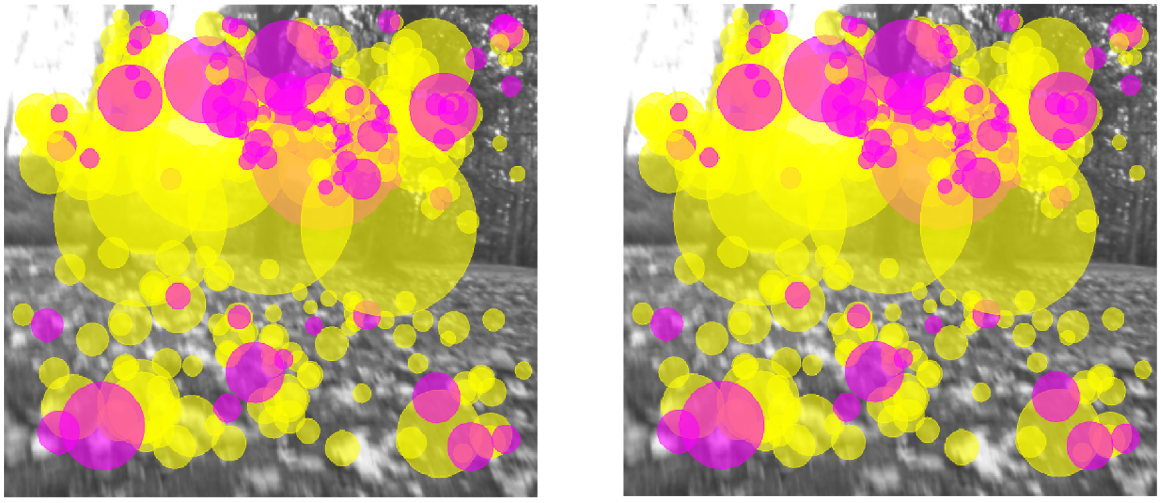


FIGURE 4.4: Example of feature detection in left and right image.

4.2.1.2 Loop Closure Detection or Visual place recognition

Recognizing a place that is already captured using the visual data is a challenging problem due to the changes in the real world environment over the time. For example the appearance of the object in the outdoor environment can vary due to the factors like the distance at which the object is perceived by the sensor and contrast lighting (sunlight) as well as shadows projecting on the objects. Our focus is to implement the online loop closure detection incrementally. The process checks if the current image has similarity or it is close to the view point of the past images. Here, the challenges are the computational power, speed, image acquisition time and detection rate. In order to identify the images with the previous set of images, we need a model that can verify and ignore already captured images. Here we used Bag of Words methods to model the dictionary that contains the features detected before.

Bag of Words:

A database of visual words formed by the extraction of features from the stereo images on-line is referred as visual dictionary. The stereo image features generated are enormous amount of data which needs to be clustered. For memory and computational reasons, clustering would be beneficial. Here kd-tree and nearest neighbor distance ratio are used to find the best possible values. The algorithm records the counts of each term that appears in the image to create a normalized histogram representing a term vector. This term vector is the Bag of Features representation of the image. The Advantage of using the bag of words approach is that it is pose invariant which means a place can be recognized irrespective of the small UAV current position in the environment. The challenges in the loop closure detection involves the image retreival task online from the past images and matching it with the current one. To explain it into details, the process starts to check with past images the amount of similarity the current image I_t has or is close to the view point of the previous images. This problem can be addressed by using the probability estimation using bayesian framework.

Here C_t is the loop closure hypothesis at time t. The event $C_t = i$ satisfies the event that current image I_t detects the loop closure by matching with the past images. If $C_t = -1$ then there is no loop closure detected. Basically it is a search task from the past images I_s as

$$s = \underset{i=-1,\ldots,t-n}{\operatorname{argmax}} P(C_t = i|I^t) \tag{4.14}$$

where $I^t = I_0, I_1, I_2, ...I_t$, are the n - neighbor images. Since the search in the neighbor images always had similarity, we are not searching on the neighbor images n. The value

of n is chosen as 20 images, because the frame rate of our image acquisition is 20 fps. This can be modified if the speed of the vehicle is more than the image acquisition. The full posterior can be estimated by Bayesian rule,

$$P(C_t|I^t) = \nu P(I_t|C_t)p(C_t|I^{t-1}) \tag{4.15}$$

where ν is the normalization term. The dictionary is formed by the features created by the SURF which will increment over the time. The state of the dictionary D_f formed by the features with incremental time index i is shown as

$$(D_f)_0 \subseteq (D_f)_1 .... \subseteq (D_f)_{t-1} \subseteq (D_f)_t \tag{4.16}$$

In the above equation f is the feature space.Since the SURF algorithm uses box filters it forms several features. The general form to represent the respective feature of the image in the whole dictionary, we can write d_{f_i} for the image I_i. Now the full posterior can be expressed using the features formed in the dictionary,

$$P(C_t|D_t^n) = \nu(D_t^n|C_t)P(C_t|D_{t-1}^n) \tag{4.17}$$

$$P(C_t|D_t^n) = \nu \left[\prod_{f=0}^{n} P((D_f)_t|C_t) \right] P(C_t|(D^n)^{t-1}) \tag{4.18}$$

$$P(C_t|D_t^n) = \nu \left[\prod_{f=0}^{n} P((D_f)_t|C_t) \right] \underbrace{\sum_{ij}^{t-p} P(C_t|C_{t-1=j})P(C_{t-1}=j)|(D^n)^{t-1}}_{belief} \tag{4.19}$$

In equation (4.19) $(C_t|C_{t-1})$ gives time evolution model. The actual estimation of full posterior is obtained by applying the time evolution model to the previous posterior. This will update the sequence of the features D_f^n generating over time. Conditions for the loop closure detection includes the following:

- No loop closure is detected at time t:

$$P(D_t = -1|D_{t-1} = -1) = 0.9$$

- Low probability of loop closure not detected at time t:

$$P(D_t = i|D_{t-1} = -1) = \frac{0.1}{(t-p)+1}$$

with $i \in [0; t-p]$.

- Loop closure at time $t-1$ with no loop closure at time t:

$$P(D_t = -1 | D_{t-1} = j) = 0.1 \text{with} j \epsilon [0; t-p]$$

  With $j \in [0; t-p]$.

- with $i, j \in [0; t-p]$

$$P(D_t = i | D_{t-1} = j)$$

  the standard deviation is non-zero for $i = j-2 \ldots j+2$.

For the image retrieval we used the bag of words model as shown in equation (4.5). We modified the significant parameters like speckle range, uniqueness ratio, number of iterations for computational and memory efficiency. While detecting the features in the image, we designed an adaptable threshold value of 30 to 70 to ignore the image in the dictionary. So that the image with less features doesn't create memory burden. To speed up and make it more efficient the detection rate of 20 hz is set and it can be adjusted according to the available power. Specifically when the UAV is moving we consider the loop closure detected with nearest one location. This will reduce the search in the dictionary which in turn reduces computation burden. So that it will satisfy the online requirements of the onboard computer[B] mounted on small UAV for processing the complete software pipeline.

Figure 4.5 shows an example of loop closure detected (denoted as L and C in green and brown color letter respectively) between node number 47 and 122, 127 and 96.

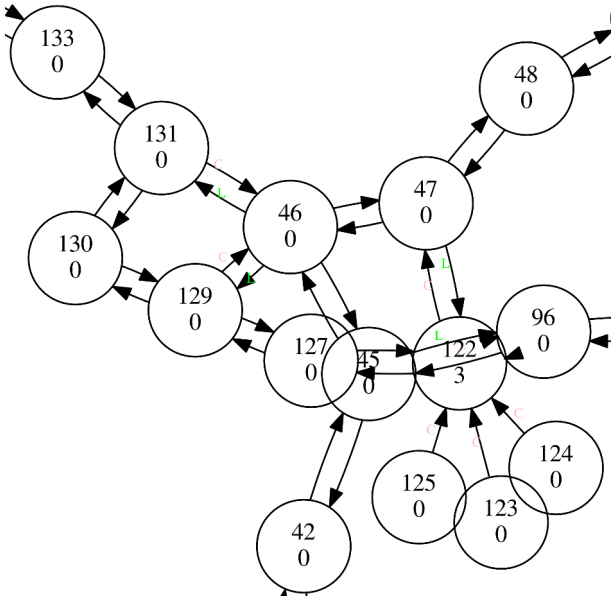


FIGURE 4.5: Loop closure detection between nodes

The Figure 4.6 shows the overview of the process of loop closure detection implemented here.

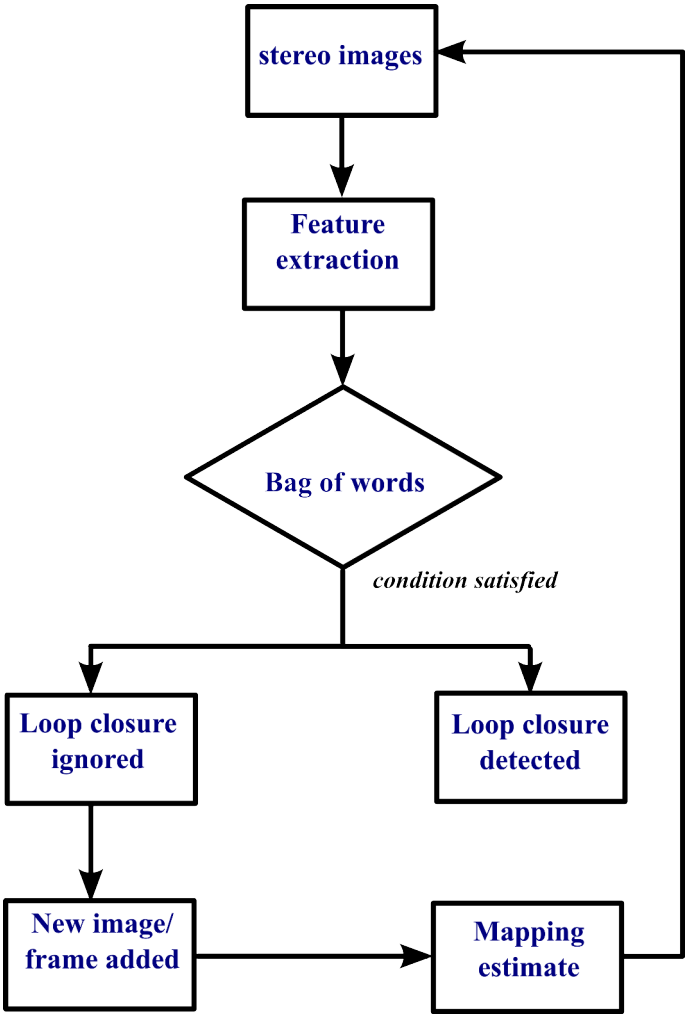


FIGURE 4.6: Schematic process of the loop closure detection

4.2.2 Back-end SLAM

As explained in the Section 4.2.1 front-end of the SLAM creates the graph with nodes and edges. The main objective of the back end SLAM is to find the configuration of the nodes by optimizing it. Because of the non-linear measurements in SLAM, this optimization problem is non-convex. Here the general graph optimization g2o is used. The error function $e_{ij}(\mathrm{p})$ between the two nodes i and j can be obtained as the difference between the observed measurement z_{ij} and the expected measurement $\hat{z}(\mathrm{p}_i, \mathrm{p}_j)$ as

$$e_{ij}(\mathrm{p}) = \hat{z}(\mathrm{p}_i, \mathrm{p}_j) - z_{ij} \tag{4.20}$$

But the measurements $\hat{z}$ and pose p are not in euclidean space. To perform this operation in non-euclidean space, new operator $\boxplus$ is defined in [97] to compute the difference in variables from different domain,

$$\tilde{e}_{ij}(\mathrm{p}) = \hat{z}(\mathrm{p}_i, \mathrm{p}_j) \boxplus z_{ij} \tag{4.21}$$

The measurement function $\hat{z}$ depends on the stereo image data setup. If two nodes has poses then the constraints are the transformation between the poses. For the pose to features correspondence we have to minimize the re-projection error of the observed feature in the frame of observed pose. The error minimization function is written as

$$\mathrm{p}^* = \arg\min_{\mathrm{p}} \sum_{ij} e_{ij}(\mathrm{p})^T \Omega_{ij}^{-1} e_{ij}(\mathrm{p}) \tag{4.22}$$

where,

- Ω_{ij} is the covariance matrix
- Ω_{ij}^{-1} is the information matrix between the nodes p_i and p_j
- p^* is the optimal configuration of the nodes with minimum error

For 3D slam the translation vector and quaternion can be used to represent the increments of $\triangle \mathrm{p}_i$ to $\mathrm{p_i}$ by an operator $\oplus$ defined in [98]

$$\breve{\mathrm{p}} \boxplus \triangle \mathrm{p}_i{}^* \overset{def.}{=} \breve{\mathrm{p}}_i \oplus \triangle \mathrm{p}_i{}^* \tag{4.23}$$

The error function using the new operator can be defined as,

$$e_{ij}(\triangle \mathrm{p}_i, \triangle \mathrm{p}_j) \overset{def.}{=} e_{ij}(\breve{\mathrm{p}}_i \boxplus \triangle \mathrm{p}_i, \breve{\mathrm{p}}_j \boxplus \triangle \mathrm{p}_j) = e_{ij}(\breve{\mathrm{p}} \boxplus \triangle \mathrm{p}) \simeq J_{ij} \triangle \mathrm{p} \tag{4.24}$$

where the Jacobian J_{ij},

$$J_{ij} = \frac{\partial e_{ij}(\breve{\mathrm{p}} + \triangle \mathrm{p})}{\partial \triangle \mathrm{p}}|_{\triangle \mathrm{p}=0} \tag{4.25}$$

is the approximate cost function in a quadratic form. The minimum of this quadratic form can be obtained by solving,

$$H \triangle \mathrm{p}^* = -b \tag{4.26}$$

where,

- $H = \sum_{ij} J_{ij}^T \Omega_{ij}^{-1} J_{ij}$
- $b = \sum_{ij} J_{ij}(\mathrm{p})^T \Omega_{ij}^{-1} e_{ij}$

4.3 Stereo Odometery

The Odometery of the small UAV is estimated using the sequence of the stereo images as input. Since the stereo camera model is derived using the multi-view geometry, the intrinsic and extrinsic parameters are retrieved. Using this knowledge of the rotation and translation matrix i.e R,t should be estimated between each stereo pair. Now this will give the motion of the small UAV.

Algorithm 1 Stereo Visual Odometry

1: **procedure** Stereo Visual Odometry
2: I_l^t, $I_r^t \leftarrow ImageAcquistion$
3: Image Rectification $\leftarrow$ this step projects the left and right image in a common plane. This will be useful to get the corresponding points
4: Extract the features using SURF algorithm $\leftarrow$ Quantize the features
5: compute disparity by using the sum of square differences(ssd) $\leftarrow$ Form the nearest neighbor index
6: Point cloud generation $\leftarrow$ Possible number of features
7: Estimate the rotation R and translation t

Here we used the library LIBVISO2 [99] for ego motion estimation. Typically the input for the odometry are the consistent features from the stereo images. The features are already computed as explained in Section 4.2.1.1. By using that we estimate the ego motion of the detected features in the left and right images. These features will be matched and 3D points are computed using camera model. Since SURF uses the integral images which in turn gives you blob detection, this library uses non-maximum and non-minimum suppression [100] which categorize the features based on the size of the blob like maximum, minimum. Assuming squared pixels and zero skew, the reprojection into the current image is given by

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_u \\ 0 & f_y & c_v \\ 0 & 0 & 1 \end{bmatrix} \left[(R_x(r_x)R_y(r_y)R_z(r_z))(t_x, t_y, t_z) \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} - \begin{bmatrix} s \\ 0 \\ 0 \end{bmatrix} \right] \tag{4.27}$$

where,

– $(u\ v\ 1)^T$ represents the homogeneous image coordinates

– f - focal length

– (c_u,c_v) - principal point

– $(R_x(r_x)R_y(r_y)R_z(r_z))$ - Rotation matrix $R(\mathrm{r})$

– (t_x, t_y, t_z) - Translation vector t

– $(x\ y\ z)^T$ - 3D point coordinates.

Now considering Gauss Newton optimization to minimize re-projection errors

$$\sum_{i=1}^{N} ||(x_i^l - \pi^l(X_i; r, t))||^2 + ||(x_i^r - \pi^r(X_i; \mathrm{r}, \mathrm{t}))||^2 \tag{4.28}$$

where, x_i^l, x_i^r are the feature positions in the left image and right image respectively, the minimization is performed with respect to the transformation parameters (r, t). The Jacobian J_π^l J_π^r can be derived from equation (4.27). Since the resolution of the image is 640×480 pixels, it produces more features. To increase the robustness of the system by rejecting potential outliers it used the Random Sample Consensus (RANSAC) method.

A Standard kalman filter is used for the estimation by assuming constant acceleration. Here the velocity vector v obtained like shown in equation

$$\mathrm{v} = (\mathrm{rt})^t/\Delta_t \tag{4.29}$$

$$\frac{1}{\Delta_t}\begin{bmatrix} v \\ a \end{bmatrix} = \begin{bmatrix} I & 0 \end{bmatrix}\begin{bmatrix} v \\ a \end{bmatrix}^t + \upsilon \tag{4.30}$$

where υ is the measurement noise.

The measured covariance value of the pose is shown in equation (4.3)

$$\begin{bmatrix} 0.1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.17 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.17 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.17 \end{bmatrix}$$

In the current process we applied the frame to frame matching. During the initial experiments we observed odometry was lost in environment with less feature. To tackle this we used odometry reset where the odometry will start from the last frame the value was estimated. This strategy helped fast movement of the camera with frame rate of 20 hz. Also it depends on the orientation changes of the stereo camera.

4.4 Analysis and observation of the experiment

Real time experiments are conducted to evaluate and analyse the performance, accuracy and robustness of the algorithm. All the experiments are conducted in practical outdoor scenario. In most of the experiments we consider the trees present in the environment as an valid obstacle for the small UAV navigation. Even though the scenarios are complex with uneven and unstructured objects present, it helps to check the frequent update of the map with respect to the movement of the stereo camera.

Initial testing are done using hand-held stereo camera movements and 16 data sets are collected in the form of rosbag[C]. Out of 16, 4 datasets results are presented here. These datasets are some of the complex test case scenarios suitable for analysis and discussion. All the data are taken in the outdoor environment with the complex dataset which includes unstructured trees, branches with different illumination condition as well as open space with large iron structure. In the cluttered environment walking humans are introduced in the data to check the moving object detection and update. To understand the complex behavior of the moving object, slow movements as well as average movements of the camera are handled. The important reason for choosing this kind of environment is to interpret the distance accuracy between the structures so that safety distance can be estimated later for complex maneuvering without uncertainty.

Next to that, shape and size of the structure generated are compared manually to the original physical structure. Three dimensional octomap for mapping and Stereo Visual Odometry for localization are shown in this section.

Figure 4.7 shows sample the 2D grid which is used for visualization of the 3D octomap. Each square in this 2D grid is 1 m of size in x and y scale. The octomap generated from each experiment is placed on the top of this type of grid.

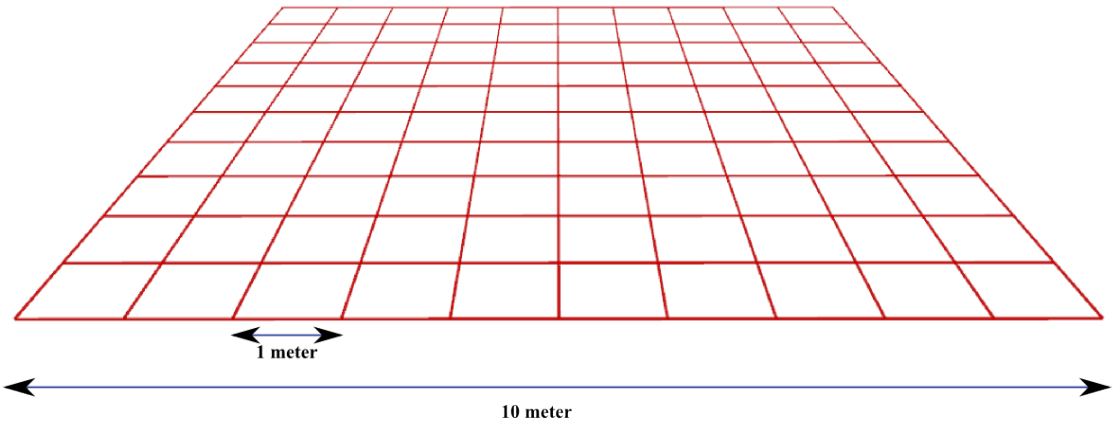


FIGURE 4.7: 2D Grid used for visualization

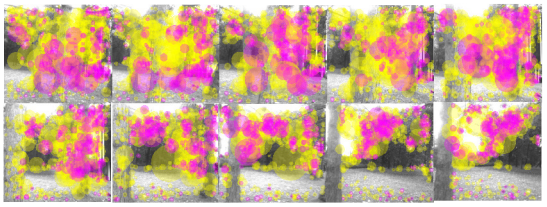
4.4.1 Input images

The stereo input images are taken using the custom stereo camera setup with different resolution ranging from 640 × 480, 540 × 480, 320 × 240 with different frame rates like 60, 30, 20 frames per second (fps). Here 15 frames per second (fps) provides reasonable matching between the left and right images features. Because the stereo camera is not doing fast maneuvering. The frame rate has an impact in the data association as explained in front end SLAM in Section 4.2.1. The Figures 4.8a and 4.8b show the 10 left images and 10 right images with resolution of 540 × 480 at the frame rate of 15 fps. From the Figures 4.8a and 4.8b we can infer the difference in the position of the tree with respect to the previous frame. Since the camera is in motion we want to check the minimum distance the algorithm needs for the feature detection.

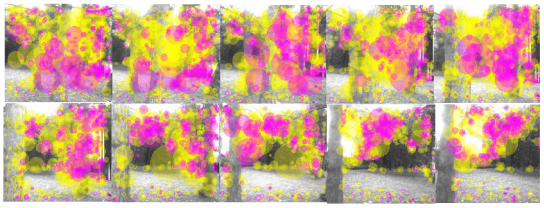
(a) Left Images

(b) Right Images

(c) Detected features in Left Images



(d) Detected features in Right Images



(e) Feature matching correspondence in left and right images.

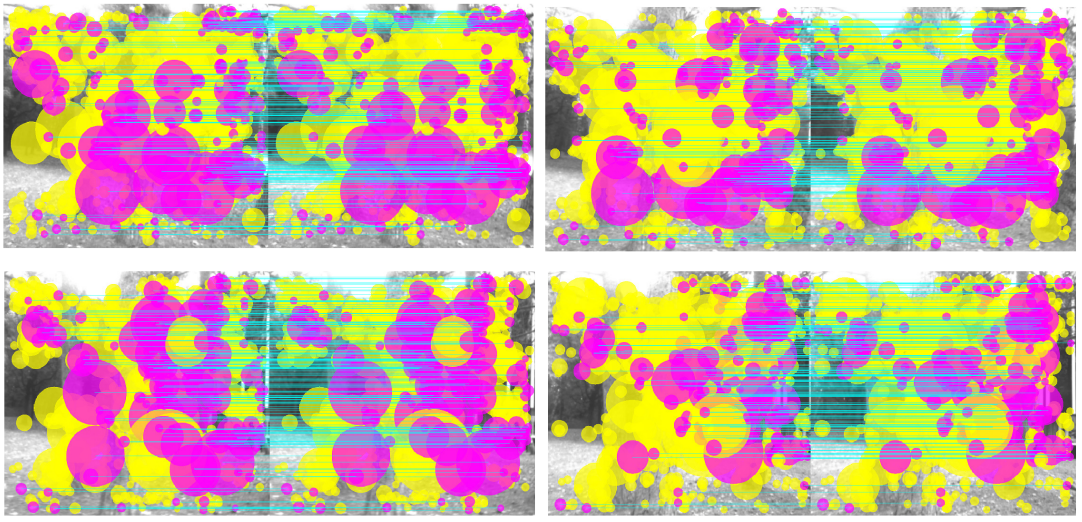


FIGURE 4.8: Left and Right image feature detection and correspondence between them.

4.4.2 Detection and matching of the features

Once the input images are taken, SURF algorithm is applied to detect the interest points i.e features. As shown in Figures 4.8c and 4.8d, the features are detected which are shown in yellow and red circles. Yellow color denotes the features that are unique (new) and pink color denotes the stable features found in the both the left and right images. The size of the blob descriptor starts from 13 to 132. Here the size of blob directly influence the Stereo Visual Odometry.

x	y	**Hessian**	**Size**	**Octave**
56.4404	271.756	1335.27	83	2
197.934	93.726	2579.61	83	2
121.912	246.09	2164.07	115	3
160.583	302.449	1721.61	62	2
212.164	367.051	7455	132	3
321.772	354.212	6504.61	115	3
240.505	398.811	6439.72	14	2
203.184	398.811	9800.04	27	1
360.888	384.173	4726.63	82	2
346.929	151.803	8214	84	2
366.38	189.198	1913.45	82	2
184.489	48.9825	5055.95	31	1
194.412	282.131	4697.53	86	2

TABLE 4.1: Values obtained for the features in terms of x, y, hessian, size and Octave.

The detected key points are matched in the left and right image as shown in Figure 4.8e. The green line in the images shows the matching between two images. The stable features in red are matched accurately when compared to the other features. Since the environment has identical objects like trees, leaves and grasses feature matching works much faster in tracking in the consecutive frames.

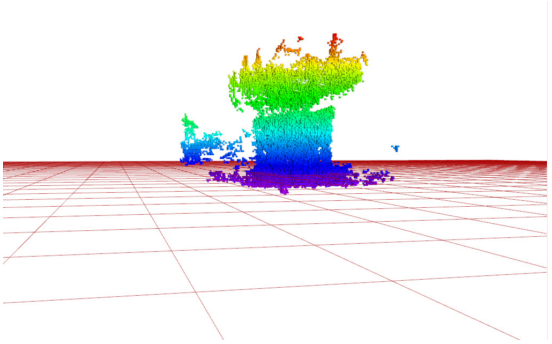
As shown in the Table 4.1, if the hessian matrix has higher value, the features will have greater stability. Also we can observe the size ranging from 14 to 132 with octave starting from 1 to 3. The distinguished features are taken for the visual dictionary for the loop closure detection using the bag of words approach.

4.4.3 Dataset 1 (Iron Structure and Bin): Map and graph

This experiment is conducted in a scenario where a big iron tank and a bin is present as shown in Figure 4.9a. The bottom of the iron tank structure has rectangle shape with wing like pattern on the top of it. The octomap generated has the same structure and shape with little speckles due to the sunlight as shown in Figure 4.9b. The Figure 4.9c shows the position of the stereo camera while moving in the environment. The dust bin which is located besides is also captured with the grass on the ground with spurious data behind. This spurious data and speckles are considered as noises which will be ignored while calculating the free and occupied space. Since the baseline of the stereo camera is 23.05 cm, it can capture up to 12 m approximately. To distinguish the objects we limit the maximum range up to 7 m for the octomap to check the precision.

(a) Scenario Image

(b) 3D Octomap



(c) Stereo Visual Odometry

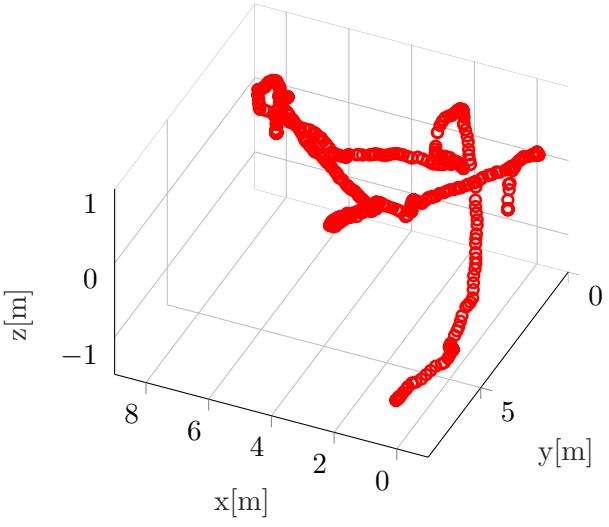


FIGURE 4.9: Image showing the scenario and its equivalent 3D Octomap with Stereo Visual Odometry.

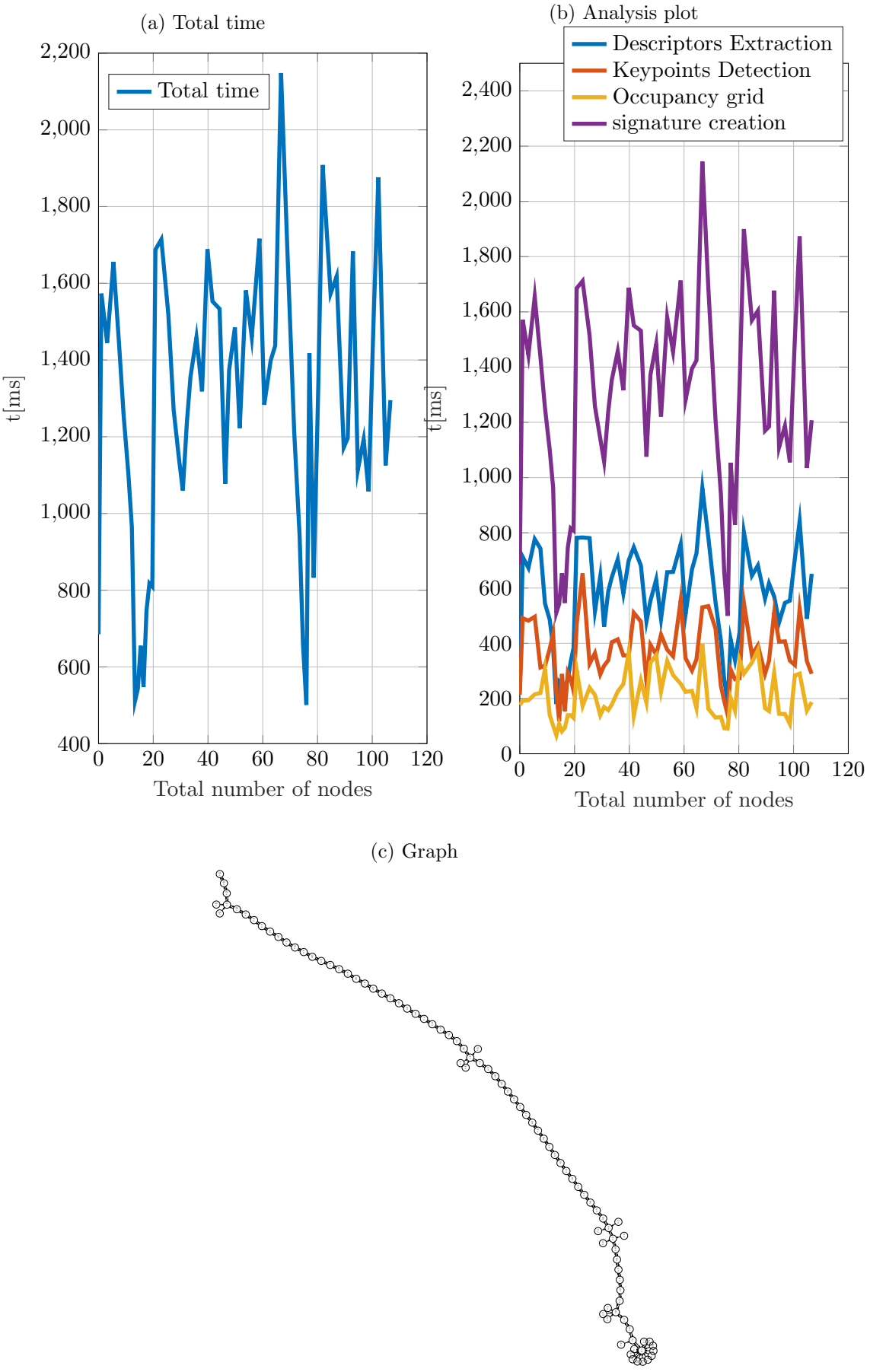


FIGURE 4.10: Plot showing the total time taken to compute, keypoint descriptor, keypoint detection, occupancy grid, signature vs total number of nodes generated and the Graph shows the nodes and edges.

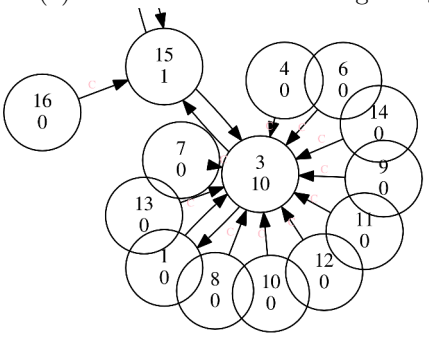
The Figure 4.10a shows the total time to compute each node starting from 0 to 106. Maximum time occurred at node number 64 with 2148 ms. This occurred when the camera is near i.e less than 0.5 m to the dust bin with wall behind of it. The high matching correspondences between the left and the right images influences the computation in terms of time. Because the stereo camera is near to object, both the left and images overlap with more than 70 percentage of similarity.

This increases the point cloud processing time for the 3D points calculation. The total feature size takes 1446.912 kb of memory. In Figure 4.10b we can observe the time taken for descriptors extraction at the same node 64 is high i.e 962 ms. While the occupancy grid took an average of 186 ms to generate the 3D map. The total length of the odometry is about 32.6886 m. While moving the stereo camera, the drifts and fast movements results in odometry loss. We estimate the odometry from the previous images where the odometry loss occurs. In this case it can use the last 2 to 3 frames approximately to get back the odometry.

As shown in Figure 4.10c the graph shows the nodes generated. Initially the nodes starting from 0 to 14 are generated around a particular place like in Figure 4.11a. While moving it generates nodes in the direction of the stereo camera.

From these experiments, we inferred that the noises due to the long range of the baseline can be adjusted by setting the maximum range for taking the point clouds into account. The shape of the structure shown in the octomap is nearly similar. Since we focus towards the collision avoidance for the small UAV the position of the obstacle is more important than the finest shape of the obstacle.

(a) Nodes formed at the beginning.



(b) Nodes formed at the end.

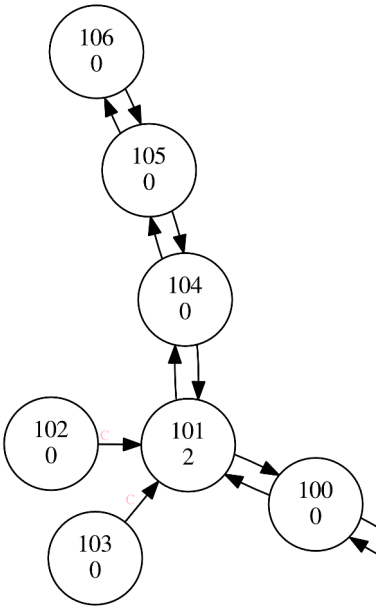


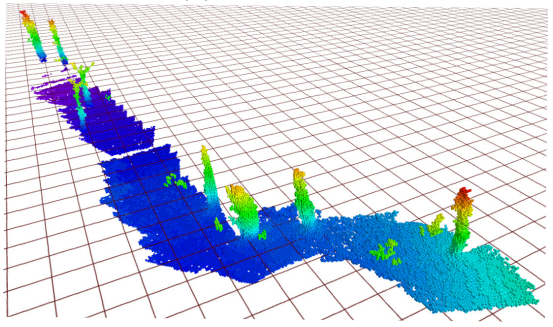
FIGURE 4.11: Graph nodes in the starting and end time.

4.4.4 Dataset 2 (Set of Trees): Map and graph

This data is taken by moving the stereo camera along the trees where the structures are not closely arranged with respect to one another. As shown in Figure 4.12a we can see the set of unstructured trees all over the place. Figure 4.12b shows the octomap where the trees position and its distance from other trees can be seen clearly. The total distance covered in this experiment is about 29.0571 m and the Stereo Visual Odometry shows the position of the camera as in Figure 4.12c. The generated graph includes 54 nodes and shape of the graph itself denotes that there are no nodes visited again. The total visual features comprises the size of 1434.880 kb pf memory. As shown in Figure 4.13a the maximum total time occurred at node 54 with 725.4 ms. Here the average total time is about 282.6 ms. Likewise Figure 4.13b shows the time taken to compute the descriptor extraction, keypoint detection, occupancy grid, signature creation. More details about the time to compute the feature, total time to generate the graph nodes are discussed in Table 4.2 and 4.3.

(a) Scenario Image

(b) 3D Octomap



(c) Graph

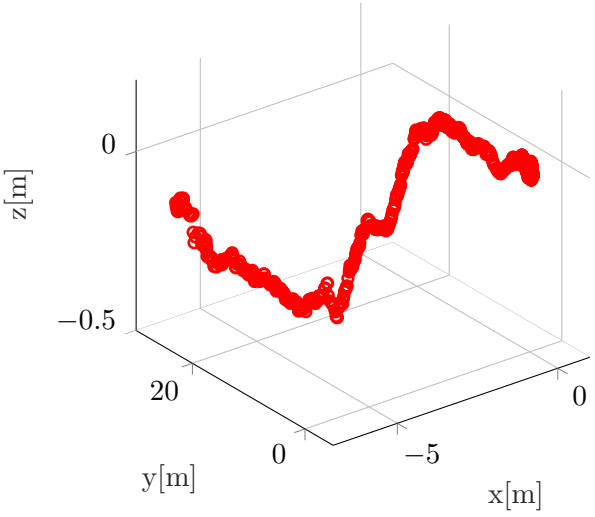


FIGURE 4.12: Image showing the scenario and its equivalent 3D Octomap with Stereo Visual Odometry.

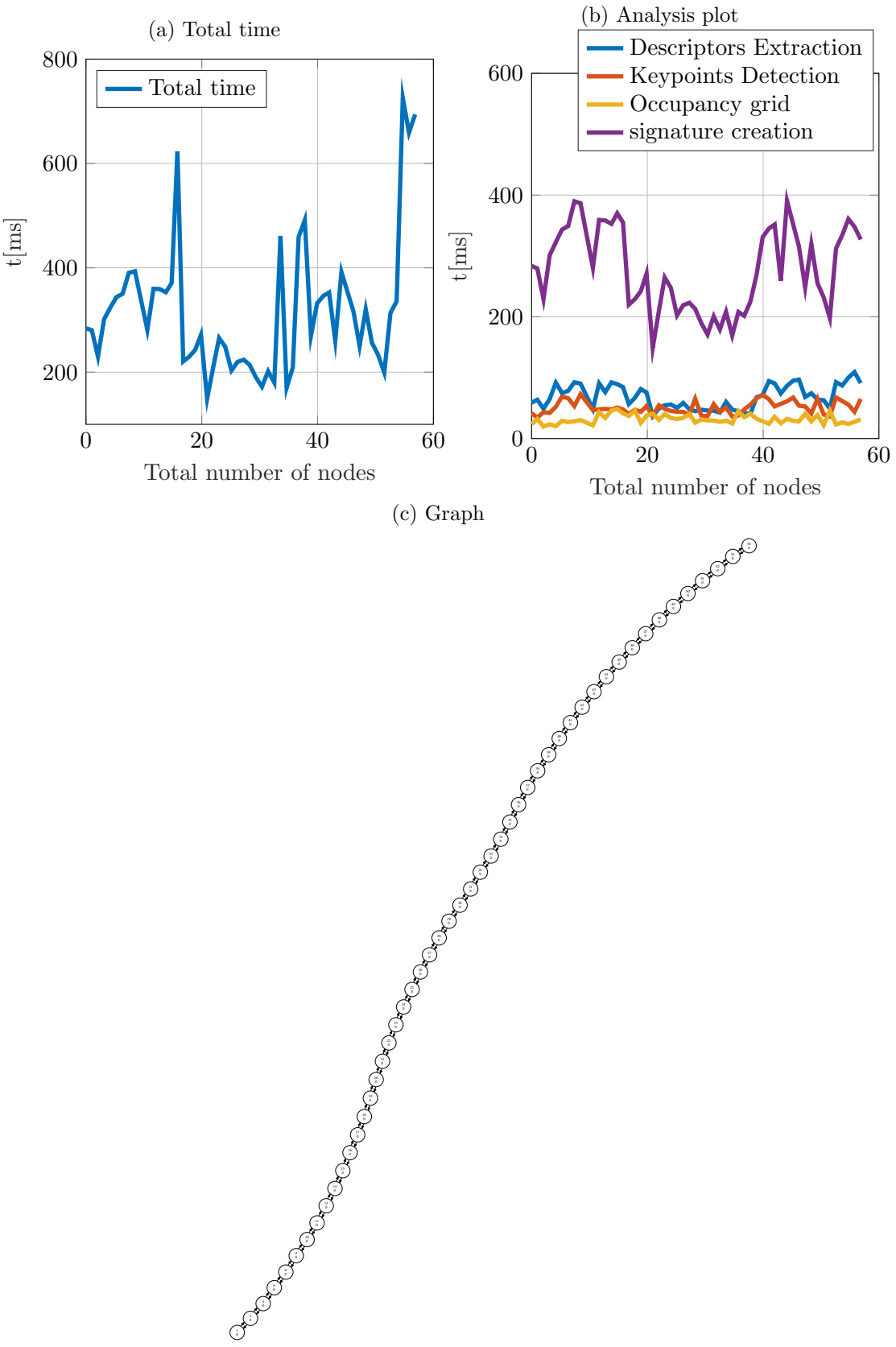


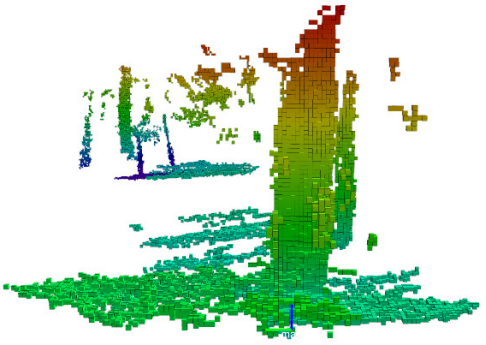
FIGURE 4.13: Plot showing the total time taken to compute, keypoint descriptor, keypoint detection, occupancy grid, signature vs total number of nodes generated and the Graph shows the nodes and edges.

4.4.5 Dataset 3 (Cluttered Environment): Map and graph

This experiment is conducted in a cluttered environment. In the beginning when there is no considerable movement of the stereo camera, the graph generates lot of nodes from 1 to 18 as shown in Figure 4.15c. In this data lot of speckles are observed due to sporadic points of the leaves in the environment. As shown in Figure 4.14a the environment contains large trees and small trees. Figure 4.14b and 4.14c shows the octomap and Stereo Visual Odometry. The small speckles are from the unstructured branches and leaves of the trees. The total distance traveled for this experiment is about 56.1779 m.

(a) Scenario Image

(b) 3D Octomap



(c) Stereo Visual Odometry

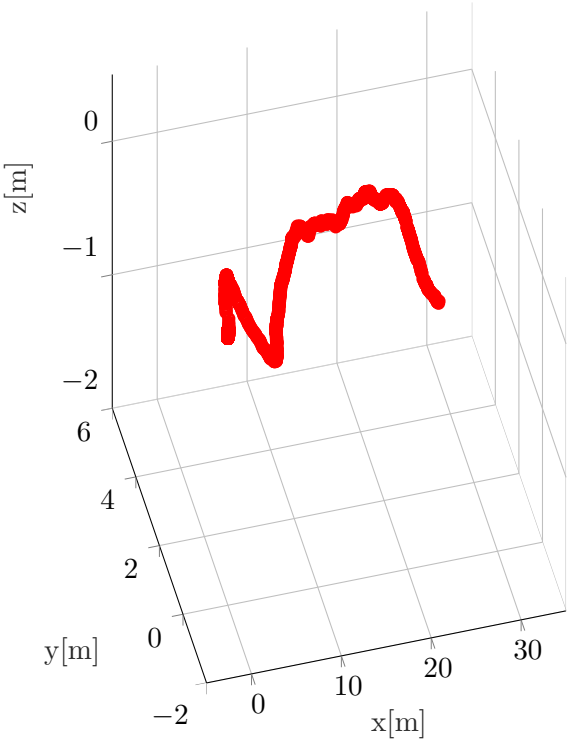


FIGURE 4.14: Image showing the scenario and its equivalent 3D Octomap with Stereo Visual Odometry.

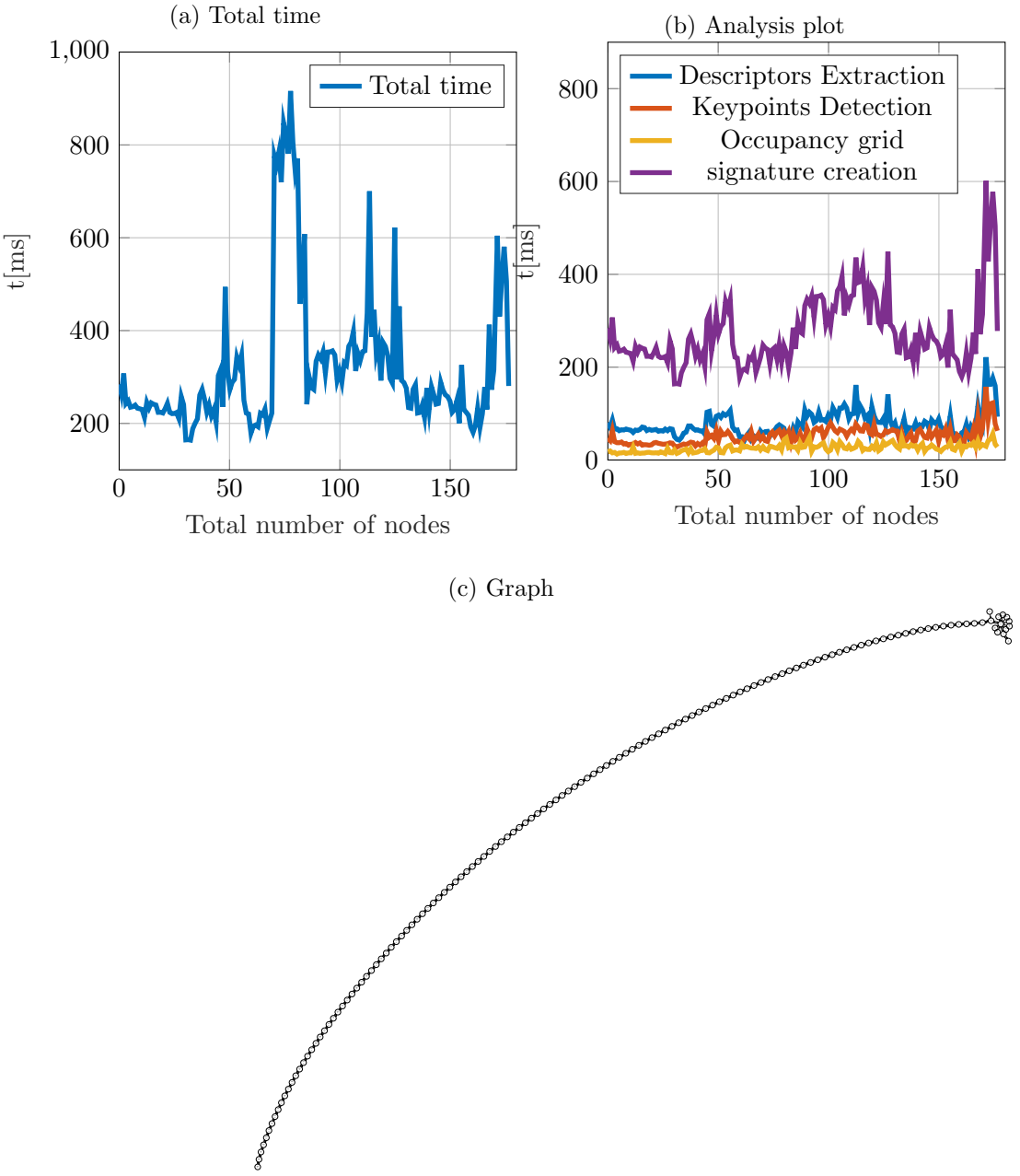


FIGURE 4.15: Plot showing the total time taken to compute, keypoint descriptor, keypoint detection, occupancy grid, signature vs total number of nodes generated and the Graph shows the nodes and edges.

The Plot shown in Figure 4.15a gives the total time to generate the nodes. Here the maximum time is taken at node number 78 with 916.138 ms. Likewise the Figure 4.15b gives the details of time taken for descriptor extraction, key point detection, occupancy grid and signature creation. The total size of the features took 4208.640 kb of memory. More details about the time to compute the feature, total time to generate the graph and visual words are discussed in Table 4.2 and 4.3

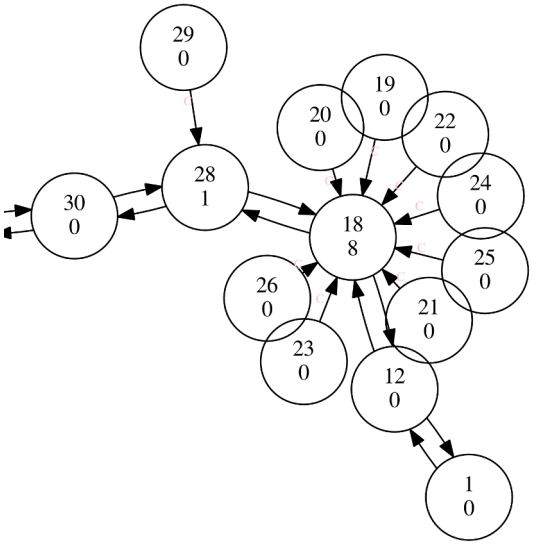


FIGURE 4.16: Graph nodes formed in the beginning.

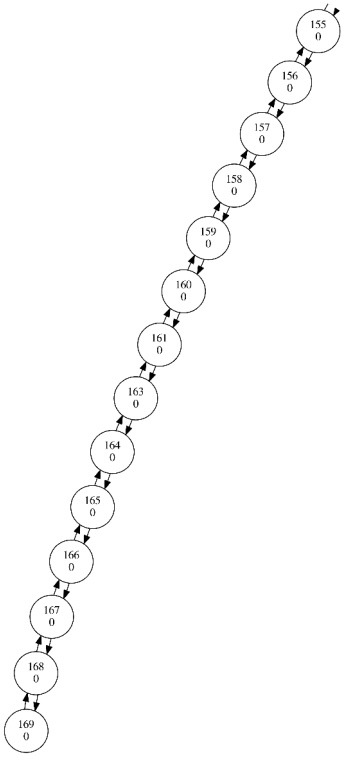


FIGURE 4.17: Graph nodes formed at the end.

4.4.6 Dataset 4 (Moving Object): Map and graph

This experiment is conducted in an environment where moving objects are introduced twice during the process. We covered a total distance of 38.950 m. The generated graph contains 214 nodes processed. 7884 visual words are taken to find the loop closure detection. As shown in Figure 4.19a initially there are only trees (static objects) where we received the map at time 12 ms. Later at time 26.946 ms the moving object (human) is introduced, which is captured by the map like in Figure 4.19c. Then the moving object got updated at time 36.044 ms. Totally it took 6 ms to update the map which is shown in Figure 4.19f.

After the camera position is changed with new viewing angle, a new moving object (human) entered in the frame to check the update frequency of the map. Figure 4.20b, 4.20e shows the map with movements of the person captured at time 93.256 ms. This got updated like in Figure 4.20f at time 128.03 ms. Now it took 11 ms to completely clear the cells to update. Again we tested the moving object while moving along the trees to observe the map update. On an average the moving object is about 8 to 11 ms.

(a) Human as moving object

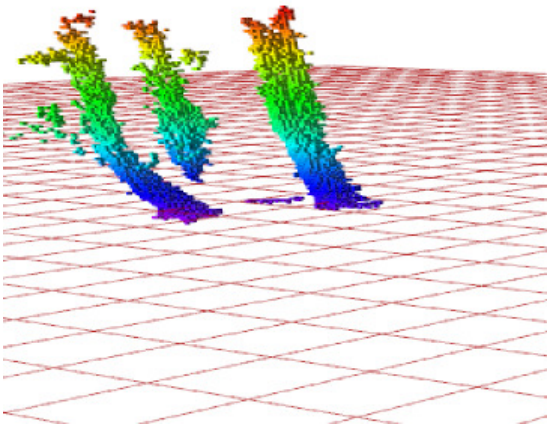
(b) Second Human as moving object

Totally 4670 left and 4670 right images are processed with a frame rate of 30 frames per second. Figure 4.22c shows the structure of the graph where the cluster of nodes are formed in the beginning of the process. In this experiment totally 214 nodes are formed. From node number 52 to 69 we can see the loop pattern which denotes that these nodes are visited again. Since the information comes from nearby images, it is not considered as loop closure. The loop closure is detected between the nodes 45 and 96, 47 and 122, 46 and 131, 46 and 129 respectively. Here the structure of the trees can be seen from the picture showing the scenario in Figure 4.21a. More details about the

maximum and minimum time to compute the feature, total time to generate the graph and visual words are discussed in Table 4.5.

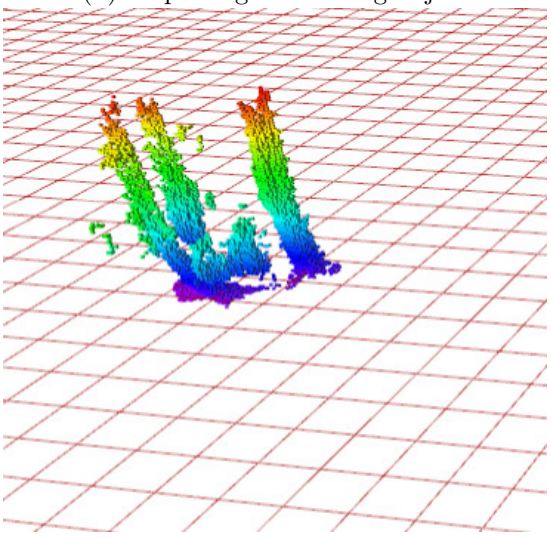
(a) At starting time

(b) Map generated at time 12 ms



(c) Moving Object entering into frame

(d) Map recognize moving object



(e) Moving Object update

(f) Map with Moving Object updated

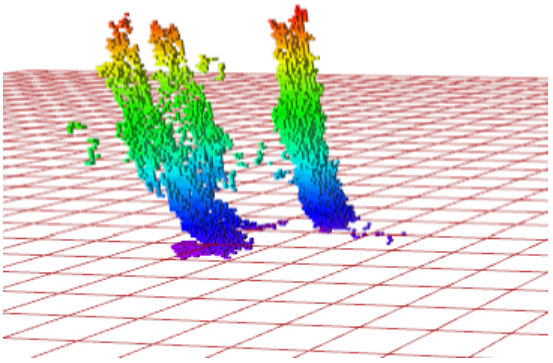
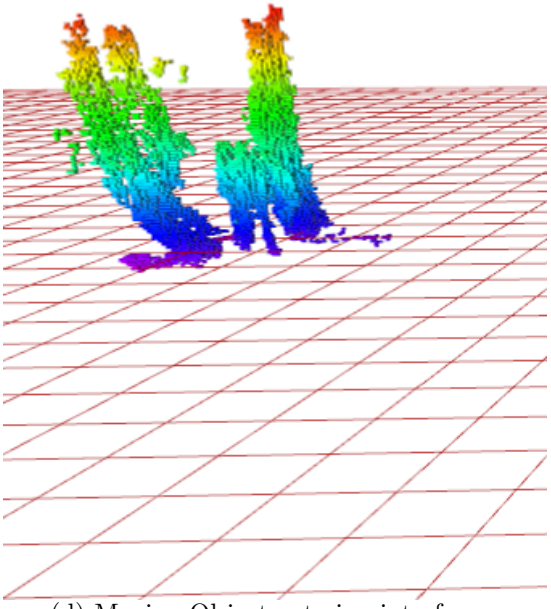


FIGURE 4.19: Octomap capturing the moving object and update over the time.

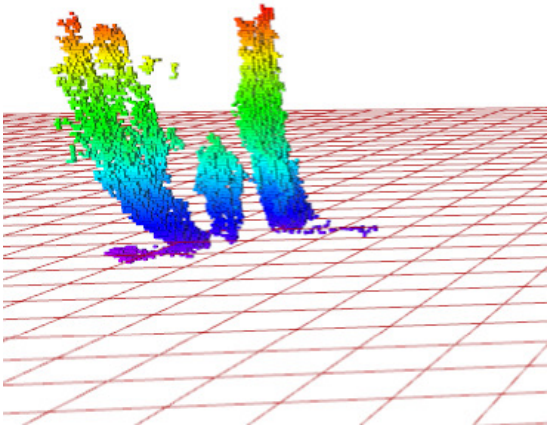
(b) Map generated at time

(a) Moving Object entering into frame



(d) Moving Object entering into frame

(c) Map generated at time



(e) Moving Object update

(f) Moving Object update

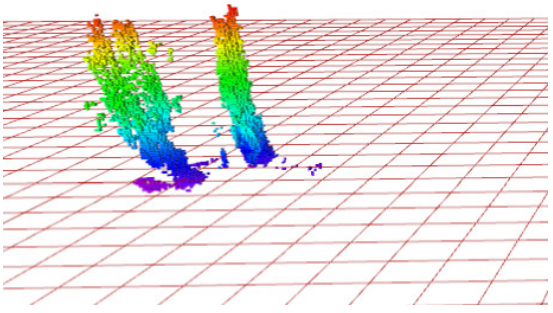
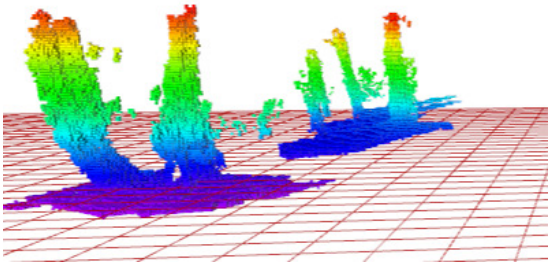


FIGURE 4.20: Octomap capturing the moving object and update over the time.

(a) Scenario Image

(b) 3D Octomap



(c) Stereo Visual Odometry

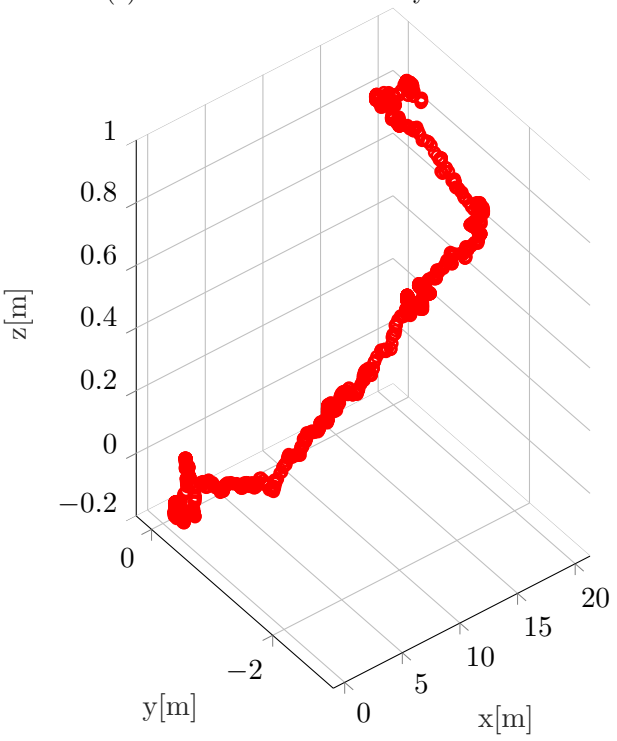


FIGURE 4.21: Image showing the scenario and its equivalent 3D Octomap with Stereo Visual Odometry.

The Figure 4.21a shows the trees with the illumination condition. As we can see in Figure 4.21b the octomap shows the structure of the different trees located near to each other. Especially one of the tree is not vertical when compared to the other trees. The same can be seen in the octomap.

The Figure 4.22a gives the total time taken by the algorithm to compute the octomap, stereo odometry and feature extraction as well as loop closure detection. The maximum time is taken by the node number 222 with 1121.86 ms, whereas the signature also takes

the maximum time of 1113.85 ms in the same node. Here the occupancy grid takes an average of 39.08 ms.

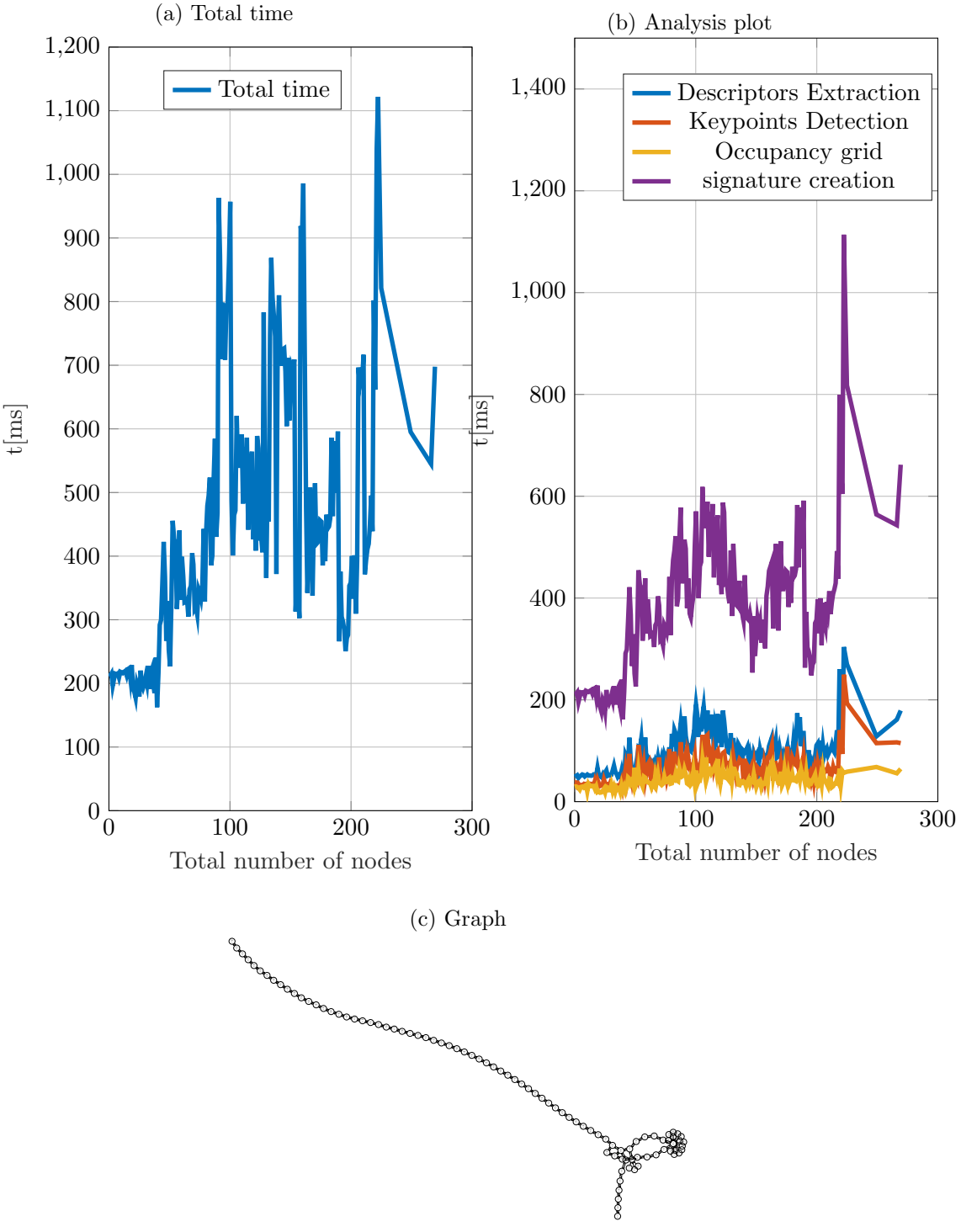


FIGURE 4.22: Plot showing the total time taken to compute, keypoint descriptor, keypoint detection, occupancy grid, signature vs total number of nodes generated and the Graph shows the nodes and edges.

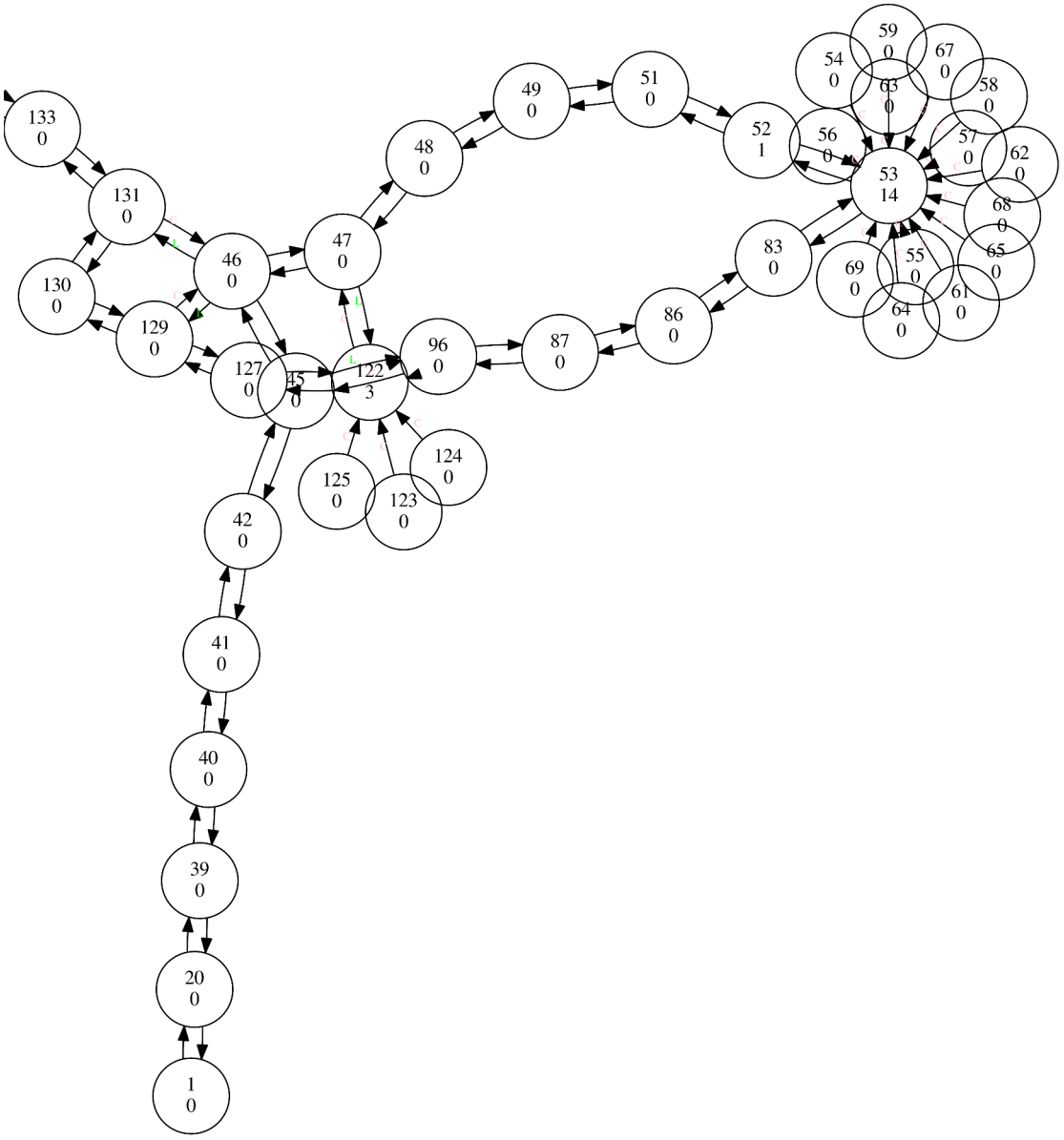


FIGURE 4.23: Graph nodes formed in the beginning.

4.5 Comparison of the results

The experimental results of all the 4 datasets are compared and presented in the Table 4.2 and 4.3 here. The major difference of computation can be noticed in dataset 1 when compared to the other experiments conducted. Besides that the total time in dataset 4 shows that the total time is more than the dataset 2 and 3.

Data	**Total time**(ms)		**Signature**(ms)		**Occupancy grid**(ms)	
	max	min	max	min	max	min
Dataset 1	2148	547.667	2144	499.5	397.4	67.17
Dataset 2	725.4	149.24	390.5	148.9	48.88	19.62
Dataset 3	916.1	164	601.5	163.5	56.55	12.85
Dataset 4	1122	162.5	1114	61.9	87.07	17.68

TABLE 4.2: Comparison of the experimental results obtained from 4 datasets in terms of Total time, Time to create the signature, Occupancygrid.

Data	**Keypoint detection**(ms)		**Descriptor extraction**(ms)	
	max	min	max	min
Dataset 1	653.1	118.2	962	179.8
Dataset 2	73.07	34.46	109.2	36.99
Dataset 3	157.6	27.63	221.5	39.2
Dataset 4	249.6	26.99	303.9	43.45

TABLE 4.3: Comparison of the results obtained from 4 datasets in terms of keypoint detection and descriptor extraction.

As we can see in the Table 4.2 the open environment with distinguished objects takes maximum time when compared to the other 3 experiments. While in cluttered environment with moving object took the minimum time with 214 nodes processed. But in cluttered environment the speckles are caused by unstructured leaves and its movements initiated by the wind.

Again signature creation needed for the loop closure detection in dataset 1 takes more time than other experiments conducted. For the occupancy grid dataset 2, 3, 4 takes an average of 186.22 ms, 30.66 ms, 25.27 ms and 39.08 ms respectively. The graph SLAM generates a maximum of 214 nodes in dataset 4 while the least number is in the dataset 2 with 54 nodes. For feature extraction and detection the lighting condition in the outdoor influences the algorithm. As we can see from Table 4.3 the descriptor extractor takes maximum time of 962 ms for the dataset 1. From the comparison the solution we proposed, works on the sparse as well as in the open environment. While at the same time the environment condition requires some field tuning to have a precise map and

odometry. For example the exposure setting of the stereo camera needs to be tuned according to the lighting condition.

4.6 Concluding Remarks

Overall the three dimensional mapping and localization using graph based SLAM is described in detail here. Our approach consists of 3D mapping with online loop closure detection. Using our approach several outdoor experiments are conducted for initial analysis and inference are summarized in detail. We inferred that the scenario with more textures provides precise localization and mapping. The 3D octomap with speckles are always due to the clear textures even in longer distances.

Through the experiments we demonstrated that the graph SLAM can be incorporated with the octomap with memory efficient in real time. Four dataset presented and the results are compared at the end. In those obtained experiment results the proposed approach can find the potential obstacles and free space in the outdoor environment. Next to that, the run time performance our approach interms of memory and total time to compute the 3D octomap also shown in results.

Chapter 5

Path planning

This chapter describes the proposed path planning algorithm and its implementation in the small UAV. We used the path planner to calculate the collision free path among static as well as dynamic obstacles estimated from the real time mapping (Octomap). Here we explained the general D* lite and its modifications made in order to suit the unknown 3D dynamic environments. Besides that, the expansion of the D* lite to 3D space and its optimal path planning in real time is presented here. Specifically the discussion of the planning and re-planning cost with respect to the cost of the 3D map is addressed. Evaluation and analysis is done through the initial experiments conducted in outdoor and its results are presented here.

5.1 Path Planning

An autonomous small UAV operating in a known or unknown environment requires a safe collision free plan from the initial point to the goal point. Moreover, Real time path planning requires map which provides the details of the obstacles so that it can avoid the obstacles that exist in the environment. Based on the map details the starting point and goal point are given to the path planing algorithm. However, the tasks depends on factors such as environment, shape of the obstacles, position and arrangement of the objects which creates the amount of complexity. Next the idea of path planning is defined [77] immediately,

Definition 5.1. (Path Planning) Assume a function $P : [0, T] \rightarrow \mathbb{R}^3$ where $P[0] = (x_{init}, y_{init}, z_{init})$ and $P[T] = (x_{goal}, y_{goal}, z_{goal})$. Between $P[0]$ and $P[T]$, if there is continuous process χ with the condition $P(\gamma) \in w_{free}$ where $\gamma \in [0, T]$, then the process χ is called path planning.

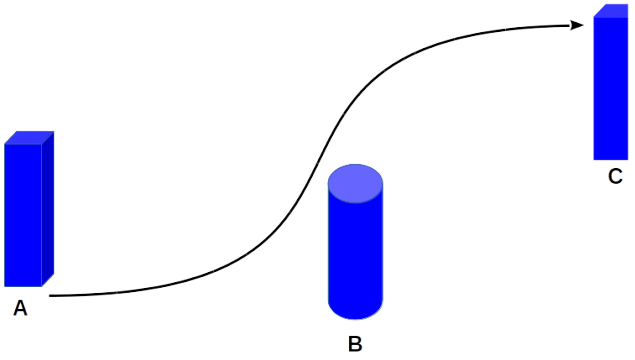


FIGURE 5.1: Static path planning with known environment

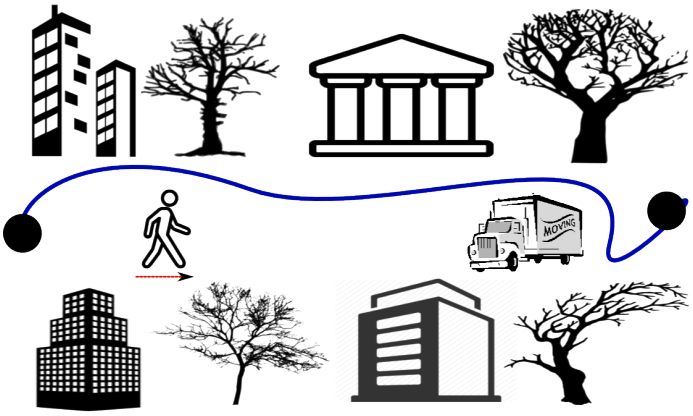


FIGURE 5.2: Path planning in one of the outdoor scenario

For known environment the workspace configuration is already given to the UAV system. With this information of free space and obstacle position, the collision free path can be calculated. For example Figure 5.1 shows the generated path from A to C by avoiding the obstacle B with as assumption that the positions of A, B, C are given already. The mission is to reach the point C from the point A while at the same time, it has to avoid the point B position. But practical scenario consists of static objects such as buildings, trees and dynamic objects like human, automotive vehicle as shown in Figure 5.2. In unknown environment more realistic approach will be creating the path from the initial point to goal point and then change it according to the update of the obstacles with respect to the current position. We focus on generating the collision free path in an unknown environment. Generating the collision free path on-line in real time is one of the complex tasks for the autonomous navigation. The complexity gets higher especially if there are any dynamic environments along with the uncertainties associated with it.

5.1.1 Motivation

To navigate a small UAV in an unknown environment where both the static and dynamic objects are present, we need a dynamic path planning and re-planning. Since the sensing radius of the vehicle is about 7 to 10 meters, the map can be generated only to that limit. So initially a complete path can be planned with an assumption that the unknown space is obstacle free. Once the map of the unknown space got updated with any obstacle in it, the plan has to change accordingly. The change should adapt the previous plan instead of generating a completely new path. Since the small UAV keeps changing its position, we need a collision free path dynamically from its current position. This basically requires frequent updates of the map and faster search technique for the planning in real time.

5.1.2 Problem formulation

We store the map information in the form of a 3D grid as shown in Figure 5.3. The color red, green, blue represents the x, y and z axis respectively. We use this three dimensional grid to search the free and occupied space to plan the collision free path. Each grid consists of set of cells that has a cost value in it, as depicted in Figure 5.4. In this Figure 5.4 the values ranging from 0.1 to 1 denotes the cost value of the cell. These cost values will change according to the update of the map. So we need a search algorithm that efficiently avoids the untraversable cells and finds the optimum traversable cell in real time.

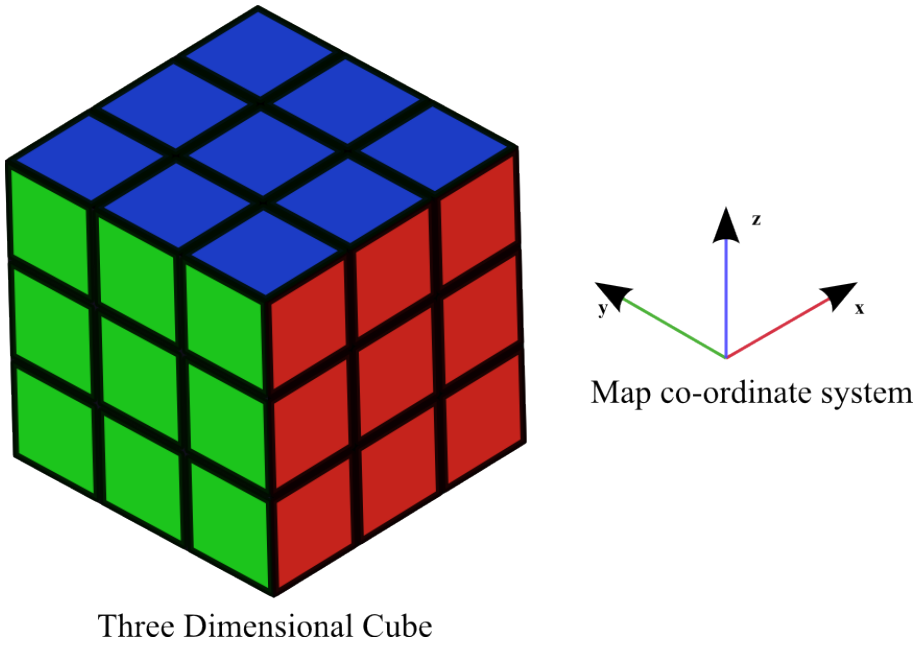


FIGURE 5.3: Three dimensional grid with map coordinate system

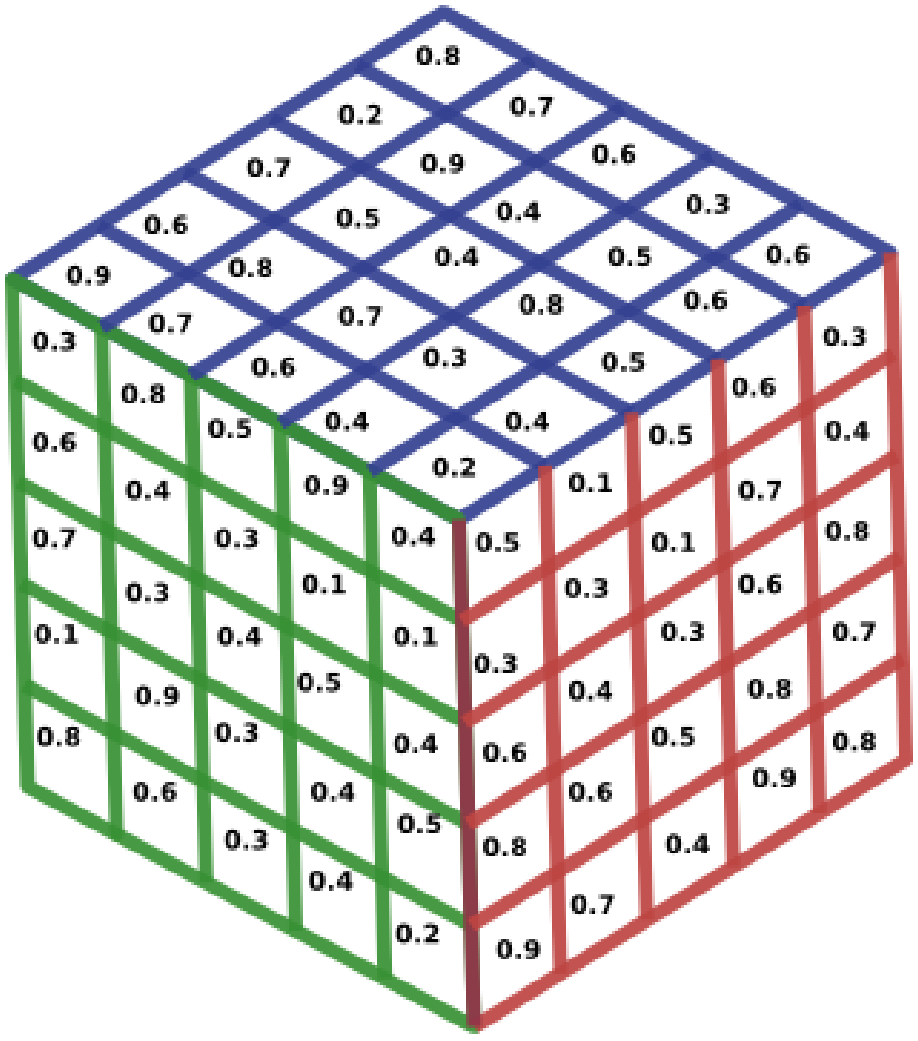


FIGURE 5.4: Three dimensional grid with values from minimum value of 0.1 to maximum of 1

5.1.3 Challenges

The challenge for a dynamic real time path planning in an unknown environment is to have fast and robust re-planning. In our case the vehicle position keeps changing, so the initial path will not be valid after the position changes and we always need re-planning from the current position. But this re-planning should not be far away from the previous plan to ensure smoothness and practical approach. To implement in real time here, the primary input is the occupancy grid. So the path planning module completely depends in the 3D occupancy. Synchronizing both at the same time in unknown environment is a challenging task. The frequent update of the map and fast re-planning process in onboard computer[B] mounted on the small UAV are challenges that need to be accomplished. For instance the signal bandwidth of the onboard computer[B] is connected via wireless with external computer in ground to launch the algorithm.

While performing the processing in real time this signal gets lost from time to time which forms the complexity in monitoring the parameters for verification.

The Section 5.2 explains the general path replanning in a three dimensional grid.

5.2 Path Re-Planning(PRP)

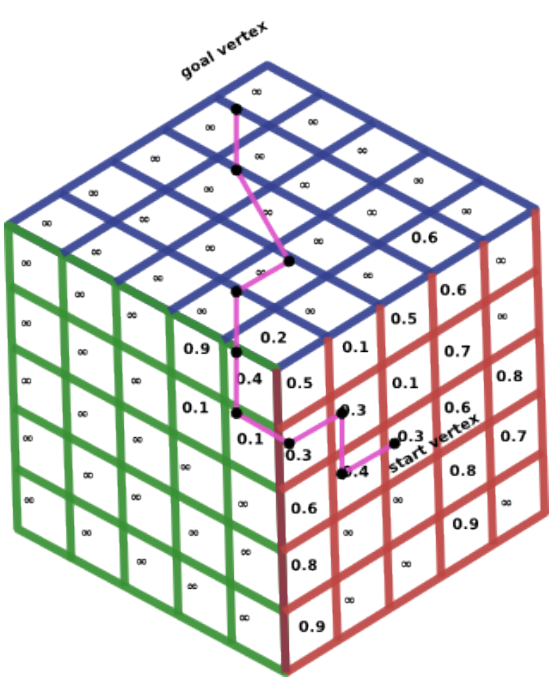
In this section we explained the path re-planning with an example. In general Figure 5.5a shows the 3D grid with cost value C_v of the each cell. The unknown cells are represented as ∞.

Here the conditions for the occupied and free space of the cell is defined as,

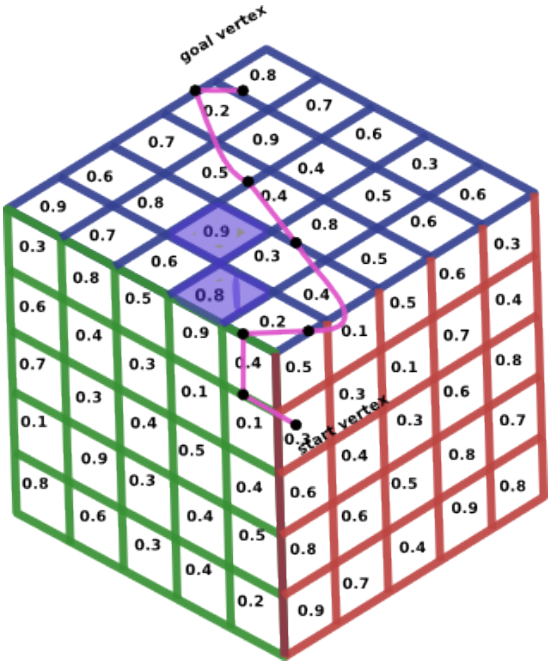
$$\begin{aligned} O_c &= C_v > 0.5 \\ f_c &= C_v < 0.5. \end{aligned} \tag{5.1}$$

Since the whole grid is not expanded at the initial point which is also expensive, the search proceeds as the graph is expanding its successors. This is done by tracking the edges of the cell where the change in cost occurs. So the previous plan should be adjusted according to the cost update of the cell as shown in Figure 5.5a. In this Figure 5.5a a collision free path is planned from the start vertex of value 0.3 to the goal vertex of value 0.2. In the next time step the cost value is updated in z axis which is highlighted in light blue color with value of 0.8 and 0.9. So these cells are not traversable. Due to the cost update, initial path plan as well as the start vertex are adjusted as shown in Figure 5.5b. Again the cells are updating and the previous plan is adjusted to the cells where the values of the cell is below 0.5. In Figure 5.5d the start vertex is very near to goal vertex but still the planner searches for the traversable to reach the goal vertex.

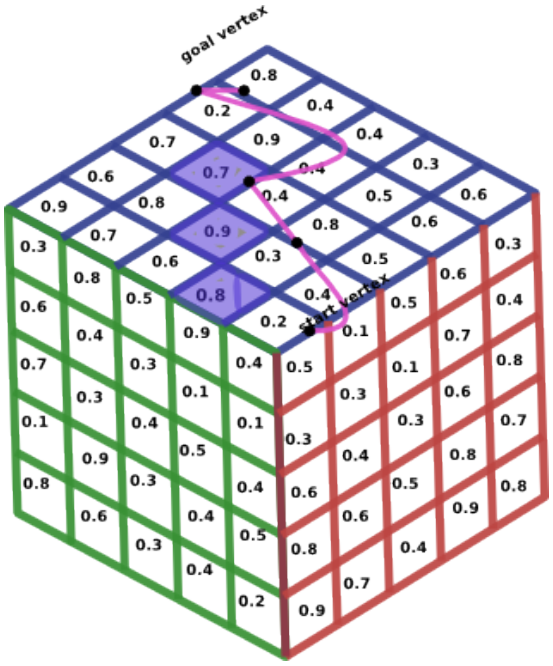
This approach can handle both the static and dynamic environment. As long as the moving object is mapped in the grid the planner will not consider those cells as traversable. For example if the object is moving at an average of 1.4 m/s and it appears in the generated collision free path, the re-planner can handle such situation because our stereo camera setup can map the objects at a range of 8 m to 10 m.



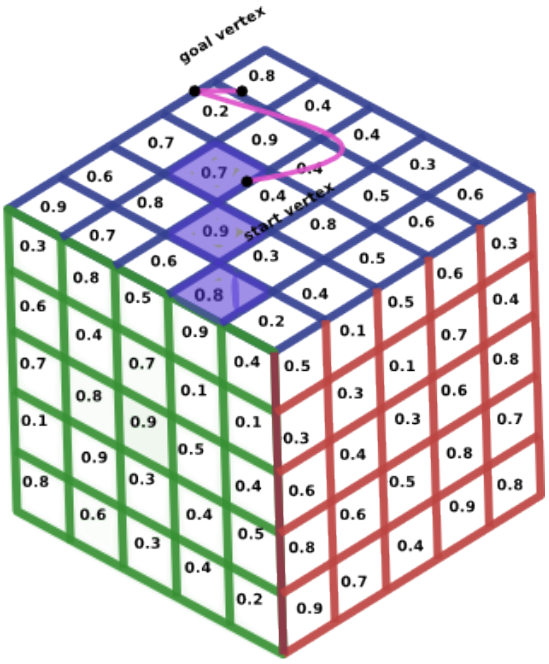
(a) Initial plan at time t=0



(b) After the first move the start vertex has changed



(c) After the update of the map and current position keeps changing



(d) After new obstacle is updated and change in the plan

FIGURE 5.5: Path planning and its updated plan in the unknown environment when the cost of the cells are updating over the time

5.3 Generic D* Lite

D* Lite is a graph based path planning algorithm. The D* Lite principle is based on incremental heuristic search method [101] which continuously finds the shortest distance between the current vertex to the goal vertex. According to the cost change in the transition between vertex, it will calculate the path. It can work without any prior information for the search. This will perfectly suit in our case. Because the small UAV moves along the current vertex i.e current position always changes with respect to the goal vertex . Instead of calculating the shortest path from the initial point, it uses the previous information of the search as well as considers the start vertex that are only relevant to calculate the shortest path towards the goal vertex. With the update of the start vertex and distance to the goal that are relevant for re-planning, the shortest path will be estimated. The detailed procedure of the D* Lite algorithm is explained in [85].

Given set of vertices V, the function Succ(v) will give the set of vertices V that are connected to each vertex v and $v \in V$. Transition from one vertex v to the next connected vertex v' is given by the cost function $c(v, v')$. Here the function Pred(v) represents the predecessors of the vertex v. It maintains the cost of the path $g(s)$ as well as rhs(v). Here rhs(v) is also a cost estimate with a step ahead of $g(v)$ like shown below,

$$\text{rhs}(v) = \begin{cases} 0 & \text{if } v = v_{start} \\ \min_{v'} \in \text{Pred}(v)(g(v') + c(v, v')) & \text{otherwise} \end{cases} \tag{5.2}$$

If two of the cost estimates for vertices v are equal $g(v) = \text{rhs}(v)$ then v is consistent. On contrary if $g(v) \neq \text{rhs}(v)$ then it is locally inconsistent. Because the cost of the vertex is not optimal towards the goal, search will be reversed from goal vertex to start vertex.

Here the heuristic (v, v') is used on the search. Condition for the heuristic is that it should satisfy the conditions,

$$h(v_{goal}, v'_{goal}) = 0 \tag{5.3}$$

$$h(v, v_{end}) \leqq c(v, v') + h(v', v_{goal}) \text{ for all vertices } v \in V \text{and } v' \in \text{Succ}(v) \tag{5.4}$$

The inconsistent vertices are stored in the priority queue Q. All the vertices in the priority queue will be categorized by their key values like shown below,

$$\begin{aligned} k_1(v) &= \min(g(v),\ \text{rhs}(v)) + h(v, v_{goal}) \\ k_2(v) &= \min(g(v),\ \text{rhs}(v)) \end{aligned} \tag{5.5}$$

During the search process the vertices are processed according to the priority queue. For the example vertex v started before v', if $k_1(v) < k_1(v')$ or $k_1(v) = k_1(v')$ then it will take order the priority queue for the search.

$$g(v) > \text{rhs}(v) \tag{5.6}$$

$$g(v) < \text{rhs}(v) \tag{5.7}$$

Equation 5.6 and 5.7 denotes the over consistent and under consistent vertex. In the case of overconsistent vertex, it can be changed to consistent vertex by making $g(v) = \text{rhs}(v)$. So all the predecessors of the rhs values will be recomputed

For under consistent vertex v, the $g(v)$ will be set to ∞ so that in the priority queue it will be as overconsistent vertex. Again the rhs values are recomputed. Here if start vertex is consistent and no vertex in priority queue is less than the start vertex, we can get a optimal path. This implies the process should be continued until the start vertex is consistent.

In our case while sensing the small UAV moving which obviously changes the start vertex and the map update changes the cost of the vertex. So rhs values will recompute to attain the consistent start vertex. This will be efficient to plan from the current position to the goal position. At the same time typically it will not give optimal paths for all scenario. For example in 2D environment which is represented in 2D discretized grid, it has only eight connected vertices for a vertex v. So it will have only eight directions to plan a path. This will limit the optimal path planning which results in suboptimal solution.

Expanding to 3D

Here we are using the 3D octomap which is based on occupany grid to store the map information. We define our three dimensional grid to search the collision free path. In this grid the vertices are connected by 26 neighborhood vertices [102]. Inside the grid, cells are not uniform. This non-uniform grid generated with the minimum and maximum boundaries of the grid. This forms the graph structure with vertex calculating the incoming and outgoing edges. The objective is to compute the path between the starting point and goal point.

$$v_{goal} = (v_x, v_y, v_z). \tag{5.8}$$

Algorithm 2 D* Lite - Procedure

1: **procedure** D*LITE
calckey(v)
2: return $[\min(g(v), \text{rhs}(v)) + h(v_{start}, v) + k_m; \min(g(v), \text{rhs}(v))]$;
Initialize()
3: Q = ∅; $k_m = 0$;
4: for all vertices $v \in V$ rhs$(v) = g(v) = \infty$;
6: rhs$(v_{goal}) = 0$;
7: Q.Insert (v_{goal}, calckey(v_{goal}));
Update Vertex(e)
8: if ($g(e) \neq$ rhs(e) AND $e \in Q$ Q.Update(e,calckey(e));
9: else if ($g(e) \neq$ rhs(e) AND $e \notin Q$) Q.Insert(e, calckey(e));
10: else if ($g(e) =$ rhs(e) AND $e \in Q$) Q.Remove(e);
ComputeShortestpath()
11: while (Q.Topkey()< calckey(v_{start}) or rhs($v_{start} > g(v_{start})$);
12: e = Q.Top();
13: k_{old} = Q.Topkey();
14: k_{new} = Calckey(e);
15: if($k_{old<K_{new}}$)
16: Q.Update(e,k_{new});
17: else if ($g(e) >$ rhs(e));
18: $g(e)$ = rhs(e)
19: Q.remove(e);
20: for all $v \in Pred$(e); rhs(v) = $min(rhs$(v),c(v,e)+ g(e));
22: Updatevertex(e);
23: else
24: g_{old} = g(e);
25: $g(e) = \infty$; $forall$v$\in Pred$(e) $\cup$ e
26: if (rhs(e) = c(v,e)+g_{old})
28: if($v \neq$v$_{goal}$ rhs(v) = $\min_{v' \epsilon Suc(v)}()c(v, v') + g(v')$)
29: Update vertex()
Main()
30: $v_{last} = v_{start}$
31: Initialize();
32: Computeshortestpath();
33: while($v_{start} \neq v_{end}$)
34: $v_{start} = arg\min_{v' \in suc(v_{start})} c(v_{start}), v' + g(v')$;
35: Move to v_{start};
36: Scan graph for changed edge costs;
37: if any edge costs changed
38: $k_m = k_m + h(v_{end}, v_{start})$;
39: $v_{last} = v_{start}$
40: $c_{old} = c(e, v)$;
41: Update the edge cost $c(e, v)$;
42: $if(c_{old} > c(e, v))$
43: rhs(e) = min (rhs(e),c(e,v)+$g(v)$);
44: else if (rhs(e) = c_{old} + g(v))
45: if ($e \neq$v$_{goal}$)rhs(e) = $\min_{v' \in suc(e)}(c(e, v') + g(v'))$;
46: Updatevertex(e);
47: Computeshortestpath();

5.4 Modified 3D D*Lite

To achieve the fast and dynamic planning we applied the D* Lite algorithm [85] and expanded it to 3D space while optimizing the extraction process in real time. Because the main objective here is to generate the collision free path with dynamic objects in unknown environment, the obvious requirement is efficient re-planning. D* Lite is taken here to develop the online fast re-planner by doing the re-planning not from the scratch but only where the cost change occurs.

Since the underlying structure is based on the graph, we used the fibonnaci heap extraction process along with lattice graph. Besides that in our implementation it has the ability to remove or add edges at anytime. Because of using the past search information, it saves the computational power as well. Besides the start vertex is directly given by stereo odometry of the small UAV.
The steps of our approach is given below,

Algorithm 3 Collision free path planning process

1: **procedure** ONLINE 3D PATH PLANNING
2: 3D Grid G →l, w, h,*occupancymap*,s$_x$, s_y, s_z, costscale, maxcost
3: Connectivity → 26*cellmooreneighbourhoodconnectivity* *Gridcost* → *Euclideandistance*
4: Cell cost → *occupancymap* *Search* → *ThreedimensionalD* ∗ *Lite*
5: Path → *Sequenceoftraversablecells* –

Where we have l, w, h are the length, width and height of the grid. Similarly s_x, s_y, s_z are the x, y and z scale of each grid cell. A 3D grid G has n number of cells where the size of the cells are unknown. Now we have to find a sequence of cells that can be traversable to reach the mission with minimum distance. This 3D grid connectivity is done by 26 moore neighbourhood connectivity [102]. Here the maximum cost will add the cell if it is less than the maximum cost. The cost of the cell will be given by the octomap. Depends on the amount of occupancy in the defined cell and the maximum cost provides the cost of the each cell. Now the grid cost to move from one cell to the another cell is defined by,

$$Cost = \sqrt{(x - x_1)^2 + (y - y_1)^2 + (z - z_1)^2} + \text{average cell cost} \tag{5.9}$$

5.4.1 Three dimensional grid search

Here the graph structure is directed with vertices and edges which are added during the process. Since it is based on the graph structure it will expand while the search proceeds. The search is initialized with graph and cost interface.

Algorithm 4 Online three dimensional grid search

1: **procedure** THREE DIMENSIONAL GRID SEARCH
2: Get the graph.
3: Get the cost interface of the cells and each cell cost
4: Get the Stereo Visual Odometry for start vertex (x, y, z)
5: Set goal vertex $(v_x(goal), v_y(goal), v_z(goal))$
6: follow the generated path until some changes in the graph are observed
7: ChangeCost( ) – for every changed cost
8: Iterate steps 1-4 for replanning

For the planning we have the input as three dimensional grid and its connectivity. The transition between one cell to the other cell is based on the cost of the cell like in equation 5.9. Plan the path from the start vertex to the goal vertex with an assumption that the unknown cells are traversable. Based on the specific application the cost interface should be provided. Because it does not define a real distance or the heuristic distance between the vertices, implementing the D* lite in 3D requires several optimizations. First of all the graph is restructured to reduce the heap and edge iterations. For faster planning the Fibonacci heap is applied instead of having the priority queue. For complex environments the focused D* is proven to be more effective for heap extraction. The search is initialized with a graph and cost interface between the vertices. The nodes in the graph will be modified by the parameters used in the search process which are then stored in the nodes. The initial plan is made from the start vertex to the goal vertex. This path is valid until the cost change is detected. If there is a cost change then it re-plans. In our case the vertex always expands the predecessors and successors, because this is implemented in unknown environment. This is not the case for known environment where the graph is already built. To plan from the start to the goal point, the successors need to be expanded. So the planning can be extended forward. Otherwise it will plan from the goal point to the start point, if the predecessors are expanded. The edges can be tracked through the cell to keep the list of edges. This will be used to adjust according to the change in the cost. A 3D grid path is calculated by the list of cells that are traversable with the total cost of the path.

5.5 Experiments

To validate the 3D path planning and re-planning algorithm in real time experiments are conducted in the outdoor practical environment. Initial testing were done in Intel processor laptop and later the same is tested in ARM processor (onboard computer[B]) mounted in the small UAV.

Several datasets are collected and three of them are presented here. In every experiment the initial position $(0, 0, 0)$ is considered to be the point where the map and stereo odometry started to generate. When a target point is given the collision free path will be calculated from the current position. Since there is continuous movement of the stereo camera throughout the experiment, the stereo odometry will provide the present position of the small UAV. Initially the occupancy grid is not expanded completely. It will expand its successors and the graph search will be done in the forward direction. Based on the cost update the search will be restructured for the heap extraction. The original collision free path is smoothened by the spline function here. In this section three datasets results are presented taken in outdoor scenario.

5.5.1 Dataset 1: 3D path planning

This dataset is taken in outdoor scenario with two trees, one big iron tank, one iron pole. The path is planned based on the update of the 3D occupancy grid values. Since it is a non-uniform grid, the grid bound values keep changing. In this experiment it ranges from $(100, 120, 100)$ to $(320, 240, 200)$. At each time instance the planner checks the cost of the cell through the graph search and if there is no change the same path remains valid. The goal here is to reach $(20.5, 2.5, 4.5)$ and plan a collision free path. The Path is planned to reach the target from the current position. The three dimensional octomap updates its grid with the known, unknown and occupied cells. Since the trees have uneven branches with leaves, the speckles are observed in the map. The map is built in meter unit and the size of the small UAV is 0.7 m. To avoid the collision, double the size of the UAV dimension would be good enough to navigate. In that case, 1.4 m of space between the object and small UAV is always considered as a valid path. The reason for sporadic map points are that leaves in the trees and grass on the ground are too little and are scattered uneven. Totally 87 paths are planned at different time intervals with a total duration of 76 s. Here the paths which remains valid in the next consecutive steps are not shown. The path which change their starting position compared to the previous one as well with cost change in the previously planned edges are presented here.

Figure 5.6 shows the outdoor environment scenario where this particular experiment is conducted. The arrangement of the two trees and the iron tank can be noticed from this Figure 5.6.

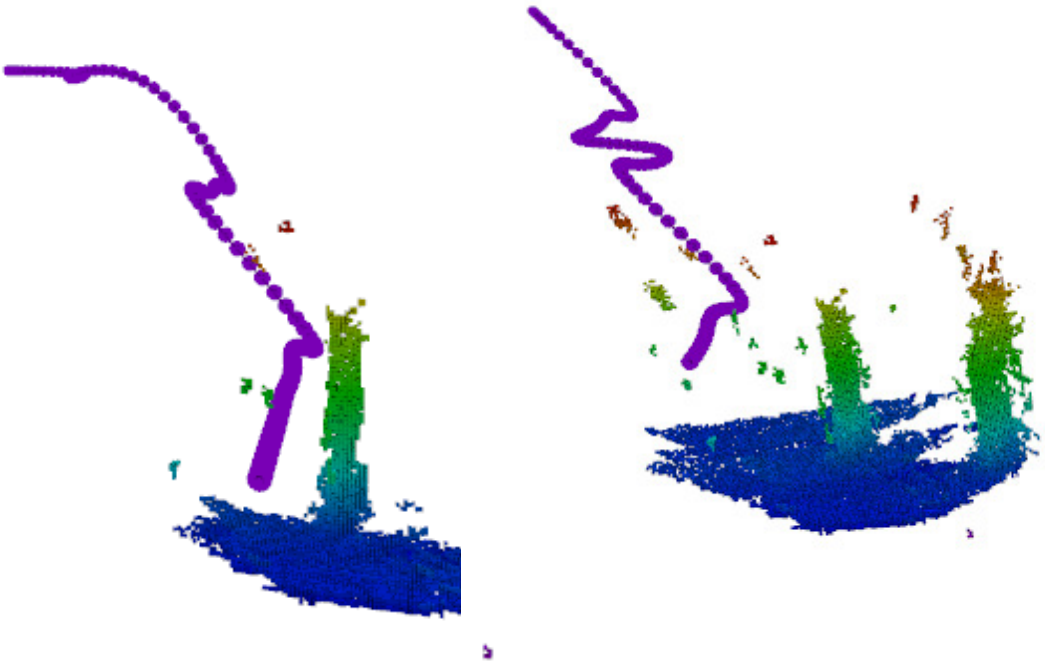
As shown in Figure 5.7a the initial collision free path is planned between the two trees. This path is valid until the edge cost are changed. In Figure 5.7b the starting position and the edge cost of the previous path is changed. So it adapted a new path to the goal point. We can clearly notice the difference between the initial plan and next plan in Figure 5.7a and Figure 5.7b. After that the search updated the edge cost of the previous plan with a new plan that has a curve like form as shown in Figure 5.7c. At the end we can see the complete environment with two trees, one pole and an iron tank. Here the path is planned above the objects as shown in Figure 5.7d.

FIGURE 5.6: Outdoor Scenario with trees and an iron tank

The Figure 5.7 shows the 3D path planned at different time steps. In Figure 5.8a the initial path planned values are shown. After the odometry changes the starting position is adjusted to the present position as shown in Figure 5.8b. Followed by that, in Figure 5.8c the shortest path plan is estimated by changing the direction. The same plan continues with change in the starting position as shown in Figure 5.8d. But in Figure 5.8f the plan again adapted like it is in the second step.

(a) Initial plan

(b) New path from the current position



(c) New path from the current position

(d) Obstacle update and new plan

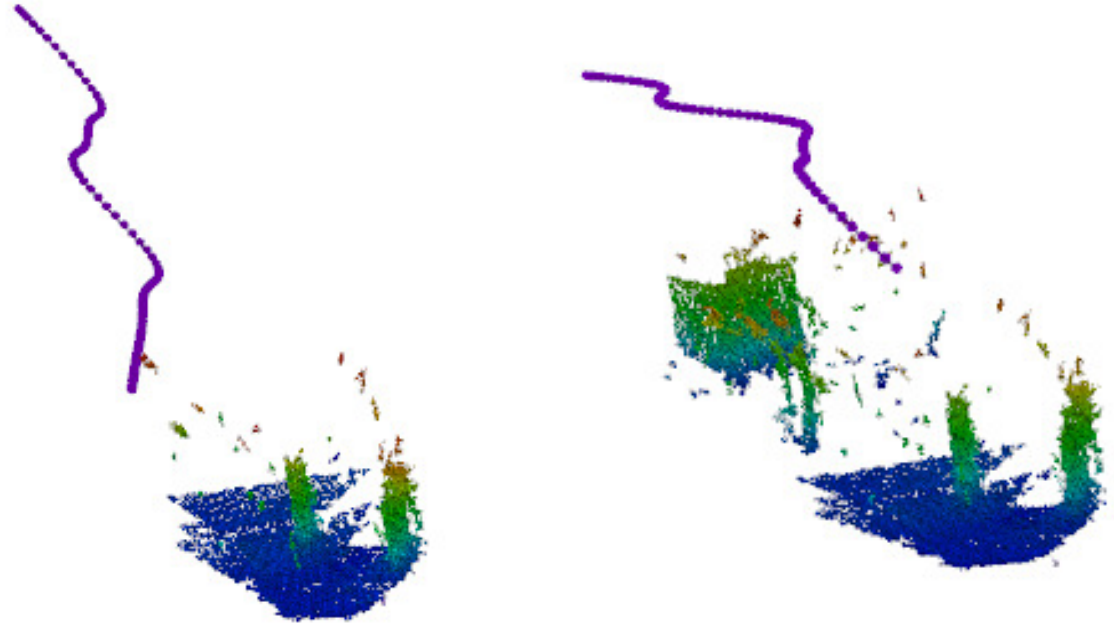


FIGURE 5.7: Three dimensional path planning among trees and iron tank in outdoor environment

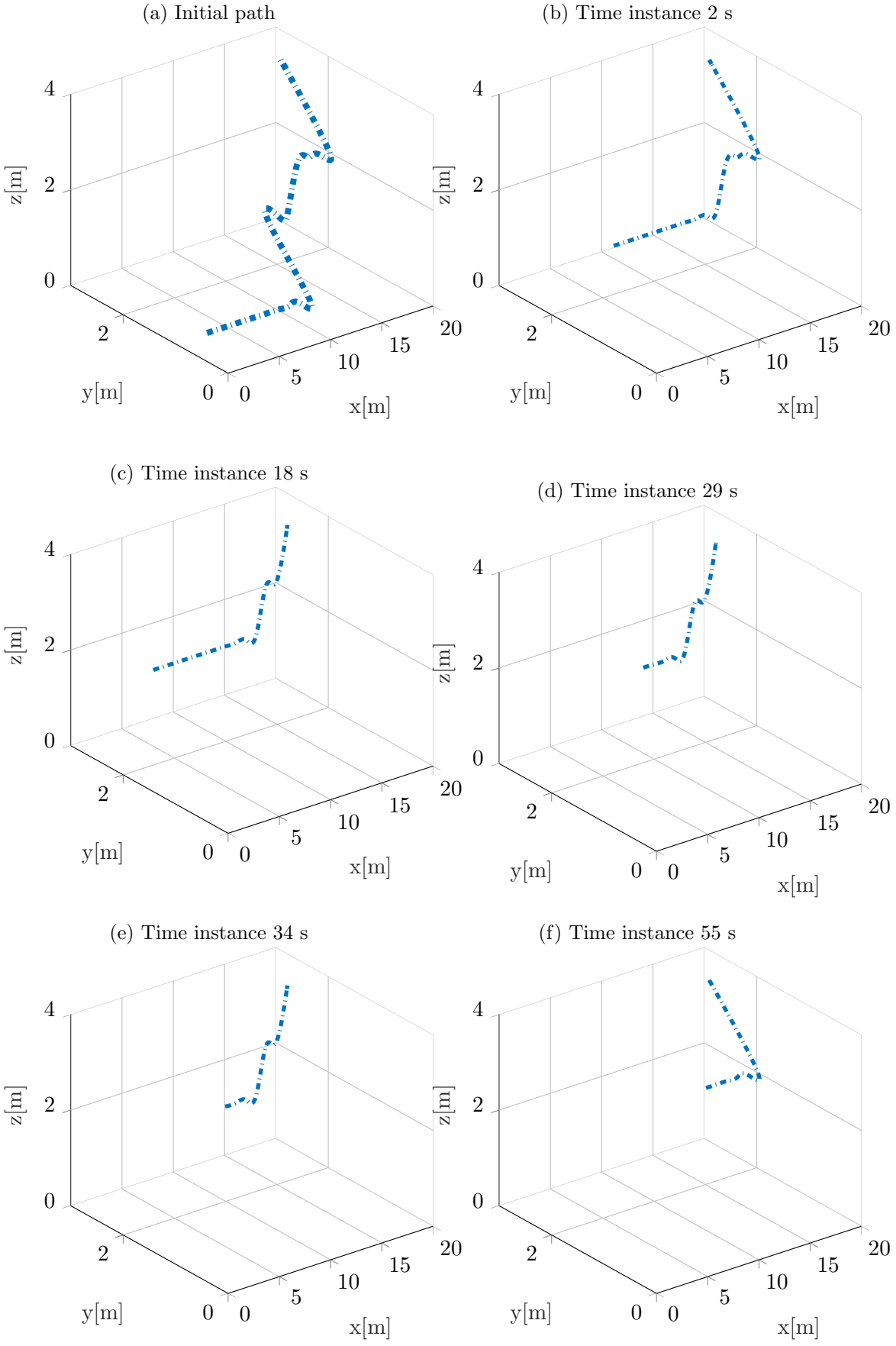


FIGURE 5.8: Three dimensional path planned from at time 1 s to 55 s

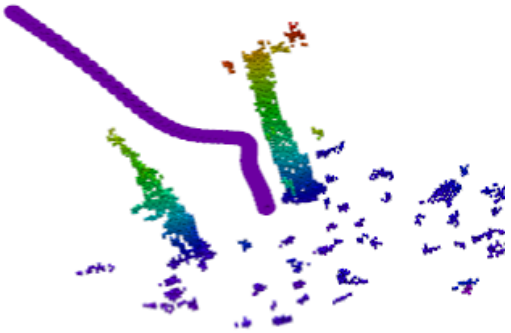
5.5.2 Dataset 2: 3D path planning

This dataset is taken only with four trees to check the collision free planning in the congested confined space. The total duration of this experiment is 1 min 10 s and it produced 27 full octomap messages in ROS. From that 27 messages totally 71 collision free paths are planned in 49.5 s with total memory of 1.1 mb. We can see that the shape of the trees are not even. The changes against the initial plan is observed at each time step provided the new update of the three dimensional grid values. In Figure 5.10a the initial plan is computed based on the cost of the map where only two trees were present, because the cost of the edge at the initial point has position of the first two trees. Now the probability of the hit has increased at the same time when the current position is also changed as shown in Figure 5.10b. In the third Figure 5.10b again the re-planning is computed based on the graph search. In the Figure 5.10d we can see all the four trees. Now the current position is changed again but the goal vertex is the same. We can evidently see all the path planned are reaching the goal vertex. At the same time, it is robustly adapting the previous path relative to the current position. So with the initial information of traversable, the cells are later adapted smoothly during the expansion of the vertex.

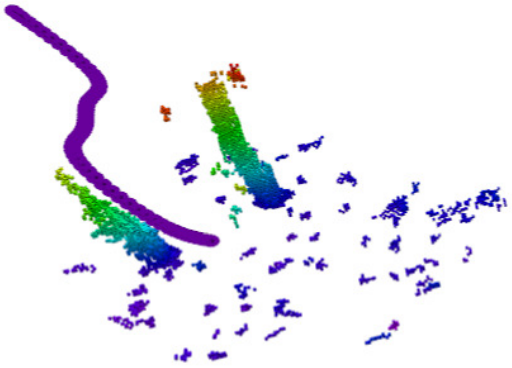
FIGURE 5.9: Outdoor Scenario with four trees.

(a) Initial plan with only two trees at 6 s 8 ms

(b) Replan after change in cost and position 7 s 3 ms



(c) New plan after the introduction of new obstacle 17 s 30 ms



(d) Replanning among four two trees 34 s 12 ms

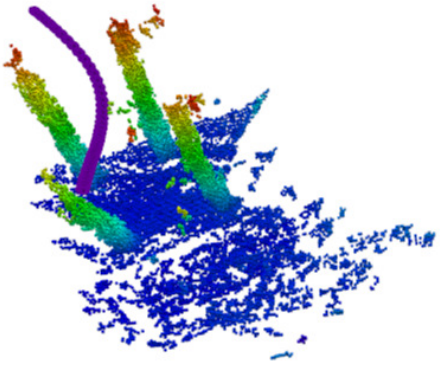


FIGURE 5.10: Path planning and re-planning among the static obstacles

The Figure 5.11 shows the path planned values among the four trees. The Figure 5.11a is the the initial plan from the start vertex to the goal vertex. From Figure 5.11b it started to adjust based on the cost change. For example in Figure 5.11d and 5.11e the difference is only the starting position. While at the same time in Figure 5.11f and 5.11g we can see the starting position as well as the path adjusted to avoid the obstacles.

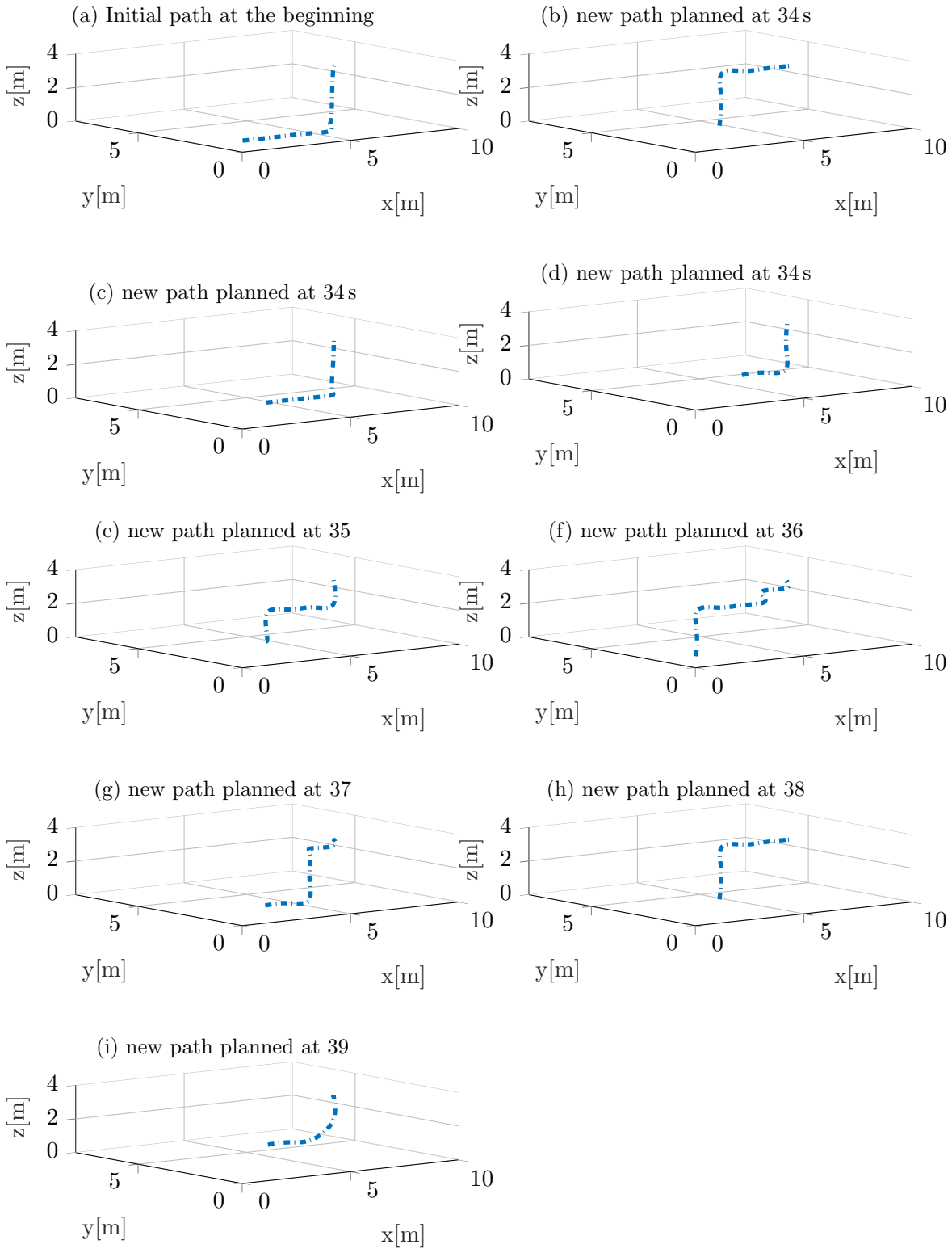


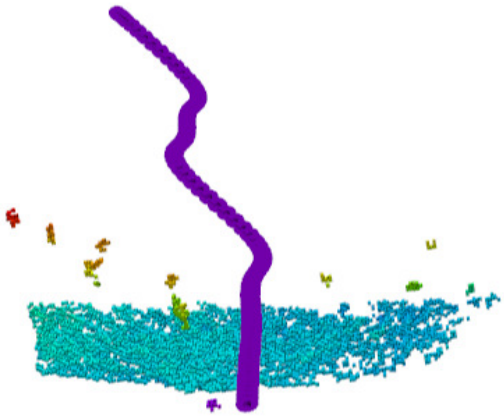
FIGURE 5.11: Experiment among four trees: 3D Path planning values with different starting position and change in edge cost

5.5.3 Dataset 3: 3D path planning

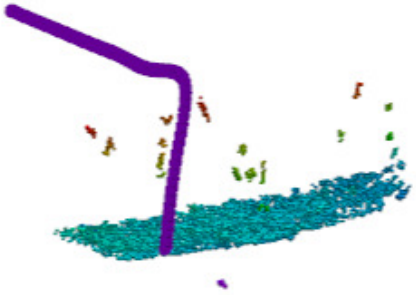
This dataset is taken in front of the big iron tank metal structure. The reason is that to analyze the planning with single obstacle of non-uniform structure on top of it. Here the octomap shows the size and structure of the object. Totally 11 figures are presented here to show the difference in path with respect to the current position as well as to the cost of each grid cell. In addition to that the values of the path at the time instances when the change occurs are presented here. The path that remains same at the consecutive time steps are not shown here. Totally 128 collision free paths are planned with a total duration of 48.3 s. Here the goal point is $(9.5, 2.5, 2.5)$ from the starting point. As shown in Figure 5.13a the planning is done with no obstacles infront initially. In the next step 5.13b previous plan edge costs are updated, so the new plan is adapted to the change. After 8 s 44 ms the change in the altitude i.e in z-axis of the previous path is observed. In Figure 5.13d the starting position is adjusted but the rest of the path is almost the same. On the other hand we can see in Figure 5.13e the edge costs are completely updated and a new plan is immediately planned. The same is continued again as shown in Figure 5.14a, where we can clearly see the map update from the ground. But in the Figure 5.14b, almost the complete front portion of the structure can be observed with little scattered points. Still the search and planner adapts quickly. From Figures 5.14c, 5.14d, 5.14e, 5.14f we can see that the map updates are fast and the planner keeps changing the plan dynamically.

FIGURE 5.12: Outdoor Scenario with big iron structure.

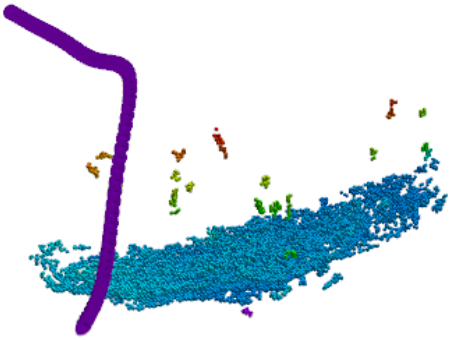
(a) Initial plan



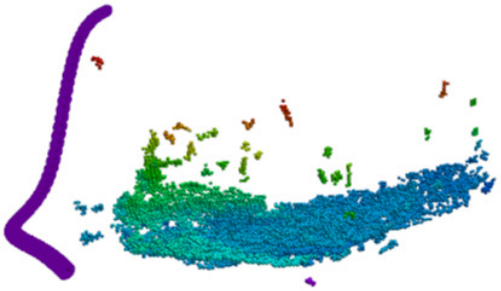
(b) New starting position



(c) Orientation Change



(d) Cost change and new current position



(e) Cost change and shortest path

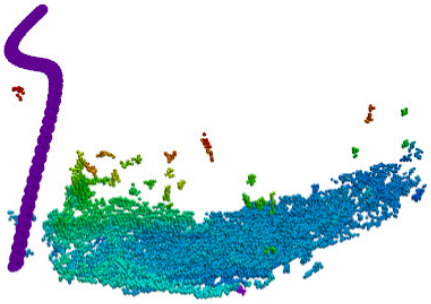


FIGURE 5.13: Three dimensional planning with single obstacle

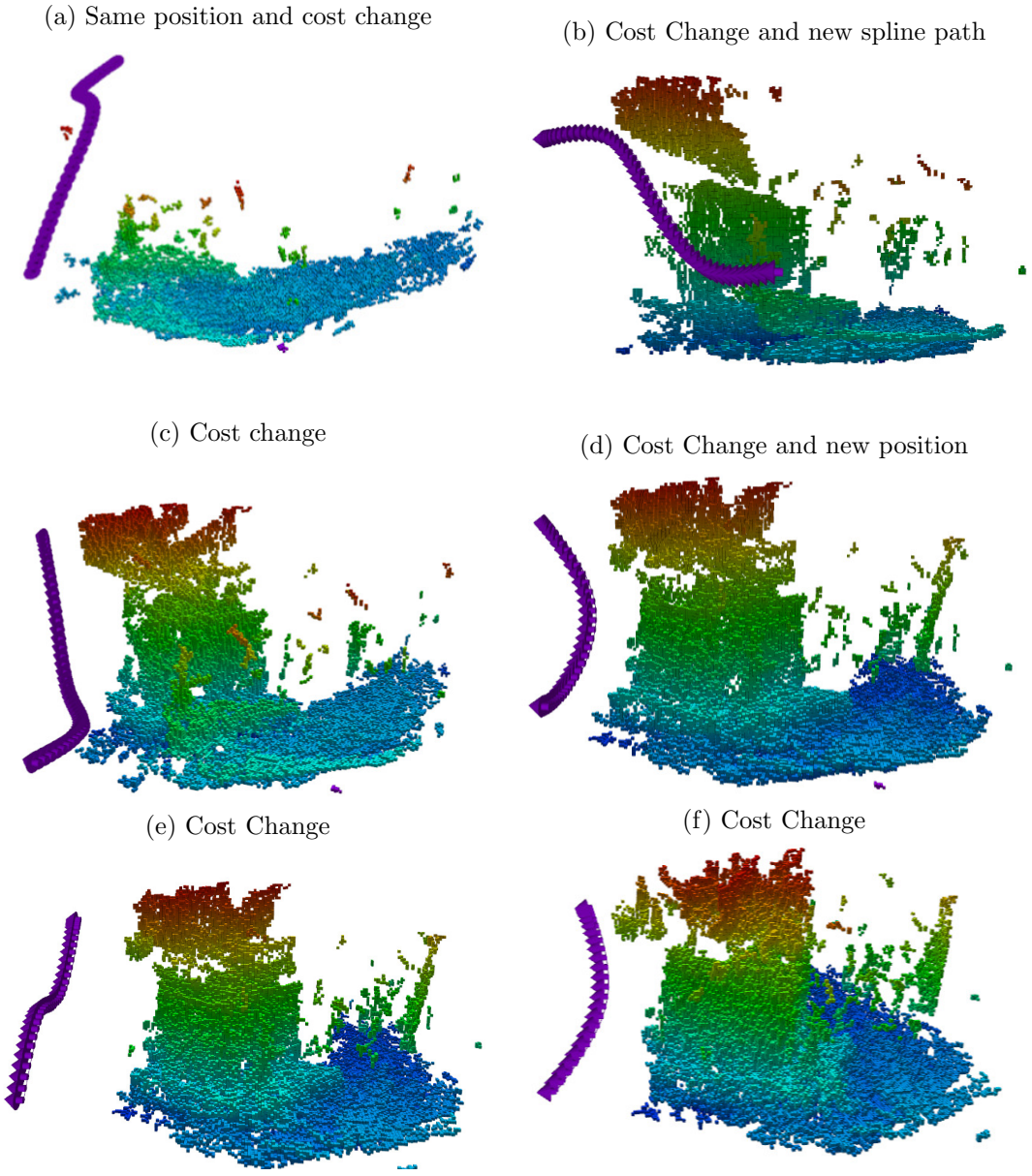


FIGURE 5.14: Three dimensional planning with single obstacle

Figure 5.15 shows the 3D path planned values of first 11 paths that changes the edge cost and starting position. Figure 5.15a is the initial plan with no knowledge about the map. From there the plan is adapted to avoid the occupied space and also the starting point as shown in Figure 5.15b. The same process continues in Figure 5.15c as well as in Fugure 5.15d. Again we are coming back to the initial point to see how the planning works as shown in Figures 5.15e, 5.15f, 5.15g, 5.15h, 5.16a.

Figure 5.16 shows the remaining path planning values. Figures 5.16b, 5.16c, 5.16d, 5.16e, 5.16f, 5.16g, 5.16h, 5.16i show the difference between the adapted path compared to previous one as well as the starting position change.

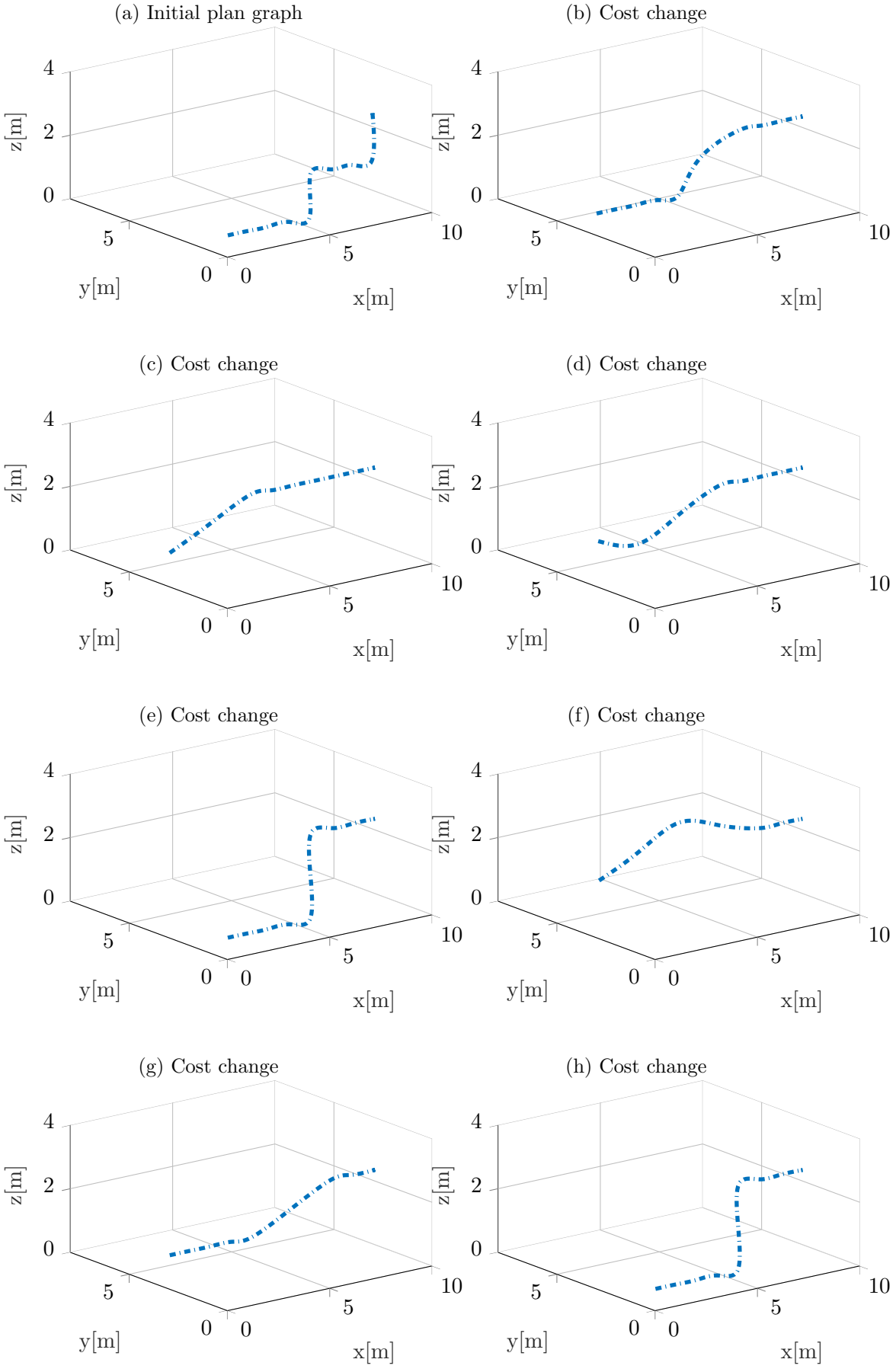


FIGURE 5.15: Three dimensional planning with single obstacle

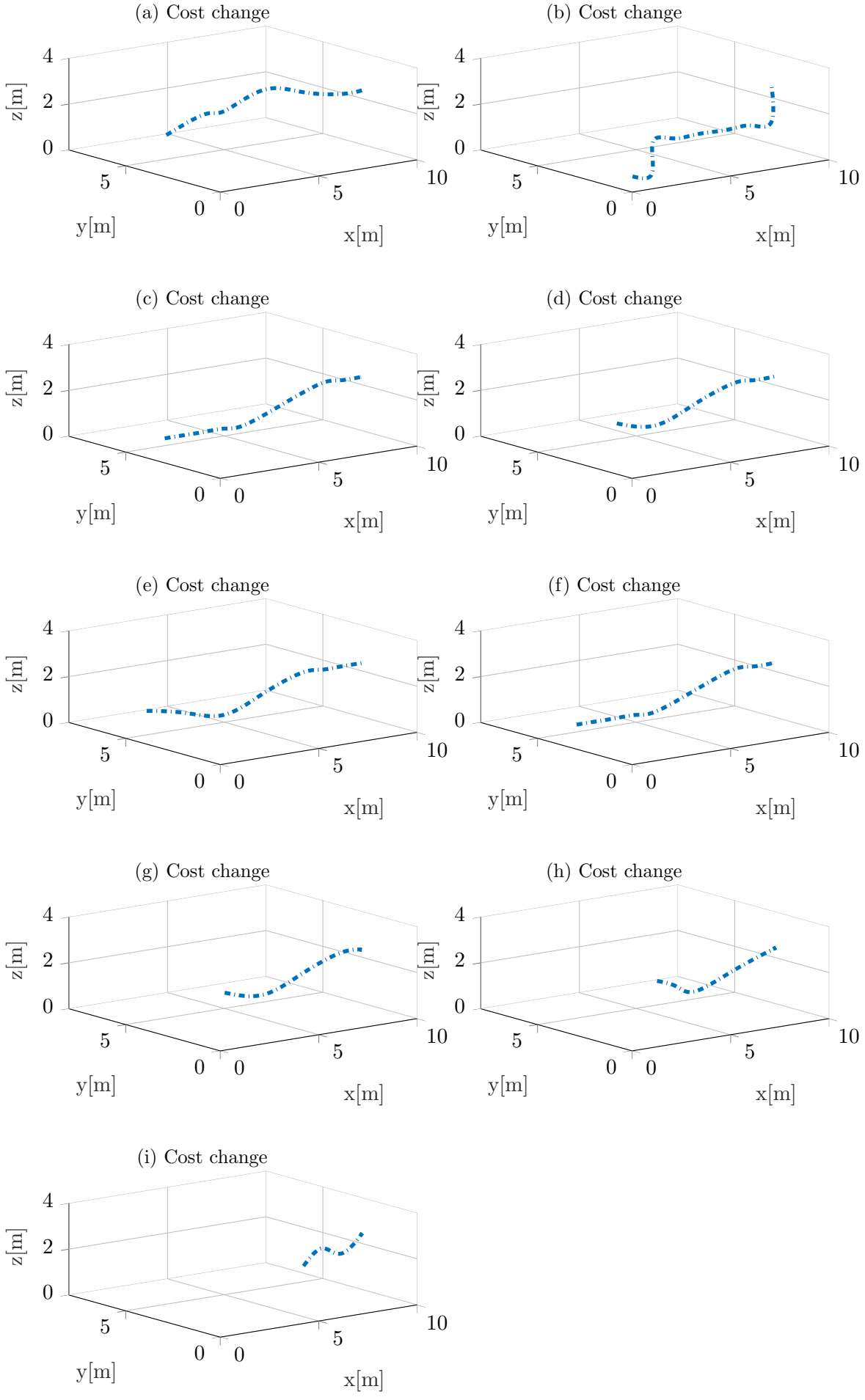


FIGURE 5.16: Three dimensional planning with single obstacle

5.6 Concluding remarks

This chapter investigated the 3D path planning algorithm developed using modified D* lite algorithm for the small UAV. First, we explained about the generic D* lite and its extension to 3D. Later the method for three grid search based on the 3D D*lite is described. Our approach is implemented through the real time experiments. We have inferred that the re-planning as well as generating spline using the original path would be more realistic solution for the real time autonomous navigation of the small UAV.

Chapter 6

Experiments and Analysis

This Chapter presents the experiment results of the whole approach conducted by combining the 3D Mapping, Localization and Path Planning in the onboard computer[B]. The Outcome of the experimental results are presented and discussed elaborately here.

6.1 Experiment setup

We used small UAV DJI matrice 100 for all the experiments conducted here. Figure 6.1 shows the experimental setup with the stereo camera mounted on the small UAV in the forward direction. The main objective of these experiments is to check the efficiency, accuracy of our combined approach which includes the online three dimensional mapping and the collision free path in the onboard computer[B].

FIGURE 6.1: Experiment setup

6.2 Experiment 1

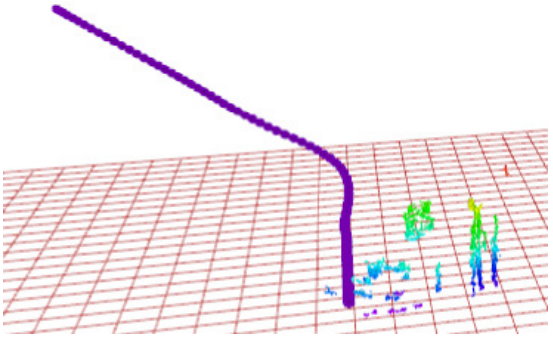
These experiments are conducted in a static outdoor environment which consist of trees, direction pole, radar (big obstacle made of iron structure), light stand fixed on the grass. We fly the small UAV using remote pilot around the obstacles with an altitude between 0 m to 2 m. We choose this base length due to the outdoor scenario where the small UAV will navigate.

The main objective is to check the performance of our complete approach which includes graph SLAM for mapping and three dimensional modified D* lite for collision free path planning. First the graph SLAM and visual stereo odometry algorithms are launched together. Once the Odometry starts to compute, we started our three dimensional path planning algorithm to find the collision free path. All the algorithms (mapping, localization and collision free planning) are computed in the on-board computer. During the experiments sensitivity to light was observed when the camera was directly exposed to sunlight. Eventhough the exposure is kept to minimum of 1 ms still it spread more light on the raw images.

Figure 6.2a shows the initial position of the small UAV in the outdoor environment. At this position, we started our mapping and localization algorithm to check the quality of the odometry. Once the mapping started, we received the minimum and maximum occupancy grid bounds. Now the goal in world coordinate system is to reach 19 m in x direction, 9 m in y direction, 5.5 m in z direction from the initial position (0,0,0) of the vehicle. While starting the path planning algorithm the current position of the vehicle is at (1, 1, 0.5). As we can see from (Figure 6.2b) the collision free path to the goal point (19, 9, 5.5) is generated with the initial map. Now after the vehicle take off Figure 6.2c the current position of the path is changed and the rest of the previous plan remains the same like in (Figure 6.2d). Figure 6.2h shows the updated map with iron structure as well as the updated collision free path. Here the previous paths are not valid because the cost of the cells and the current position is entirely new. The same path is adapted with the new current position and orientation like in Figure 6.3d, 6.3f, 6.3h. Here the main obstacle is the iron structure which has a non uniform structure. Based on the cost update the mapping and collision free planning is amended. The total flight time is 163 ms. During this flight our method computed 88 octomap messages and 808 stereo visual odometry messages (ROS format). For collision free path we have generated 58 paths. Out of 58 paths, 16 paths are shown here. These 16 paths show changes either in the starting point of the previous paths or in the cost of the cell where the previous paths are planned. If the cost of the cell reaches the maximum then previous paths are not valid for collision free. So it has to change and find the collision free path.

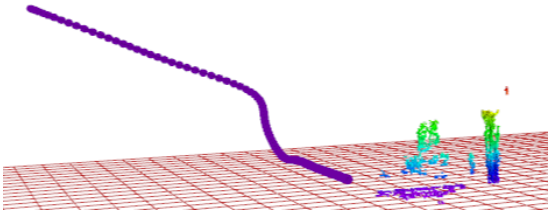
(a) Outdoor scenario and Small UAV's initial position

(b) Initial map at time 1 s and collision free path



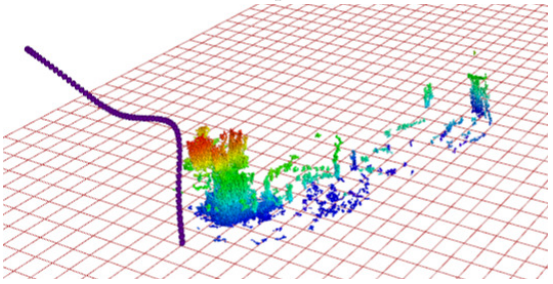
(c) Small UAV take off

(d) At time 6 s updated map and its collision free path



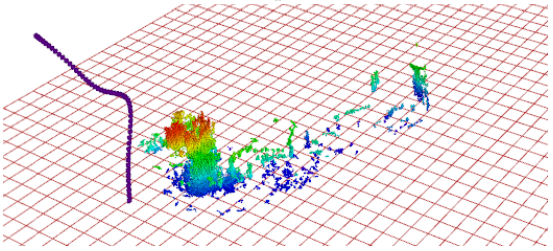
(e) Image showing the iron structure and the small UAV (left side)

(f) At time 99 s updated map and its collision free path



(g) Small UAV navigating around the iron structure

(h) At time 115 s updated map and its collision free path



(i) Small UAV navigating around the iron structure

(j) At time 102 s updated map and its collision free path

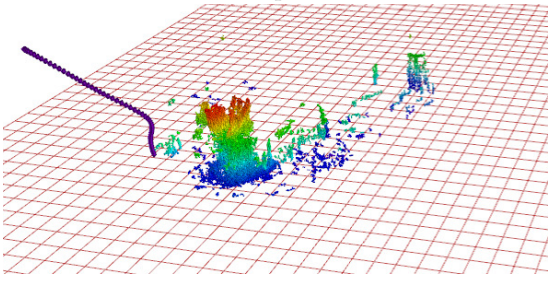
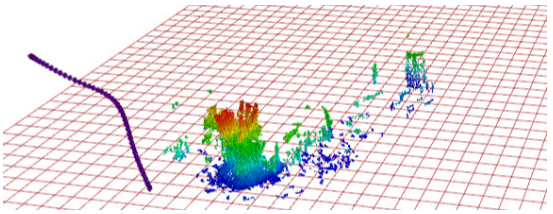


FIGURE 6.2: Outdoor experiment with mapping, localization, 3D path planning

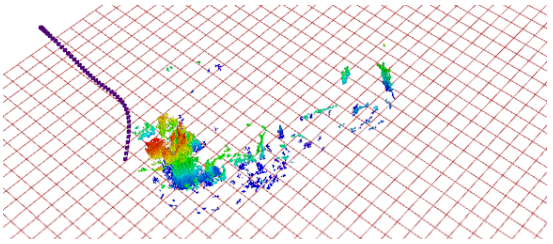
(a) Small UAV navigating around the iron structure

(b) Updated map and collision free path generated at time 121 s



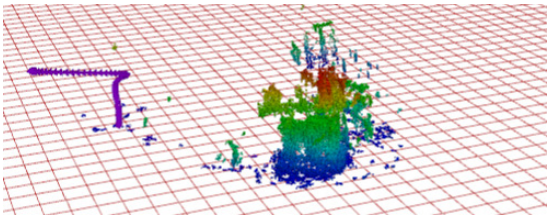
(c) Small UAV navigating near to the iron structure

(d) Updated map and collision free path generated at time 127 s



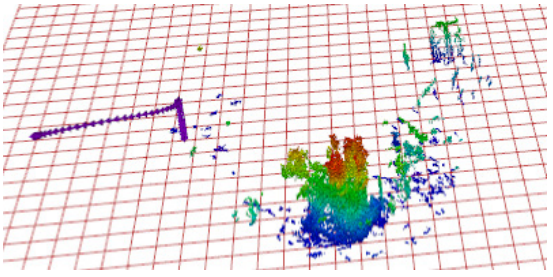
(e) Small UAV navigating around the iron structure

(f) Updated map and collision free path generated at time 129 s



(g) Small UAV navigating around the iron structure

(h) Updated map and collision free path generated at time 140 s



(i) Small UAV navigating around the iron structure

(j) Updated map and collision free path generated at time 156 s

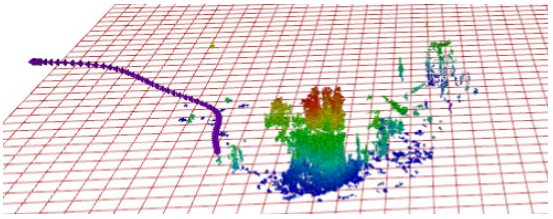
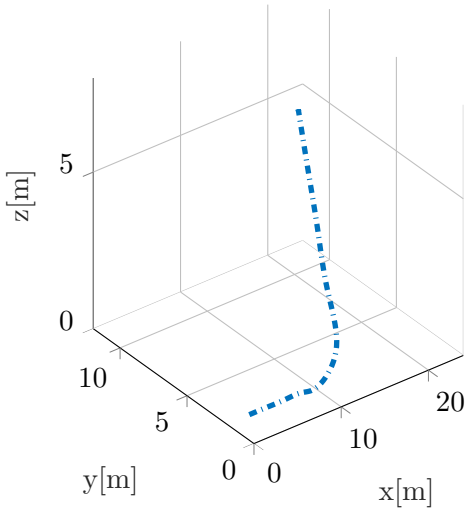


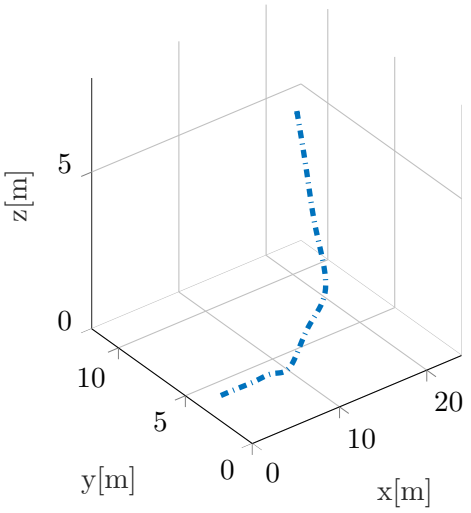
FIGURE 6.3: Small UAV in the Outdoor scenario with three dimensional octomap and 3D collision free path.

Figure 6.4a shows the initial plan which is adapted according to the map update at time 6 s. Then it is updating the current position of the vehicle in Figure 6.4c. Later it is modified due to the obstacle (iron structure) in Figure 6.4d. Figure 6.5a and Figure 6.5b gives the updated current position as well the path. Now the vehicle altitude is increased to 1.62 m at time 46 s. From this position the obstacle is not infront of the vehicle, the collision free path is just straight forward like in Figure 6.5c. The same plan continues until time 83 s with current position changing at time 72 s and 81 s and 84 s. Now the vehicle is on the other side of the iron structure and the goal point is above the vehicle like shown in Figure 6.6b. Later it is changed at time 129 s which is shown in Figure 6.6c.

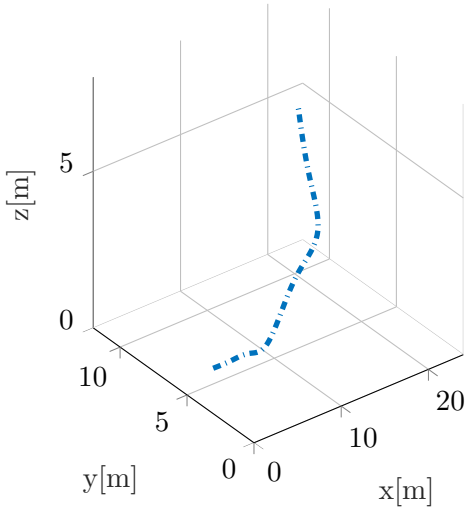
(a) Initial plan



(b) Updated plan at time instance 6 s



(c) Updated plan at time instance 13 s



(d) Updated plan at time instance 28 s

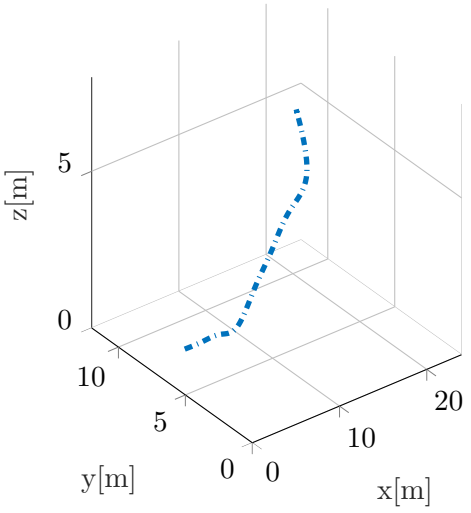
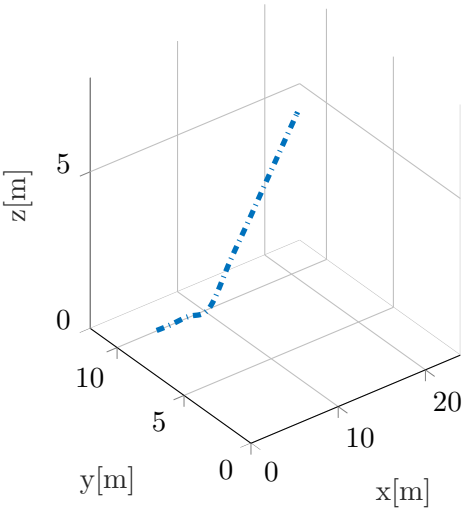
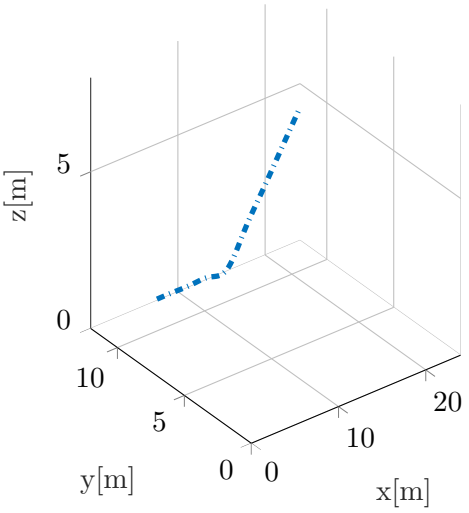


FIGURE 6.4: Online three dimensional collison free path planned in the outdoor environment from time 0 to 28 s

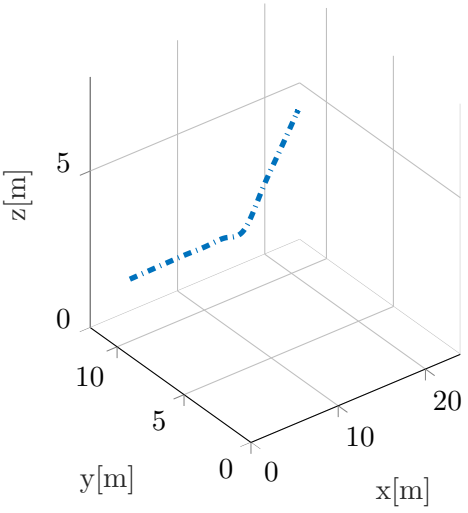
(a) Updated plan at time instance 33 s



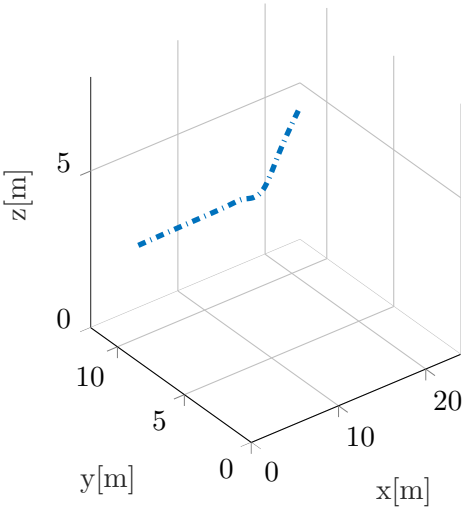
(b) Updated plan at time instance 36 s



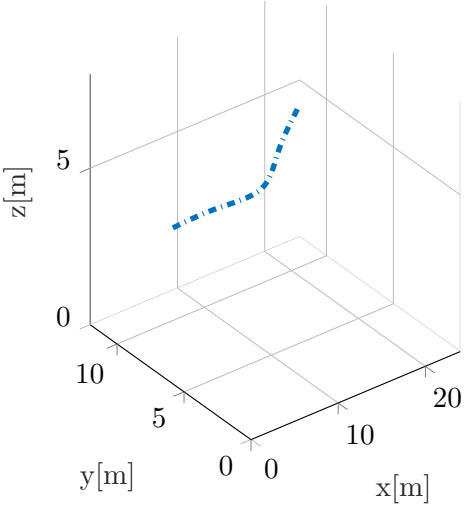
(c) Updated plan at time instance 46 s



(d) Updated plan at time instance 72 s



(e) Updated plan at time instance 81 s



(f) Updated plan at time instance 84 s

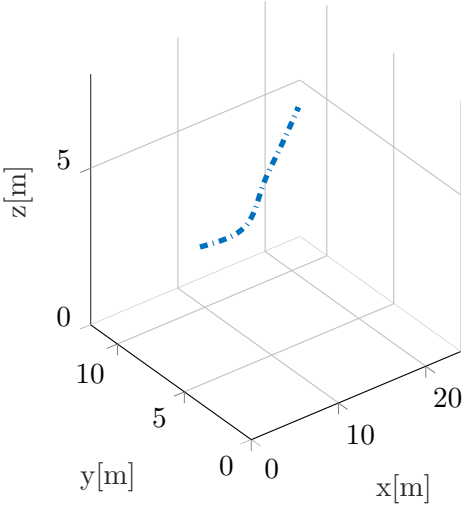
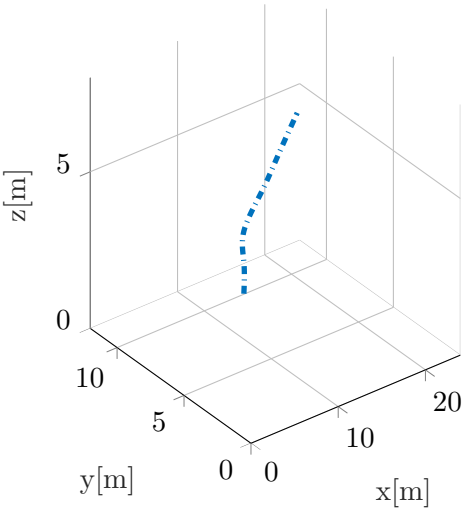
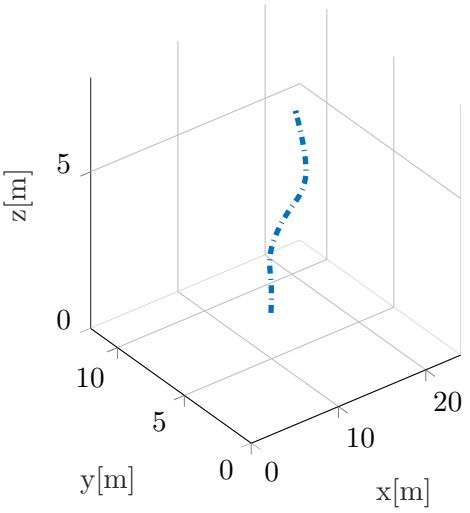


FIGURE 6.5: Online three dimensional collison free path planned in the outdoor environment from time 33 s to 84 s

(a) Updated plan at time instance 102 s



(b) Updated plan at time instance 108 s



(c) Updated plan at time instance 129 s

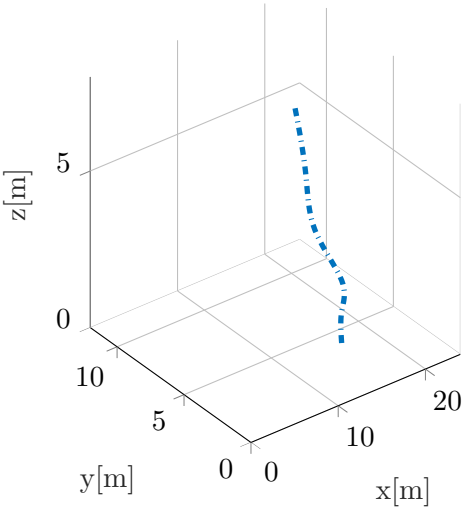


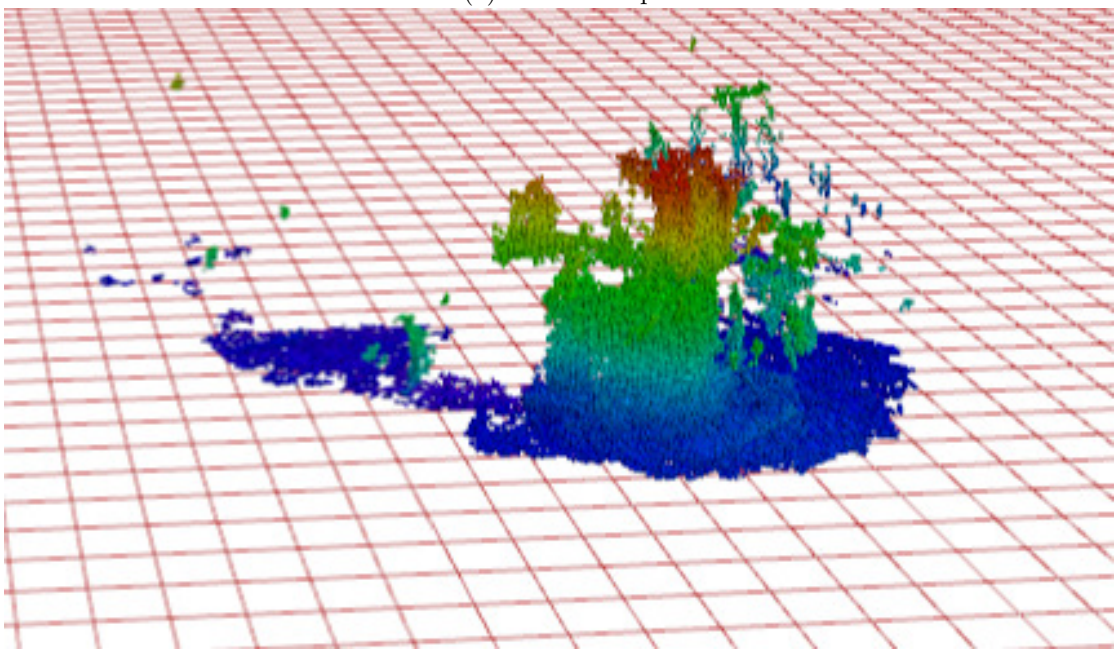
FIGURE 6.6: Online three dimensional collison free path planned in the outdoor environment from time 102 s to 129 s

Figure 6.7a shows the whole 3D octomap obtained at the end of this experiment. Here the stereo visual odometry is shown with the position as well as the orientation of the stereo camera like in Figure 6.7b.

As shown in Figure 6.8c overall 192 nodes are formed. Of that the maximum time of 676.967 s is taken by the node 192 whereas the minimum time of 2.645 s is taken by the node 32. During the experiments the computation varies for the same configuration. This is due to the continuous usage of the processor. For example if process starts while the CPU was not in use for long hours then we observed speed pickup in the process while on the other hand, if it used after couple of experiments then there is a lag in the processing power.

In Figure 6.8b we can observe the time taken to detect the keypoints in the stereo images. On an average, 2.61 ms is taken for the keypoint detection. At the same time the descriptor extraction is increased linearly with respect to the nodes. This implied that the features are tracked in each frame of the images continuously. The average time for descriptor extraction is 1.43 ms. Along with that the signature creation took a minimum time of 6.76 ms. Here the occupancy grid took an average of 6.15 ms. With this performance real time collision free path planning is achieved in real time with an average of 28 ms. Also this average applies only to this particular experiment conditions like 3d map update, memory availability and computational power. The advantage here is that it adapts to the previous path instead of creating entirely new path in comparison to the previous plan. This will help in realistic precise navigation.

(a) 3D Octomap



(b) Stereo Visual Odometry

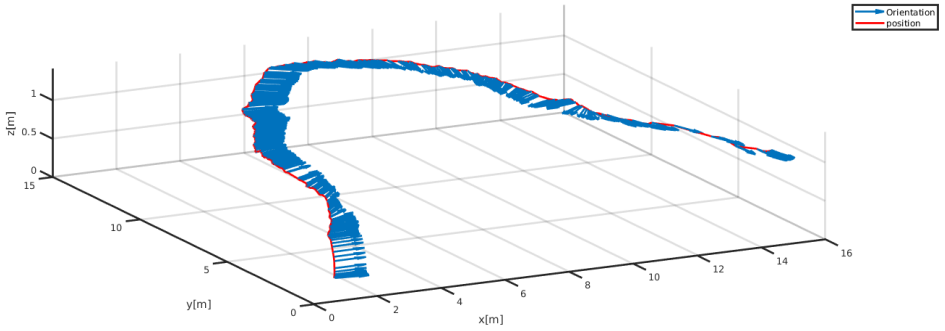


FIGURE 6.7: Final Octomap and the Stereo Visual Odometry

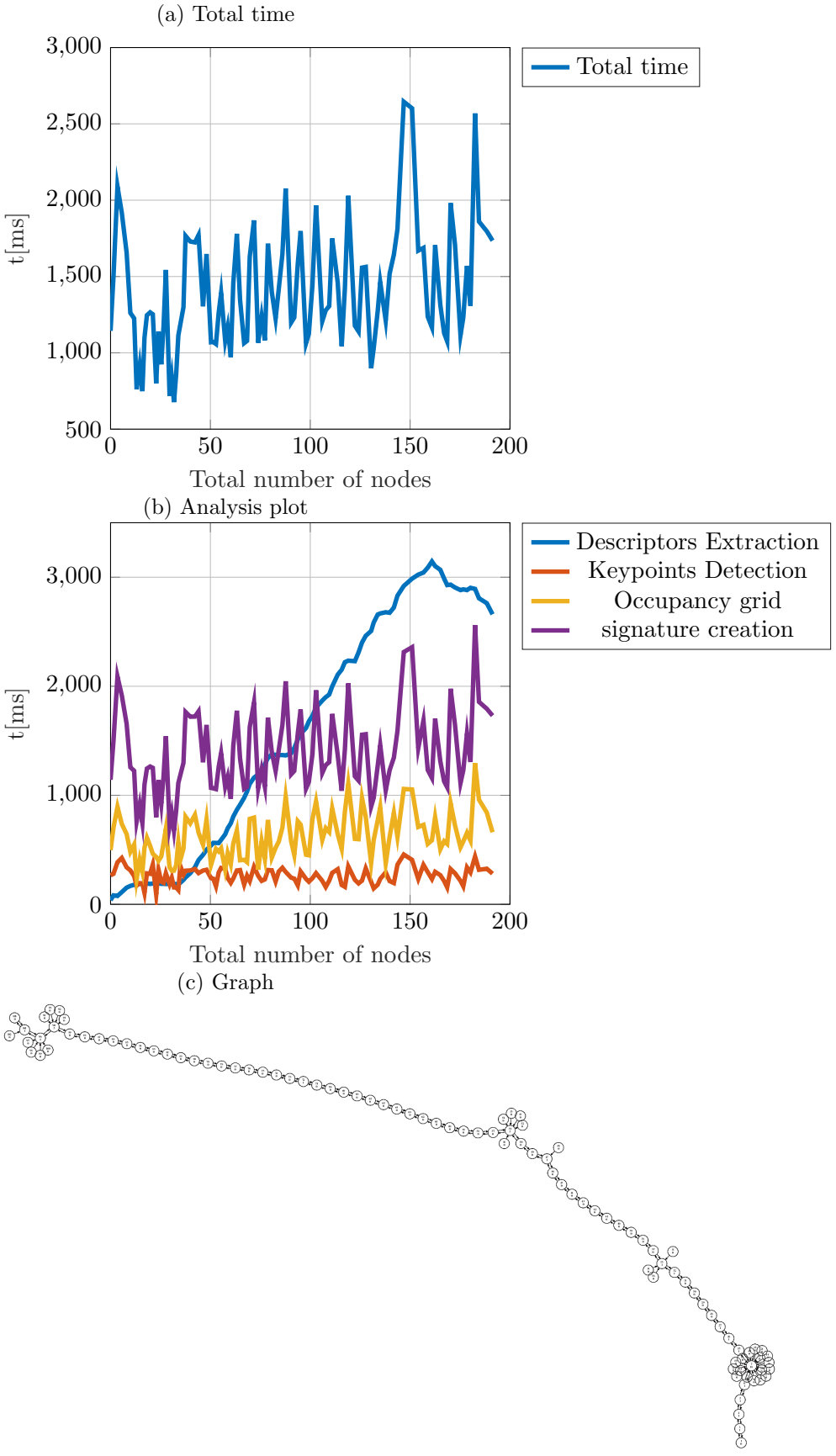


FIGURE 6.8: Plot showing the total time taken to compute, keypoint descriptor, keypoint detection, occupancy grid, signature vs total number of nodes generated and the Graph shows the nodes and edges.

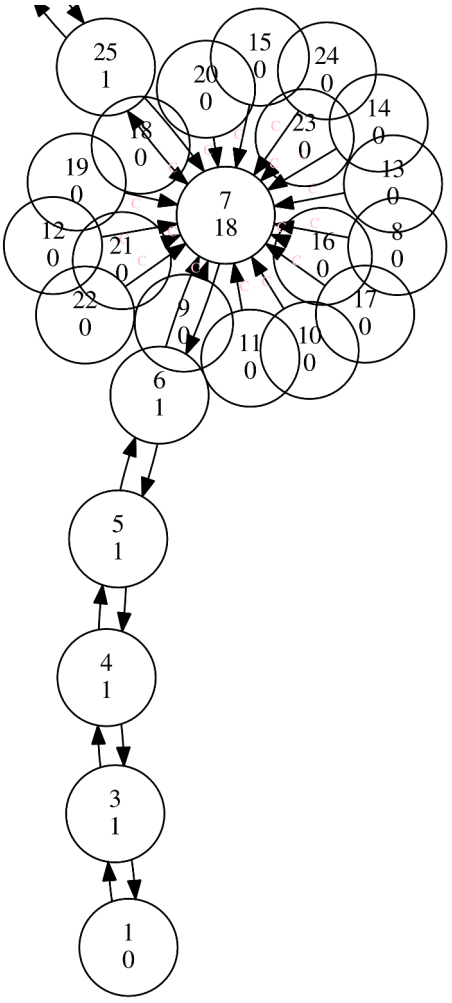


FIGURE 6.9: Graph nodes formed in the beginning.

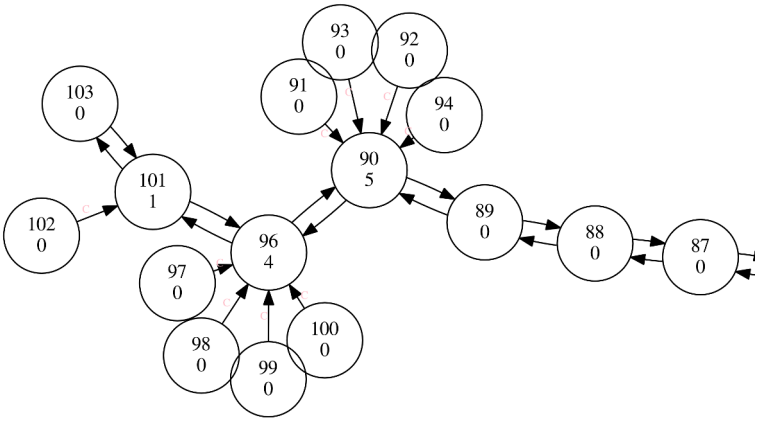


FIGURE 6.10: Graph nodes formed at the end.

6.3 Experiment 2

This experiment was also conducted in the same outdoor environment. But in contrast to the previous experiment the initial and final position are reversed to the other direction and the obstacles are viewed in from different position. Since the range of the camera is limited to 8 m, we have reduced the speckles generated from long range objects. Like in the previous experiment we focus on the time taken to compute realistic collision free path in the onboard computer.

Figure 6.11a shows the initial position of the small UAV on the ground. On the right side there are few green plants cluttered around. Now the goal in world coordinate system is to reach 17 m in x-direction, 9 m in y-direction, 6.5 m in z-direction by avoiding the obstacles coming on the way. The reason to choose this goal point is that it is directly in a collision course with the big iron structure like shown in Figure 6.11a.

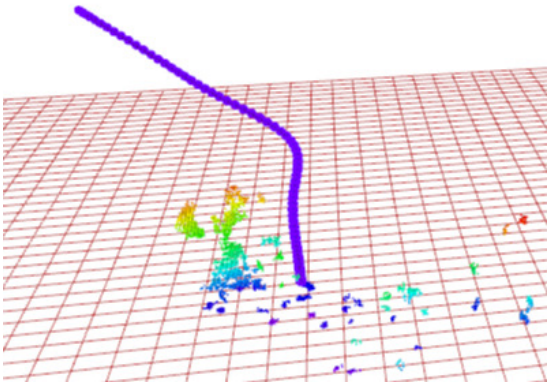
We started the mapping and stereo visual odometry to generate the occupancy grid. Then 3D planning started at time instance 1 s. The complete collision free plan without all the information about the obstacles is shown in Figure 6.11b. The same path is valid until 3 s except the current position is adapted in the plan. Later at time instance 5 s it is beyond the iron structure like shown in Figure 6.11h. It finds the shortest path since there are no obstacles around.

Next the vehicle moved near the iron structure and now the plan avoids the obstacle and generates a collision free path like in Figure 6.12d. In the same Figure 6.12d the upper part of the map is updated, which can be noticed. The updated plan mainly shows the difference in the z direction which is beneficial in terms of short path. Figure 6.12h shows the collision free path where the vehicle is before the wall. Here the ground map is shown because it is not taken into account for the planning.

The total flight time for the current experiment is 118 s. During this experiment we generated 55 octomap messages and 753 stereo odometry messages (ROS format). While at the same time our 3D path planning approach generated 69 realistic collision free path. With this performance real time collision free path planning is achieved in real time with an average of 39 ms.Out of 69 collision free path we presented 10 paths in terms of images for explanation purpose. Similarly 13 messages which show change either in the starting position or cell cost update coming from the online map are also presented as graphs in Figure 6.13 and Figure 6.14.

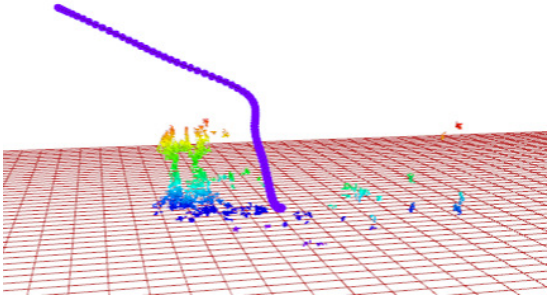
(a) Scenario and small UAV initial position

(b) At time 1 s map and the collision free path



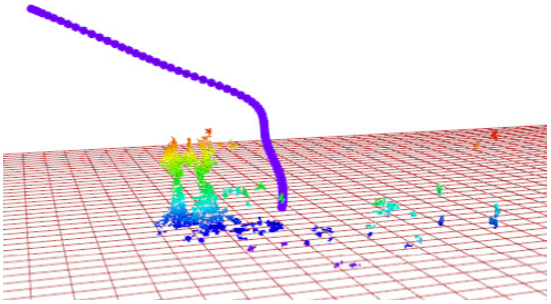
(c) Scenario and small UAV near to iron structure

(d) At time 2 s map and the collision free path



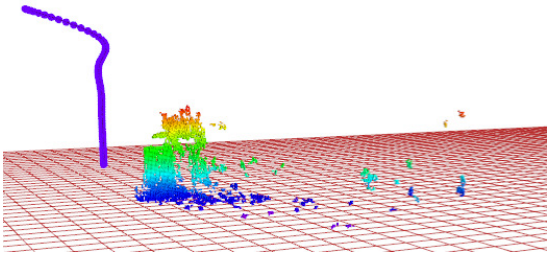
(e) Scenario and small UAV around the iron structure

(f) At time 3 s map and the collision free path



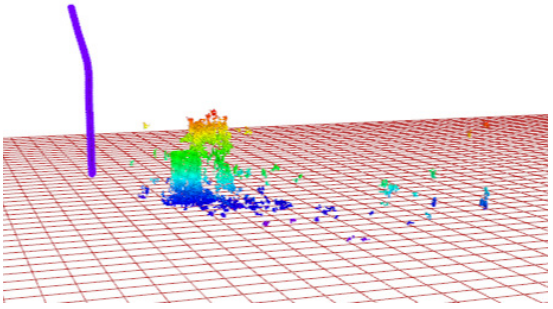
(g) Scenario and small UAV around the iron structure

(h) At time 21 s map and the collision free path



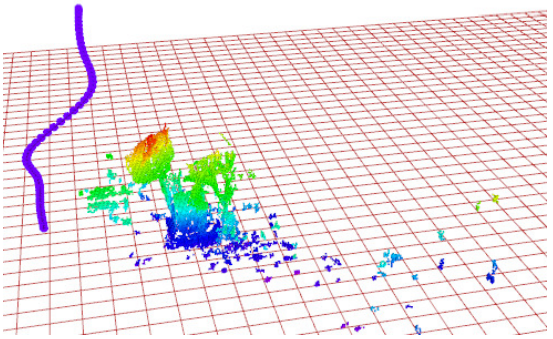
(i) Scenario and small UAV behind the iron structure

(j) At time 32 s map and the collision free path



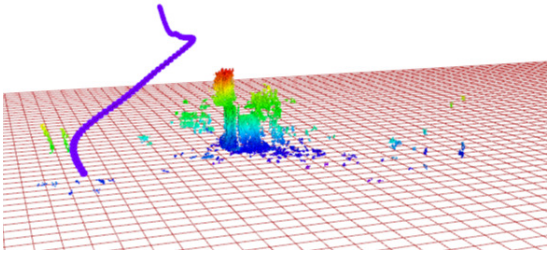
(a) Scenario and small UAV near to the rion structure

(b) At time 43 s map and the collision free path



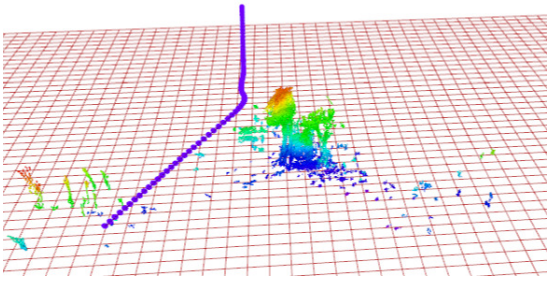
(c) Scenario and small UAV around the iron structure

(d) At time 74s updated map and path



(e) Scenario and small UAV around the iron structure

(f) At time 85 s updated map and the path



(g) Scenario and small UAV around the iron structure

(h) At time 112 s updated map and the path

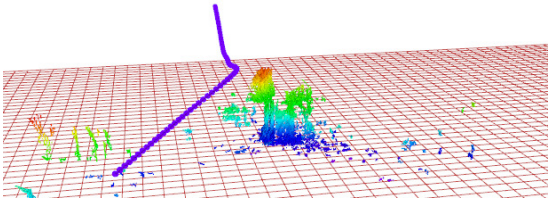
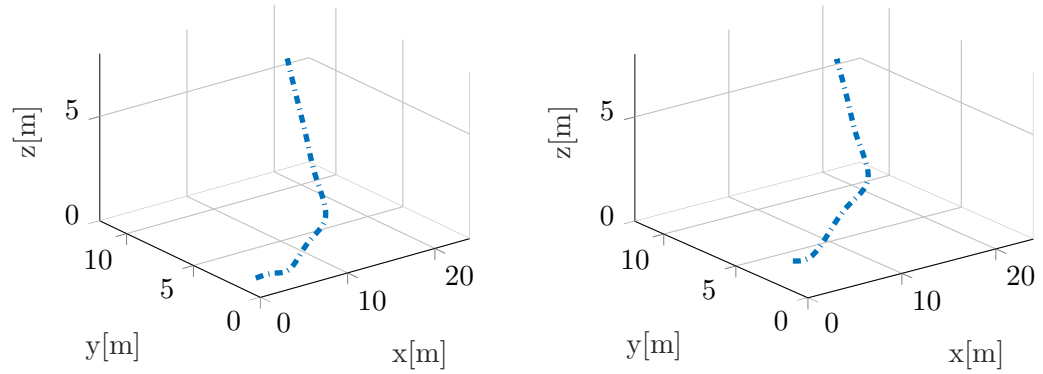


FIGURE 6.12: Outdoor experiment with mapping, localization, 3D path planning

Figure 6.13a shows the initial plan which is adapted and changed at time instance 3 s like in Figure 6.13b. The same plan is valid until 21 s. At time instance 32 s to 43 s there is considerable online map cost update for the planner. But later at time instance 46 s there are no objects infront of the vehicle. The obstacle is upated in the map at time instance 52 s and it is avoided by the planner like in Figure 6.14a. From time instance 53 s to 73 s the vehicle doesn't make considerable movements so the previous plan is valid for the collision free path. The current position is updated at 74 s and a new path is also generated as seen in Figure 6.14b.

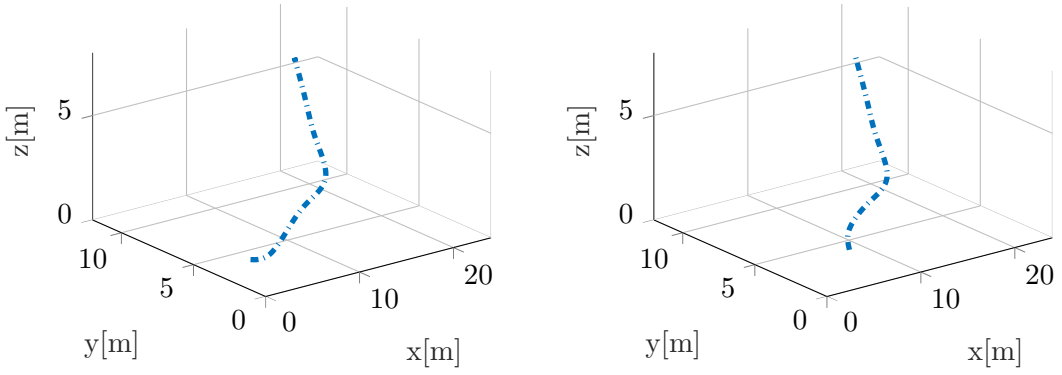
(a) Initial plan at time instance 1 s

(b) Updated plan at time instance 3 s



(c) Updated plan at time instance 21 s

(d) Updated plan at time instance 32 s



(e) Updated plan at time instance 43 s

(f) Updated plan at time instance 46 s

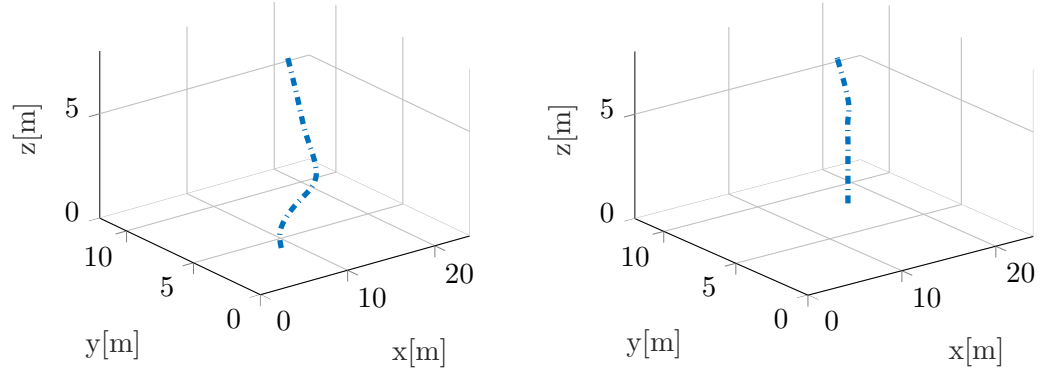
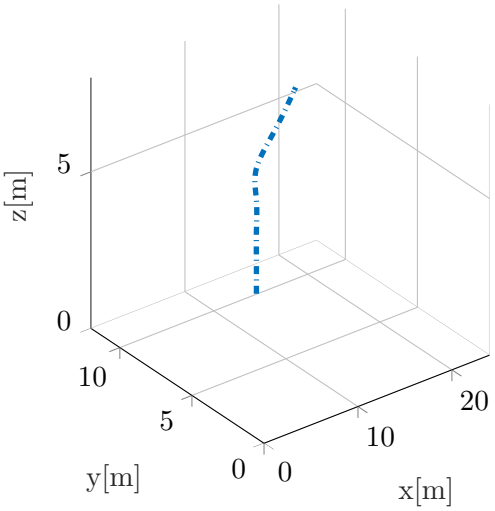
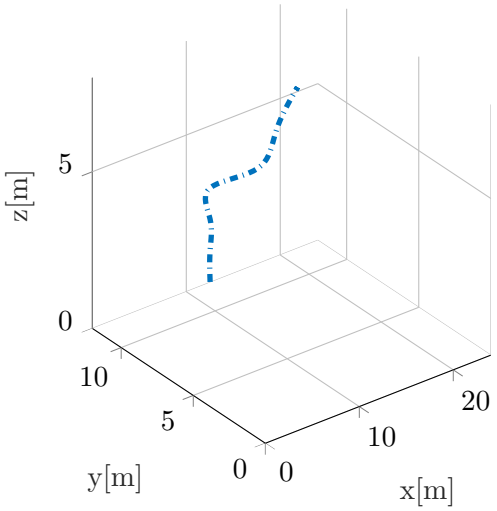


FIGURE 6.13: Online three dimensional collision free path planned in the outdoor environment from time 1 s to 46 s

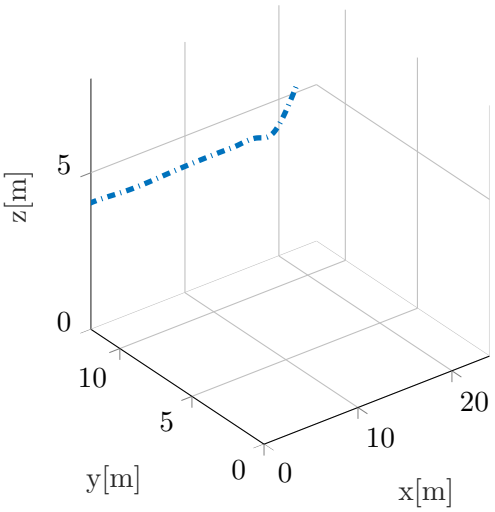
(a) Updated plan at time instance 52 s



(b) Updated plan at time instance 74 s



(c) Updated plan at time instance 85 s



(d) Updated plan at time instance 112 s

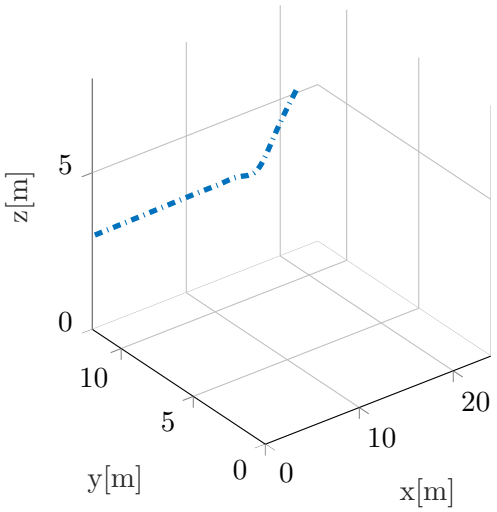
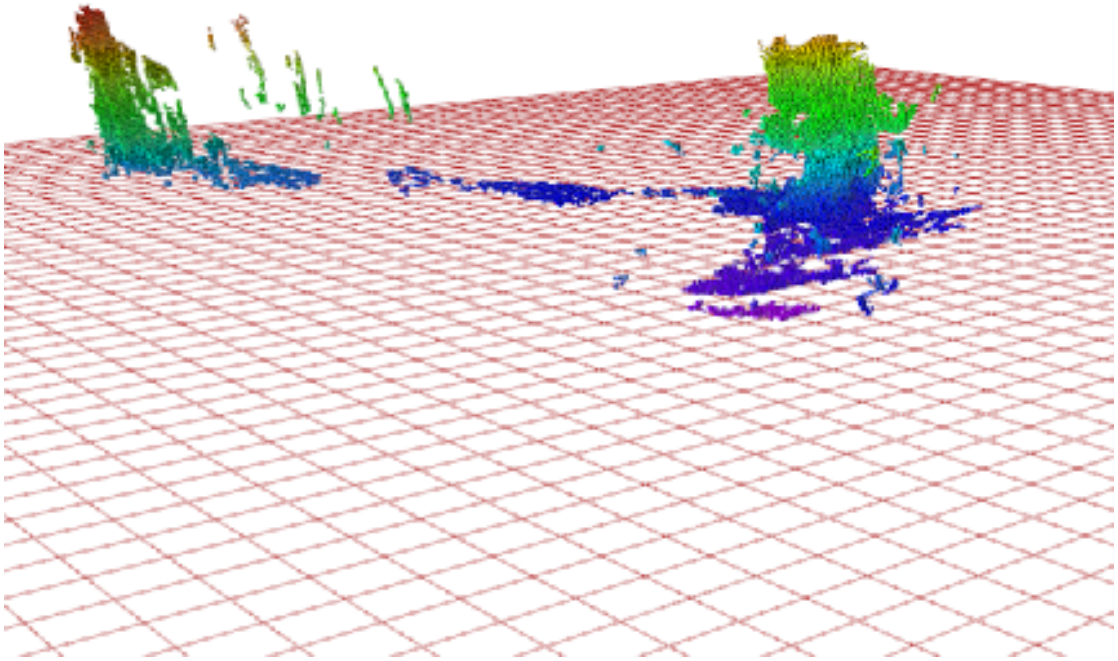


FIGURE 6.14: Online three dimensional collison free path planned in the outdoor environment from time 52 s to 112 s

The final octomap and stereo visual odometry is shown in Figure 6.15. The two main objects that can be noticed in the map: iron structure and the wall. Totally 215 nodes are formed in the form of graph which is shown in Figure 6.16c. The time taken by each node to compute the detection of the Keypoint, , descriptor extraction, occupancy grid and signature creation is shown in Figure 6.16b. On an average all the nodes took 1.9 ms to extract the descriptors where as to detect the keypoints it took 3.25 ms. There is no loop closure detected in this experiment because of the path we travelled. More specifically no places visited again with a frame difference of more than 10 frames. As we can see in Figure the 6.16b the same pattern like previous experiment forms here. Again it is due to arrangement of objects (which provide features) in the environment.

The average total time for processing the complete approach to generate occupancy grid and stereo visual odometry is about 16.69 ms.

(a) 3D Octomap



(b) Stereo Visual Odometry

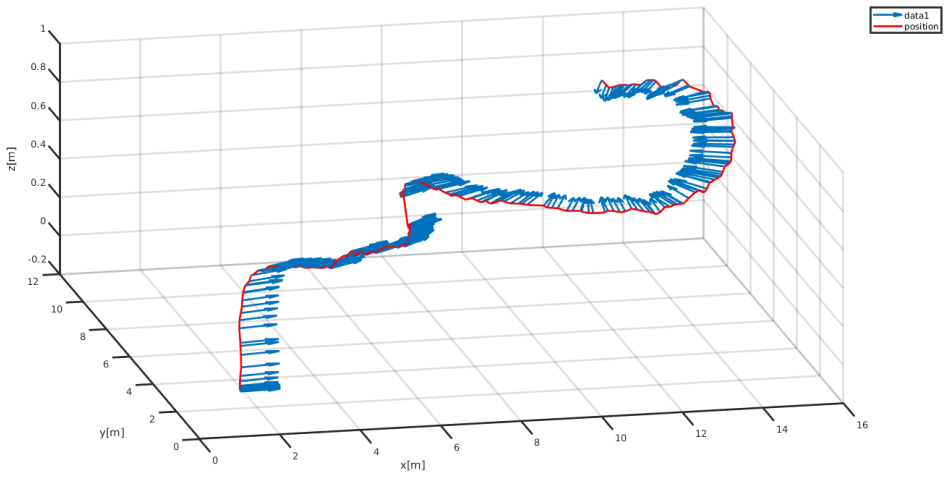
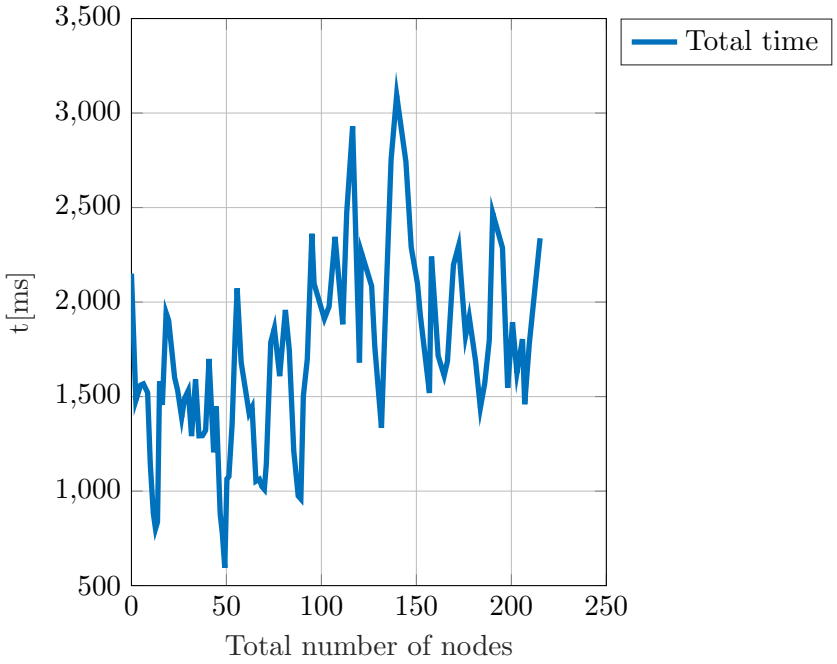
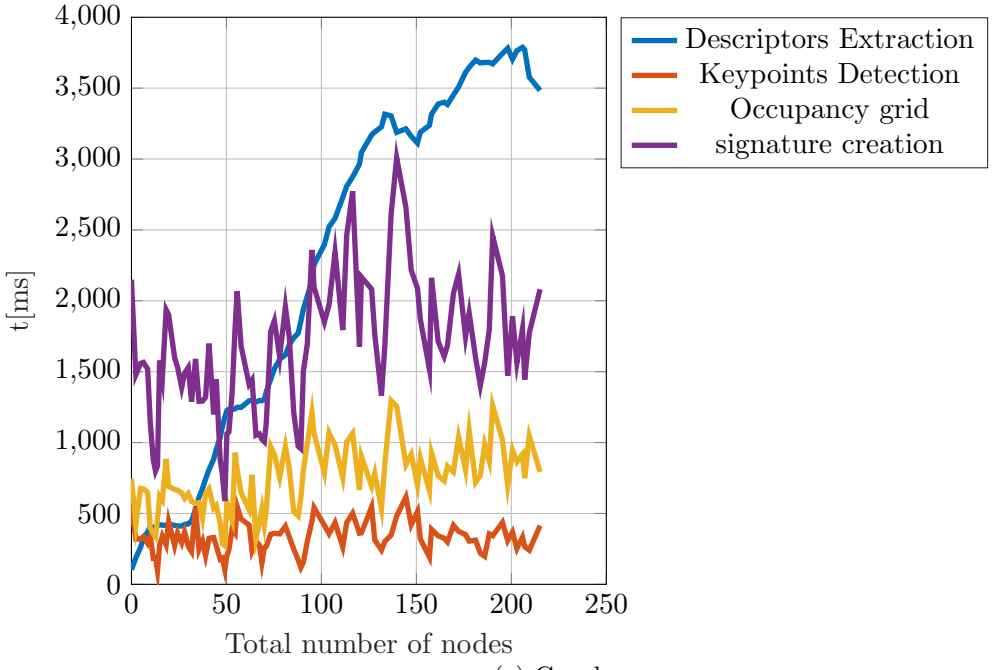


FIGURE 6.15: Final Octomap and the Stereo Visual Odometry

(a) Total time



(b) Analysis plot



(c) Graph

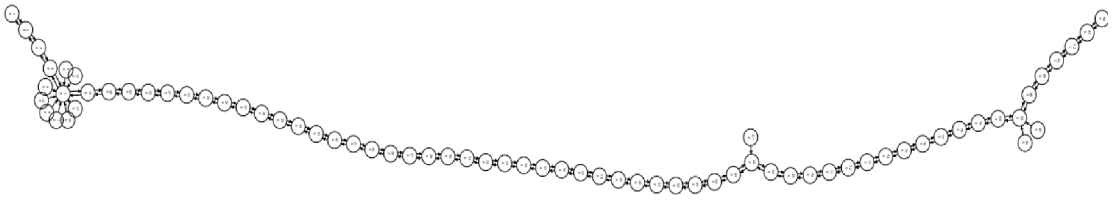


FIGURE 6.16: Plot showing the total time taken to compute, keypoint descriptor, keypoint detection, occupancy grid, signature vs total number of nodes generated and the Graph shows the nodes and edges.

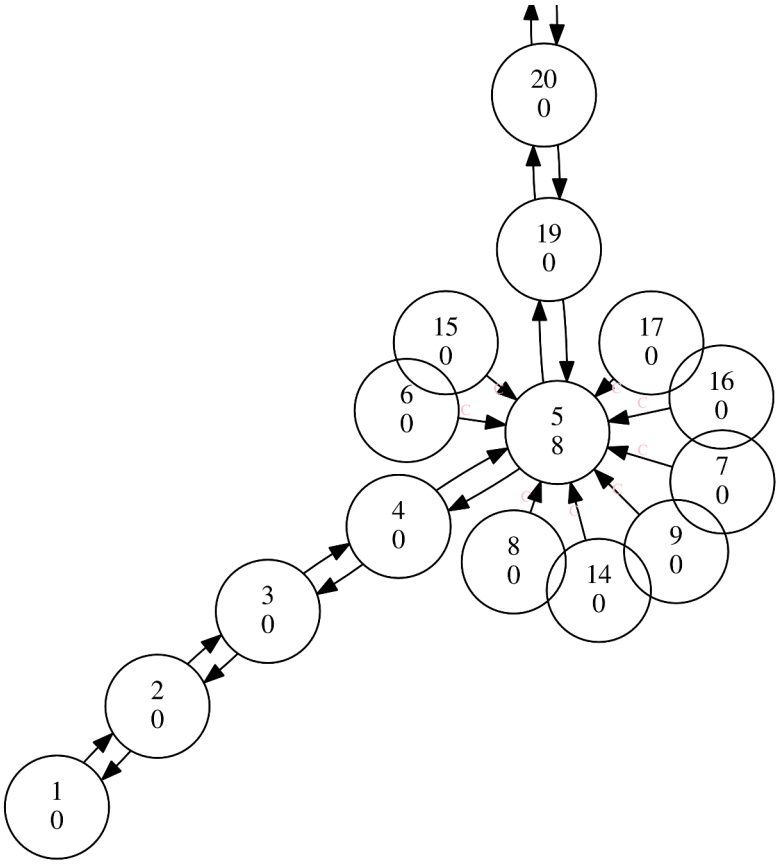


FIGURE 6.17: Graph nodes formed in the beginning.

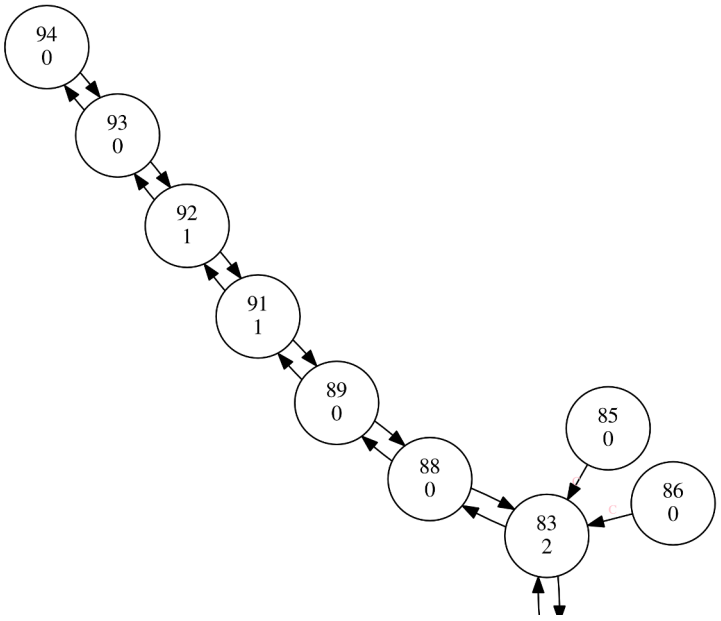


FIGURE 6.18: Graph nodes formed at the end.

6.4 Concluding remarks

We conducted two experiments on the small UAV combined with 3D mapping, localization and 3D collision free planning in the onboard computer. The presented experimental results showed that the generated collision free path has a minimum distance of atleast 1 m to 1.5 m from the obstacle. Since both the experiment conducted in the similar outdoor environment the starting point for the both the experiments are different from each other. we can see how the light variation created an impact in the feature extraction process. Though the speckles were formed in both the octomaps obtained through experiments, it doesn't affect the collision free planning generation in the real time.

Chapter 7

Conclusion

This chapter summarizes the investigations as well as the developed solution for small UAVs collision avoidance in unknown outdoor environment. The goal of this work is to develop real time collision avoidance for small UAV that can be deployed in outdoor environment. By conducting outdoor experiments regularly with our developed solution we have presented the detailed results obtained. In extension to that we are also giving a perspective for the future work here.

7.1 Summary

For stereo vision based navigation, we developed our own custom made stereo camera in Chapter 3. First the design issues and hardware implementation was addressed through two custom made stereo camera models. Similarly the problem of sensitivity and synchronization were discussed. Then further in the context of stereo camera the fundamental aspects pertaining to camera calibration, depth estimation, stereo matching and image rectification is addressed. Through our experiments, our approach handled the challenges in generating 3D point clouds in different outdoor condition by the stereo camera. Besides that we have explained the necessity and advantage of the custom stereo camera instead of using the COTS product. Further more we have demonstrated the adjustable baselength can be an option to reduce the noises and speckles.

Next to that, we used graph based SLAM for real time 3D mapping and Localization. The 3D mapping method considered here also includes an online loop closure detection method to identify the previously visited areas and optimize the nodes in parallel. It is proven in our experiments that no artificially created landmarks or known objects are required for the 3D mapping. Besides, the proposed solution can also handle the moving

objects by overwriting the point clouds when the object is moving from the field of the view of the stereo camera. The proposed approach for 3D mapping is executed alone and later it is combined with our 3D path planning in various scenarios. The memory and time to compute the map are under the constraints of the onboard computer. In our experiments we also observed the influence of available image textures on the precision of localization and mapping. Similalry the 3D octomap with speckles are often due to clear textures.

Then, the three dimensional path planning in real time is developed using the modified D* Lite algorithm for online to use the octomap to calculate the cost of th cell in real time. Taking practical scenario into consideration we implemented the path smoothing using the spline generation in addition to the original path. In all the experiments we have tested the cost calculation of the cells, optimal path and realistic path (spline curve) using the spline function. Since we used our own custom stereo camera we can adjust the sensor range so that the distance refinement will not be an issue for 3D path planning. This significantly helped in calculating the safety distance between the obstacle and the small UAV. The efficiency of the re-planning is tested through the several real time experiments. This 3D path planning is combined with online 3D mapping and executed in the same onboard computer. The combined experimental results proved the collision free navigation in the outdoor environment. Here the main criteria are time, memory and speed which are considered during the implementation. Our approach achieved the collision free planning online at an average of 28 ms to 39 ms on an online octomap.

7.2 Future work

We consider the insights of proposed solution from this thesis to suggest how this can be improved and extended in appropriate research direction. First of all the proposed 3D mapping and collision free planning can be combined with suitable real time controller for autonomous navigation.

We already designed in a way that the 3D path planning can be easily integrated with a controller for real time experiments. For practical implementation the controller should consume less computational power. The simplest of algorithms which fall under this category includes Proportional Integral Derivative (PID) methods and other algorithms where the controller gains are precomputed offline such as Linear Quadratic Gaussian (LQG), Linear Quadratic Regulator (LQR), H_2, H ∞.But these methods (LQG, LQR, H_2, H∞) also require precise system models for computing the gains. Similarly methods such as direct adaptive controls can also be used provided the adaptation of controller gains is fast and convergence can be assured in closed loop. Due to the advancements

of technology this can be tested step by step by considering the constraints of the environment.

Later the 3D mapping approach can be improved with more than one dynamic object in the environment and the same scenario can be implemented with proposed 3D collision free planner. Though the combined approach is tested with small UAV (quadrotor) platform, a complete validation of the approach would give the option to use it with other platforms.

Appendix A

Camera Configuration

The custom made stereo camera used in this thesis is designed using the two ueye camera. The connections of this stereo camera is done through master slave configuration which can be found in Figure A.1

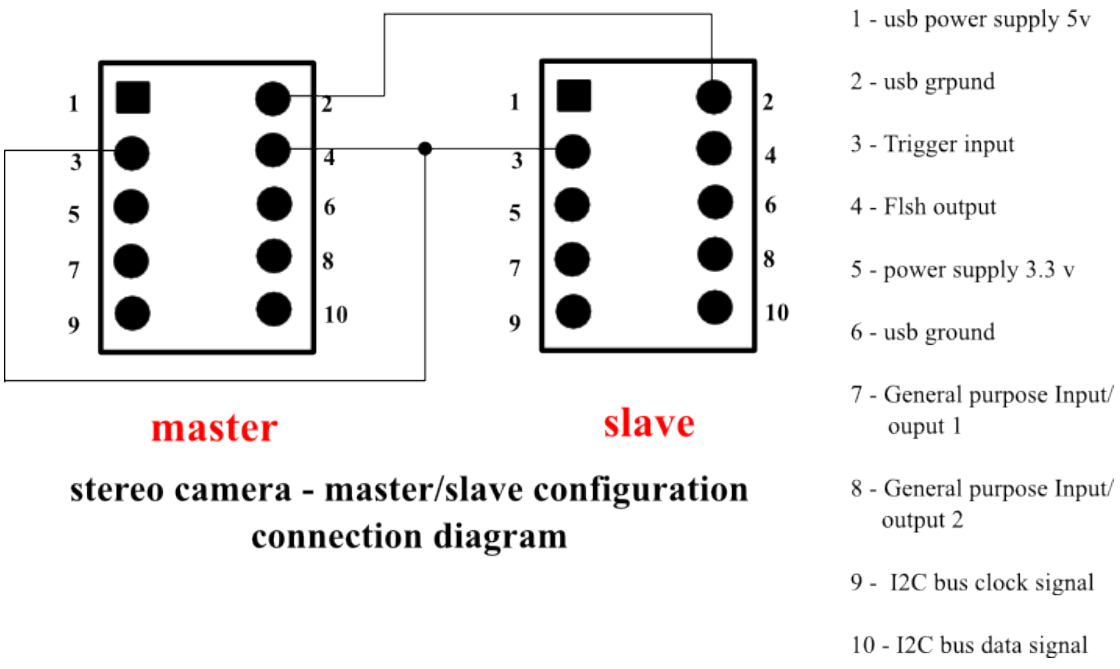


FIGURE A.1: Master slave configuration with pin and its wiring connection settings.

Table A.1 and Table A.2 shows the specification of the ueye camera model 1221LE-C-HQ and 1220LE-C-HQ respectively.

Color mode	rgb8
Sensor type	CMOS color
Shutter	Global shutter
Resolution	752 x 480
Gain (master/RGB)	4x/5x
Exposure time(min-max)	0.080 -5580ms
Frame rate trigger (max)	83
Pixel size	6 µm
Pixel clock range	5 - 40 MHz
Sensor Model	MT9V032C12STC
Power consumption	0.4 W - 1 W
Interface connector	USB 2.0 mini-B
Power supply	USB cable
Lens Mount	S-Mount
Dimensions H/W/L	36 x 36 x 20.2 (mm)
Mass	16g
Device temperature (operation)	0 °C-55 °C

TABLE A.1: Specification of ueye-1221LE-C-HQ model

Color mode	rgb8
Sensor type	CMOS color
Shutter	Global shutter
Resolution	752 x 480
Gain (master/RGB)	4x/5x
Exposure time(min-max)	80 µm-5580ms
Frame rate trigger (max)	83
Pixel size	6 µm
Pixel clock range	5 - 40 MHz
Sensor Model	MT9V032STM / MT9V032STC
Power consumption	0.4 W - 1 W
Interface connector	USB 2.0 mini-B
Power supply	USB cable
Lens Mount	**CS-Mount**
Dimensions H/W/L	**44 x 44 x 25.4 (mm)**
Mass	41g

TABLE A.2: Specification of ueye-1220LE model

FIGURE A.2: Tamaron lens model 13FM22IR

Table A.3 shows the specification of the tamaron lens used with the ueye-camera model 1220 LE.

Focal length	2.2 mm
Aperture Range	1.2 Close
Angle of view	118.6 °C x 90 °C
Focusing Range	0.2 ∞
Operating Temperature	−20 °C - 60 °C

TABLE A.3: Specification of Tamaron lens 13FM22IR model

The custom stereo camera explained in Chapter 3 (Figure 3.4) shows one of the custom stereo camera used in this thesis. This stereo setup is clamped on the carbon rod using the carbon plates which is shown in Figure A.3

(a) Baseplate 1

(b) Baseplate 2

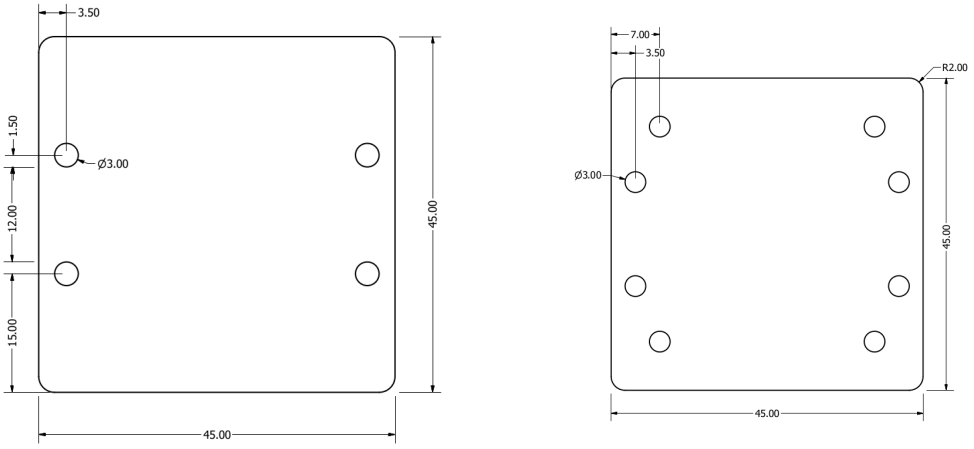


FIGURE A.3: Baseplates used to clamp the camera

Appendix B

Onboard computer

The Onboard computer which is used for the experiments is shown in the Figure B.1. The specifications of the onboard computer can be found in Table [103]. It has Quad-core, 4-plus-1 ARM Cortex-A15 MPcore Processor with Neon technology.

FIGURE B.1: Manilfold(Onboard computer)

Weight	197g
Dimension	110m x 110m x 26 mm
Memory	2GB DDR3L system RAM 16GB eMMC 4.51 storage
Processors	Quadcore, 4-plus-1 arm cortex
Input Voltage	14v to 26v
Power Consumption	5w to 15w
Operating Temperature	$-10\,^{\circ}\mathrm{C}$ $45\,^{\circ}\mathrm{C}$

TABLE B.1: Manifold specification (onboard computer) used to implement all the algorithms

Appendix C

Software

The hardware and software used in this thesis are explained below,

Software:

- Ubuntu 14.04 Operating system
- Kernel version:4.4.0-131-generic
- Robotics Operating System (ROS) Indigo (version)
- MATLAB R2016b

The terminology used in this thesis in the context of ROS are explained here,

- rosbag : Tool for recording and playing the wanted topics.
- messages: Simple file contains the data of the particular topic that is already defined.

Hardware platform:

- Laptop Platform:Intel(R) Core(TM) i7-4600U CPU @ 2.10GHz, 8GB RAM @ 1600MHz
- Onboard computer : Quad-core, 4-plus-1 ARM Cortex-A15 MPcore Processor with Neon technology.
- kernel on the onboard computer: Debian kernel 3.1.0

Appendix D

Preliminaries

This chapter gives on overview of the coordinate transformations used in this work.

D.1 Coordinate systems

A point in the space and its displacement can be represented through the coordinate system . Since this work deals with the real time data collection from the environment through stereo camera mounted on the small UAV. Hence the transformation of the one coordinate system to the other coordinate systems have been done here.

D.2 Coordinate frames transformations

In general the position and orientation of the real world points are represented in the global frame i.e world coordinate frame W. Camera frame is represented as C. Besides, that the odometry of the camera is denoted in the odometry frame O. We have used the transformation library [104] in ROS. Detailed information about the conventions and notions of the coordinate system can be found in [105].

Combination of both the translation and rotation forms the transformation of one coordinate frame to the other.

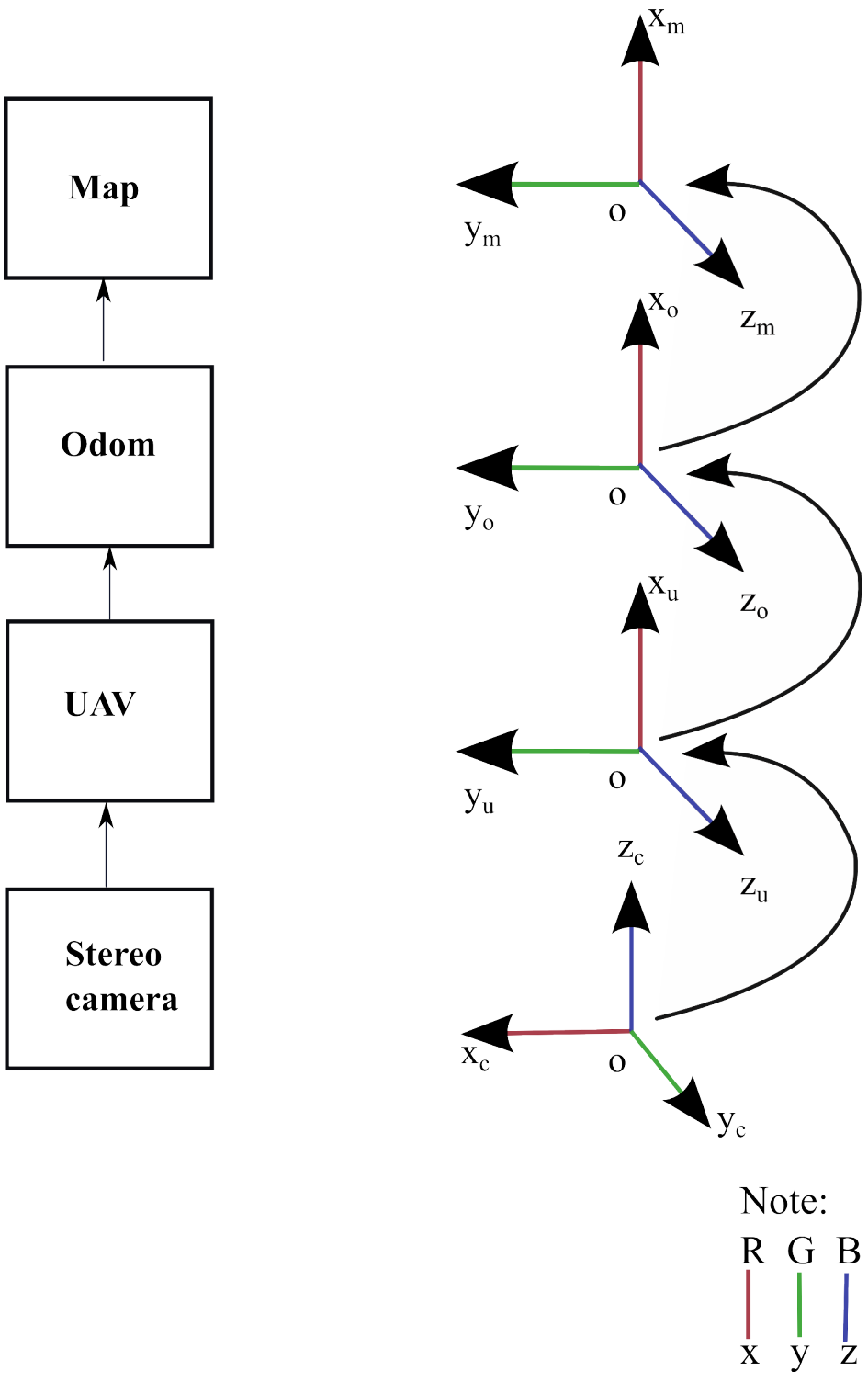


FIGURE D.1: Overview of the all the coordinate frame tree starting from stereo camera, UAV, odom, Map

Bibliography

[1] Pwc. Clarity from above pwc global report on the commercial applications of drone technology, 2016. URL https://www.pwc.pl/clarityfromabove.

[2] *International Organization for Standardization. Robots and robotic devices vocabulary ISO 8373:2012 (International Organization for Standardization, 2012).*, 2012.

[3] A. Censi, E. Mueller, E. Frazzoli, and S. Soatto. A power-performance approach to comparing sensor families, with application to comparing neuromorphic to traditional vision sensors. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3319–3326, May 2015. doi: 10.1109/ICRA.2015.7139657.

[4] A. Martinelli. Vision and imu data fusion: Closed-form solutions for attitude, speed, absolute scale, and bias determination. *IEEE Transactions on Robotics*, 28 (1):44–60, Feb 2012. ISSN 1552-3098. doi: 10.1109/TRO.2011.2160468.

[5] U. Papa and G. Del Core. Design of sonar sensor model for safe landing of an uav. In *2015 IEEE Metrology for Aerospace (MetroAeroSpace)*, pages 346–350, June 2015. doi: 10.1109/MetroAeroSpace.2015.7180680.

[6] N. Gageik, P. Benz, and S. Montenegro. Obstacle detection and collision avoidance for a uav with complementary low-cost sensors. *IEEE Access*, 3:599–609, 2015. doi: 10.1109/ACCESS.2015.2432455.

[7] Filip Mroz and Toby P Breckon. An empirical comparison of real-time dense stereo approaches for use in the automotive environment. *EURASIP Journal on Image and Video Processing*, 2012(1):13, Aug 2012. ISSN 1687-5281. doi: 10.1186/1687-5281-2012-13.

[8] M. Jalobeanu, G. Shirakyan, G. Parent, H. Kikkeri, B. Peasley, and A. Feniello. Reliable kinect-based navigation in large indoor environments. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 495–502, May 2015. doi: 10.1109/ICRA.2015.7139225.

[9] N. Jeong, H. Hwang, and E. T. Matson. Evaluation of low-cost lidar sensor for application in indoor uav navigation. In *2018 IEEE Sensors Applications Symposium (SAS)*, pages 1–5, March 2018. doi: 10.1109/SAS.2018.8336719.

[10] Christoforos Kanellakis and George Nikolakopoulos. Survey on computer vision for uavs: Current developments and trends. *Journal of Intelligent & Robotic Systems*, 87(1):141–168, Jul 2017. ISSN 1573-0409. doi: 10.1007/s10846-017-0483-z. URL https://doi.org/10.1007/s10846-017-0483-z.

[11] Naoki Suganuma and Takaaki Kubo. Fast dynamic object extraction using stereovision based on occupancy grid maps and optical flow. *2011 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, pages 978–983, 2011.

[12] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J.J. Leonard. Past, present, and future of simultaneous localization and mapping: Towards the robust-perception age. *IEEE Transactions on Robotics*, 32(6): 13091332, 2016.

[13] Stefano Soatto. Steps towards a theory of visual information: Active perception, signal-to-symbol conversion and the interplay between sensing and control. *CoRR*, abs/1110.2053, 2011. URL http://arxiv.org/abs/1110.2053.

[14] Yali Li, Shengjin Wang, Qi Tian, and Xiaoqing Ding. A survey of recent advances in visual feature detection. *Neurocomputing*, 149:736 – 751, 2015. ISSN 0925-2312. doi: https://doi.org/10.1016/j.neucom.2014.08.003. URL http://www.sciencedirect.com/science/article/pii/S0925231214010121.

[15] David G Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *Int. J. Comput. Vision*, 60(2):91–110, November 2004. ISSN 0920-5691. doi: 10.1023/B:VISI.0000029664.99615.94. URL http://dx.doi.org/10.1023/B:VISI.0000029664.99615.94.

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

[17] Michael Calonder, Vincent Lepetit, Christoph Strecha, and Pascal Fua. Brief: Binary robust independent elementary features. In *Proceedings of the 11th European Conference on Computer Vision: Part IV*, ECCV'10, pages 778–792, Berlin, Heidelberg, 2010. Springer-Verlag. ISBN 3-642-15560-X, 978-3-642-15560-4. URL http://dl.acm.org/citation.cfm?id=1888089.1888148.

[18] E. Rosten, R. Porter, and T. Drummond. Faster and better: A machine learning approach to corner detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(1):105–119, Jan 2010. ISSN 0162-8828. doi: 10.1109/TPAMI.2008.275.

[19] Henrik Aanæs, Anders Lindbjerg Dahl, and Kim Steenstrup Pedersen. Interesting interest points. *International Journal of Computer Vision*, 97(1):18–35, Mar 2012. ISSN 1573-1405. doi: 10.1007/s11263-011-0473-8. URL https://doi.org/10.1007/s11263-011-0473-8.

[20] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. Orb: An efficient alternative to sift or surf. In *2011 International Conference on Computer Vision*, pages 2564–2571, Nov 2011. doi: 10.1109/ICCV.2011.6126544.

[21] Stefan Leutenegger, Margarita Chli, and Roland Y. Siegwart. Brisk: Binary robust invariant scalable keypoints. In *Proceedings of the 2011 International Conference on Computer Vision*, ICCV '11, pages 2548–2555, Washington, DC, USA, 2011. IEEE Computer Society. ISBN 978-1-4577-1101-5. doi: 10.1109/ICCV.2011.6126542. URL http://dx.doi.org/10.1109/ICCV.2011.6126542.

[22] A. Alahi, R. Ortiz, and P. Vandergheynst. Freak: Fast retina keypoint. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 510–517, June 2012. doi: 10.1109/CVPR.2012.6247715.

[23] Pablo Fernández Alcantarilla, Adrien Bartoli, and Andrew J. Davison. Kaze features. In Andrew Fitzgibbon, Svetlana Lazebnik, Pietro Perona, Yoichi Sato, and Cordelia Schmid, editors, *Computer Vision – ECCV 2012*, pages 214–227, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg. ISBN 978-3-642-33783-3.

[24] Jianbo Shi and C. Tomasi. Good features to track. In *1994 Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 593–600, Jun 1994. doi: 10.1109/CVPR.1994.323794.

[25] D. Scaramuzza and F. Fraundorfer. Visual odometry [tutorial]. *IEEE Robotics Automation Magazine*, 18(4):80–92, Dec 2011. ISSN 1070-9932. doi: 10.1109/MRA.2011.943233.

[26] A. Milella and R. Siegwart. Stereo-based ego-motion estimation using pixel tracking and iterative closest point. In *Fourth IEEE International Conference on Computer Vision Systems (ICVS'06)*, pages 21–21, Jan 2006. doi: 10.1109/ICVS.2006.56.

[27] Jonathan Kelly and Gaurav S. Sukhatme. An experimental study of aerial stereo visual odometry. *IFAC Proceedings Volumes*, 40(15):197 – 202, 2007. ISSN 1474-6670. doi: https://doi.org/10.3182/20070903-3-FR-2921.00036. URL http://www.sciencedirect.com/science/article/pii/S1474667016346614. 6th IFAC Symposium on Intelligent Autonomous Vehicles.

[28] Jonathan Kelly and Gaurav S. Sukhatme. An experimental study of aerial stereo visual odometry. *IFAC Proceedings Volumes*, 40(15):197 – 202, 2007. ISSN 1474-6670. doi: https://doi.org/10.3182/20070903-3-FR-2921.00036. URL http://www.sciencedirect.com/science/article/pii/S1474667016346614. 6th IFAC Symposium on Intelligent Autonomous Vehicles.

[29] Jianke Zhu. Image gradient-based joint direct visual odometry for stereo camera. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, IJCAI'17, pages 4558–4564. AAAI Press, 2017. ISBN 978-0-9992411-0-3. URL http://dl.acm.org/citation.cfm?id=3171837.3171924.

[30] J. Witt and U. Weltin. Robust stereo visual odometry using iterative closest multiple lines. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4164–4171, Nov 2013. doi: 10.1109/IROS.2013.6696953.

[31] Maimone Mark, Cheng Yang, and Matthies Larry. Two years of visual odometry on the mars exploration rovers. *Journal of Field Robotics*, 24(3):169–186. doi: 10.1002/rob.20184. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/rob.20184.

[32] A. E. Johnson, S. B. Goldberg, Yang Cheng, and L. H. Matthies. Robust and efficient stereo feature tracking for visual odometry. In *2008 IEEE International Conference on Robotics and Automation*, pages 39–46, May 2008. doi: 10.1109/ROBOT.2008.4543184.

[33] R. Gomez-Ojeda and J. Gonzalez-Jimenez. Robust stereo visual odometry through a probabilistic combination of points and line segments. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2521–2526, May 2016. doi: 10.1109/ICRA.2016.7487406.

[34] D. Nister. An efficient solution to the five-point relative pose problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(6):756–770, June 2004. ISSN 0162-8828. doi: 10.1109/TPAMI.2004.17.

[35] M. Tomono. Robust 3d slam with a stereo camera based on an edge-point icp algorithm. In *2009 IEEE International Conference on Robotics and Automation*, pages 4306–4311, May 2009. doi: 10.1109/ROBOT.2009.5152529.

[36] A. I. Comport, E. Malis, and P. Rives. Accurate quadrifocal tracking for robust 3d visual odometry. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 40–45, April 2007. doi: 10.1109/ROBOT.2007.363762.

[37] B. Kitt, F. Moosmann, and C. Stiller. Moving on to dynamic environments: Visual odometry using feature classification. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5551–5556, Oct 2010. doi: 10.1109/IROS.2010.5650517.

[38] H. Durrant-Whyte and T. Bailey. Simultaneous localization and mapping: part i. *IEEE Robotics Automation Magazine*, 13(2):99–110, June 2006. ISSN 1070-9932. doi: 10.1109/MRA.2006.1638022.

[39] T. Bailey and H. Durrant-Whyte. Simultaneous localization and mapping (slam): part ii. *IEEE Robotics Automation Magazine*, 13(3):108–117, Sept 2006. ISSN 1070-9932. doi: 10.1109/MRA.2006.1678144.

[40] Cesar Cadena, Luca Carlone, Henry Carrillo, Yasir Latif, Davide Scaramuzza, José Neira, Ian D. Reid, and John J. Leonard. Simultaneous localization and mapping: Present, future, and the robust-perception age. *CoRR*, abs/1606.05830, 2016. URL http://arxiv.org/abs/1606.05830.

[41] Kuka Robotics. Kuka Navigation Solution. http://www.kuka-robotics.com/res/robotics/Products/PDF/EN/KUKA_Navigation_Solution_EN.pdf, 2016.

[42] E. Ackerman. Dyson's Robot Vacuum Has 360-Degree Camera, Tank Treads, Cyclone Suction. *IEEE Spectrum*, September 2014.

[43] M. Maimone, Y. Cheng, and L. Matthies. Two years of Visual Odometry on the Mars Exploration Rovers. *Journal of Field Robotics*, 24(3):169–186, 2007.

[44] F. Lu and E. Milios. Globally consistent range scan alignment for environment mapping. 4(4):333–349, 1997.

[45] Adam Harmat, Michael Trentini, and Inna Sharf. Multi-camera tracking and mapping for unmanned aerial vehicles in unstructured environments. *Journal of Intelligent & Robotic Systems*, 78(2):291–317, May 2015. ISSN 1573-0409. doi: 10.1007/s10846-014-0085-y. URL https://doi.org/10.1007/s10846-014-0085-y.

[46] R. C. Leishman, T. W. McLain, and R. W. Beard. Relative navigation approach for vision-based aerial gps-denied navigation. In *2013 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 343–352, May 2013. doi: 10.1109/ICUAS.2013.6564707.

[47] Jos A. Castellanos, Jos Neira, and Juan D. Tards. Limits to the consistency of ekf-based slam 1.

[48] J.A. Castellanos, R. Martinez-Cantin, J.D. Tards, and J. Neira. Robocentric map joining: Improving the consistency of ekf-slam. *Robotics and Autonomous Systems*, 55(1):21 – 29, 2007. ISSN 0921-8890. doi: http://dx.doi.org/10.1016/j.robot.2006.06.005. URL //www.sciencedirect.com/science/article/pii/S0921889006001448. Simultaneous Localisation and Map Building.

[49] Shoudong Huang and G. Dissanayake. Convergence analysis for extended kalman filter based slam. In *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, pages 412–417, May 2006. doi: 10.1109/ROBOT.2006.1641746.

[50] Luca Marchetti, Giorgio Grisetti, and Luca Iocchi. *A Comparative Analysis of Particle Filter Based Localization Methods*, pages 442–449. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007. ISBN 978-3-540-74024-7. doi: 10.1007/978-3-540-74024-7_44. URL http://dx.doi.org/10.1007/978-3-540-74024-7_44.

[51] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, 2005. ISBN 0262201623.

[52] R. Martinez-Cantin, N. de Freitas, and J. A. Castellanos. Analysis of particle methods for simultaneous robot localization and mapping and a new algorithm: Marginal-slam. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 2415–2420, April 2007. doi: 10.1109/ROBOT.2007.363681.

[53] J. Folkesson and H. Christensen. Graphical slam - a self-correcting map. In *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004*, volume 1, pages 383–390 Vol.1, April 2004. doi: 10.1109/ROBOT.2004.1307180.

[54] Tom Duckett, Stephen Marsland, and Jonathan Shapiro. Fast, on-line learning of globally consistent maps. *Autonomous Robots*, 12(3):287–300, May 2002. ISSN 1573-7527. doi: 10.1023/A:1015269615729. URL https://doi.org/10.1023/A:1015269615729.

[55] Michael Kaess, Hordur Johannsson, Richard Roberts, Viorela Ila, John J Leonard, and Frank Dellaert. isam2: Incremental smoothing and mapping using the bayes tree. *The International Journal of Robotics Research*, 31(2):216–235, 2012. doi: 10.1177/0278364911430419. URL https://doi.org/10.1177/0278364911430419.

[56] Takafumi Taketomi, Hideaki Uchiyama, and Sei Ikeda. Visual slam algorithms: a survey from 2010 to 2016. *IPSJ Transactions on Computer Vision and Applications*, 9(1):16, Jun 2017. ISSN 1882-6695. doi: 10.1186/s41074-017-0027-2. URL https://doi.org/10.1186/s41074-017-0027-2.

[57] A. I. Mourikis and S. I. Roumeliotis. A multi-state constraint kalman filter for vision-aided inertial navigation. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 3565–3572, April 2007. doi: 10.1109/ROBOT.2007.364024.

[58] Simon Lynen, Torsten Sattler, Michael Bosse, Joel A. Hesch, Marc Pollefeys, and Roland Siegwart. Get out of my lab: Large-scale, real-time visual-inertial localization. In *Robotics: Science and Systems*, 2015.

[59] Gabe Sibley, Christopher Mei, Ian Reid, and Paul Newman. Vast-scale outdoor navigation using adaptive relative bundle adjustment. *The International Journal of Robotics Research*, 29(8):958–980, 2010. doi: 10.1177/0278364910369268. URL https://doi.org/10.1177/0278364910369268.

[60] G. Schindler, M. Brown, and R. Szeliski. City-scale location recognition. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–7, June 2007. doi: 10.1109/CVPR.2007.383150.

[61] S. Lowry, N. Snderhauf, P. Newman, J. J. Leonard, D. Cox, P. Corke, and M. J. Milford. Visual place recognition: A survey. *IEEE Transactions on Robotics*, 32 (1):1–19, Feb 2016. ISSN 1552-3098. doi: 10.1109/TRO.2015.2496823.

[62] Jana Koeck, Fayin Li, and Xialong Yang. Global localization and relative positioning based on scale-invariant keypoints. *Robotics and Autonomous Systems*, 52(1):27 – 38, 2005. ISSN 0921-8890. doi: https://doi.org/10.1016/j.robot.2005.03.008. URL http://www.sciencedirect.com/science/article/pii/S092188900500062X. Advances in Robot Vision.

[63] J. Sivic and A. Zisserman. Video google: a text retrieval approach to object matching in videos. In *Proceedings Ninth IEEE International Conference on Computer Vision*, pages 1470–1477 vol.2, Oct 2003. doi: 10.1109/ICCV.2003.1238663.

[64] Fei-Fei Li and Pietro Perona. A bayesian hierarchical model for learning natural scene categories. In *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 2 - Volume 02*, CVPR '05, pages 524–531, Washington, DC, USA, 2005. IEEE Computer Society. ISBN 0-7695-2372-2. doi: 10.1109/CVPR.2005.16. URL http://dx.doi.org/10.1109/CVPR.2005.16.

[65] Hans P. Moravec. Robot spatial perception by stereoscopic vision and 3d evidence grids. Technical report, 1996.

[66] Brian Curless and Marc Levoy. A volumetric method for building complex models from range images. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '96, pages 303–312, New York, NY, USA, 1996. ACM. ISBN 0-89791-746-4. doi: 10.1145/237170.237269. URL http://doi.acm.org/10.1145/237170.237269.

[67] Nchter Andreas, Lingemann Kai, Hertzberg Joachim, and Surmann Hartmut. 6d slam3d mapping outdoor environments. *Journal of Field Robotics*, 24(89):699–722. doi: 10.1002/rob.20209. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/rob.20209.

[68] D. Murray and C. Jennings. Stereo vision based mapping and navigation for mobile robots. In *Proceedings of International Conference on Robotics and Automation*, volume 2, pages 1694–1699 vol.2, Apr 1997. doi: 10.1109/ROBOT.1997.614387.

[69] Kurt Konolige, Motilal Agrawal, Robert C. Bolles, Cregg Cowan, Martin Fischler, and Brian Gerkey. *Outdoor Mapping and Navigation Using Stereo Vision*, pages 179–190. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008. ISBN 978-3-540-77457-0. doi: 10.1007/978-3-540-77457-0_17. URL https://doi.org/10.1007/978-3-540-77457-0_17.

[70] Armin Hornung, Kai M. Wurm, Maren Bennewitz, Cyrill Stachniss, and Wolfram Burgard. Octomap: an efficient probabilistic 3d mapping framework based on octrees. *Auton. Robots*, 34:189–206, 2013.

[71] Fei Yan, Yi-Sha Liu, and Ji-Zhong Xiao. Path planning in complex 3d environments using a probabilistic roadmap method. *International Journal of Automation and Computing*, 10(6):525–533, Dec 2013. ISSN 1751-8520. doi: 10.1007/s11633-013-0750-9. URL https://doi.org/10.1007/s11633-013-0750-9.

[72] F. Schler. *3d path planning for autonomous aerial vehicles in constrained spaces*. PhD dissertation, Aalborg University, Aalborg, Denmark, 2012.

[73] S. M. LaValle. *Planning Algorithms*. Cambridge University Press, Cambridge, U.K., 2006. Available at http://planning.cs.uiuc.edu/.

[74] Howie Choset, Kevin M. Lynch, Seth Hutchinson, George A. Kantor, Wolfram Burgard, Lydia E. Kavraki, and Sebastian Thrun. *Principles of Robot Motion: Theory, Algorithms, and Implementations*. MIT Press, Cambridge, MA, June 2005.

[75] Enric Galceran and Marc Carreras. A survey on coverage path planning for robotics. *Robot. Auton. Syst.*, 61(12):1258–1276, December 2013. ISSN 0921-8890. doi: 10.1016/j.robot.2013.09.004. URL http://dx.doi.org/10.1016/j.robot.2013.09.004.

[76] Luca De Filippis, Giorgio Guglieri, and Fulvia Quagliotti. Path planning strategies for uavs in 3d environments. *Journal of Intelligent & Robotic Systems*, 65(1):247–264, Jan 2012. ISSN 1573-0409. doi: 10.1007/s10846-011-9568-2. URL https://doi.org/10.1007/s10846-011-9568-2.

[77] Liang Yang, Juntong Qi, Dalei Song, Jizhong Xiao, Jianda Han, and Yong Xia. Survey of robot 3d path planning algorithms. *J. Control Sci. Eng.*, 2016:5–, March 2016. ISSN 1687-5249. doi: 10.1155/2016/7426913. URL http://dx.doi.org/10.1155/2016/7426913.

[78] L. E. Kavraki, P. Svestka, J. C. Latombe, and M. H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, 12(4):566–580, Aug 1996. ISSN 1042-296X. doi: 10.1109/70.508439.

[79] Steven M. Lavalle. Rapidly-exploring random trees: A new tool for path planning. 1998.

[80] B. Miller, K. Stepanyan, A. Miller, and M. Andreev. 3d path planning in a threat environment. In *2011 50th IEEE Conference on Decision and Control and European Control Conference*, pages 6864–6869, Dec 2011. doi: 10.1109/CDC.2011.6160385.

[81] Daniel Meyer-Delius, Maximilian Beinhofer, and Wolfram Burgard. Occupancy grid models for robot mapping in changing environments. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence*, AAAI'12, pages 2024–2030. AAAI Press, 2012. URL http://dl.acm.org/citation.cfm?id=2900929.2901014.

[82] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numer. Math.*, 1(1):269–271, December 1959. ISSN 0029-599X. doi: 10.1007/BF01386390. URL http://dx.doi.org/10.1007/BF01386390.

[83] S. Koenig and M. Likhachev. Improved fast replanning for robot navigation in unknown terrain. In *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No.02CH37292)*, volume 1, pages 968–975 vol.1, 2002. doi: 10.1109/ROBOT.2002.1013481.

[84] Anthony Stentz. The focussed d* algorithm for real-time replanning. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'95, pages 1652–1659, San Francisco, CA, USA, 1995. Morgan Kaufmann Publishers Inc. ISBN 1-55860-363-8. URL http://dl.acm.org/citation.cfm?id=1643031.1643113.

[85] A. Stentz. Optimal and efficient path planning for partially-known environments. In *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*, pages 3310–3317 vol.4, May 1994. doi: 10.1109/ROBOT.1994.351061.

[86] R. Yue, J. Xiao, S. Wang, and S. L. Joseph. Modeling and path planning of the city-climber robot part ii: 3d path planning using mixed integer linear programming. In *2009 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 2391–2396, Dec 2009. doi: 10.1109/ROBIO.2009.5420650.

[87] E. Masehian and G. Habibi. Modeling and path planning of the city-climber robot part ii: 3d path planning using mixed integer linear programming. In *International journal journal of Mechanical system science and Engineering*, volume 23, pages 26–31, 2007.

[88] A. Chamseddine, Y. Zhang, C. A. Rabbath, C. Join, and D. Theilliol. Flatness-based trajectory planning/replanning for a quadrotor unmanned aerial vehicle. *IEEE Transactions on Aerospace and Electronic Systems*, 48(4):2832–2848, October 2012. ISSN 0018-9251. doi: 10.1109/TAES.2012.6324664.

[89] John H. Holland. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*. MIT Press, Cambridge, MA, USA, 1992. ISBN 0262082136.

[90] Pablo Moscato and Michael G. Norman. A "memetic" approach for the traveling salesman problem implementation of a computational ecology for combinatorial optimization on message-passing systems. In *IN PROCEEDINGS OF THE INTERNATIONAL CONFERENCE ON PARALLEL COMPUTING AND TRANSPUTER APPLICATIONS*, pages 177–186. IOS Press, 1992.

[91] M. Dorigo, V. Maniezzo, and A. Colorni. Ant system: optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 26(1):29–41, Feb 1996. ISSN 1083-4419. doi: 10.1109/3477.484436.

[92] M. Antunes, J. P. Barreto, C. Premebida, and U. Nunes. Can stereo vision replace a laser rangefinder? In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5183–5190, Oct 2012. doi: 10.1109/IROS.2012.6385844.

[93] Peter Corke. *Robotics, Vision and Control: Fundamental Algorithms In MATLAB, Second Edition*. Springer Publishing Company, Incorporated, 2nd edition, 2017. ISBN 3319544128, 9783319544120.

[94] Y. M. Wang, Y. Li, and J. B. Zheng. A camera calibration technique based on opencv. In *The 3rd International Conference on Information Sciences and Interaction Sciences*, pages 403–406, June 2010. doi: 10.1109/ICICIS.2010.5534797.

[95] S. Thrun and M. Montemerlo. The GraphSLAM algorithm with applications to large-scale mapping of urban structures. *International Journal on Robotics Research*, 25(5/6):403–430, 2005.

[96] G. Grisetti, R. Kummerle, C. Stachniss, and W. Burgard. A tutorial on graph-based slam. *IEEE Intelligent Transportation Systems Magazine*, 2(4):31–43, winter 2010. ISSN 1939-1390. doi: 10.1109/MITS.2010.939925.

[97] G. Grisetti, R. Kmmerle, C. Stachniss, U. Frese, and C. Hertzberg. Hierarchical optimization on manifolds for online 2d and 3d mapping. In *2010 IEEE International Conference on Robotics and Automation*, pages 273–278, May 2010. doi: 10.1109/ROBOT.2010.5509407.

[98] R. Kuemmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard. g2o: A general framework for graph optimization. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 3607–3613, Shanghai, China, May 2011. doi: 10.1109/ICRA.2011.5979949.

[99] Andreas Geiger, Julius Ziegler, and Christoph Stiller. Stereoscan: Dense 3d reconstruction in real-time. In *Intelligent Vehicles Symposium (IV)*, 2011.

[100] A. Neubeck and L. Van Gool. Efficient non-maximum suppression. In *18th International Conference on Pattern Recognition (ICPR'06)*, volume 3, pages 850–855, 2006. doi: 10.1109/ICPR.2006.479.

[101] Sven Koenig, Maxim Likhachev, Yaxin Liu, and David Furcy. Incremental heuristic search in ai. *AI Mag.*, 25(2):99–112, June 2004. ISSN 0738-4602. URL http://dl.acm.org/citation.cfm?id=1017133.1017140.

[102] Moore Neighbor connectvity. http://www.imageprocessingplace.com/downloads_V3/root_downloads/tutorials/contour_tracing_Abeer_George_Ghuneim/mmain.html. Accessed: 2016-10-28.

[103] DJI. Manifold user manual, 2018. URL https://dl.djicdn.com/downloads/manifold/en/Manifold_User_Manual__en__v1.0.pdf.

[104] TF (ROS). coordinate systems transfromations and rotations, 2018. URL http://wiki.ros.org/tf/Overview/Transformations.

[105] John J. Craig. *Introduction to Robotics: Mechanics and Control*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2nd edition, 1989. ISBN 0201095289.