Doctoral Thesis

Flexible Configuration Stereo Vision using Aerial Robots

March 2024

Doctoral Program in Advanced
Mechanical Engineering and Robotics
Graduate School of Science and
Engineering Ritsumeikan University

Borwonpob Sumetheeprasit

Doctoral Thesis Reviewed
by Ritsumeikan University

Flexible Configuration Stereo Vision using Aerial Robots
（飛行ロボットを用いた可変構成ステレオビジョン）

March 2024

2024年3月

Doctoral Program in Advanced
Mechanical Engineering and Robotics
Graduate School of Science and Engineering
Ritsumeikan University
立命館大学大学院理工学研究科
機械システム専攻博士課程後期課程

Borwonpob Sumetheeprasit
ボーウォンポブ　スメーチープラシット

Supervisor：Professor SHIMONOMURA Kazuhiro
研究指導教員：下ノ村　和弘　教授

Acknowledgment

Firstly, I would like to express my utmost appreciation to the advisor of the Integrated Sensors and Intelligence laboratory, Professor Kazuhiro Shimonomura, and the former assistant professor, Professor Robert Ladig, for providing the advice, guidance, equipment and support over the past five years since my entry as a member of the laboratory that enables the development and completion of this work. The research activities during my time in this laboratory could not have been made possible without their support.

Secondly, I would like to express my gratitude for the members and colleagues in the laboratory for their active roles in supporting my research activities. The experience as a member of the laboratory, both on campus and off campus, has been made a special memory because of my colleagues. I strongly believe that their support in countless aspects has greatly enhanced my ability and initiative as a researcher, as a student, and in the future as a member of the scientific society.

Finally and most warmly, I would like to express my warmest appreciation and love for my family who has always been supporting my dream and my decision to study abroad and carried out my research activities in Japan. I would not have been here and this experience would not have been possible without their support.

Abstract

This work develops a rapid and movement efficient 3D mapping method using multiple aerial robots. The method used for 3D reconstruction is flexible configuration stereo vision. In contrast to traditional stereo vision where the two cameras of the stereo camera are fixed in a predefined position, flexible configuration stereo distributes the viewpoints of the stereo camera independently. The distribution of viewpoints allows the adjustment of stereo vision configuration, including baseline distance, horizontal or vertical axis, and adjustment of inward tilt angle. The viewpoints of the flexible configuration stereo in the proposed system are distributed on separate multirotor aerial robots. The mobility of the multirotor platform facilitates the adjustment of stereo configuration. In this work, three prototype implementations are developed using different tracking methods. The first prototype utilizes two visible range monocular cameras on gimbals for pose tracking of each aerial robot. The second prototype utilizes the infrared range stereo camera in combination with odometry for pose tracking. The final system utilizes custom made prototype aerial robots and inside-out tracking and sensors for relative pose measurement. Additionally, this work proposed two configuration point cloud fusion algorithms, their performance has been verified on both simulation and physical experiments. The work has been shown to create an accurate long-range map creation of the range up to 400 meters with the movement of only 10 meters in the simulation with the total operation time of less than two minutes. In an outdoor experiment, the system has been shown to create a map of the range of up to 100 meters with 20 meters of movement and less than three minutes of the total operation time.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Research background

The development of radio-controlled aircraft opens the possibility of flights without the risk to the human pilot otherwise needed onboard. Unmanned aerial vehicles (UAVs) have their origin in the 20th century and began their emergence in the US military as the radio-controlled munition platforms [1]. Several years later in the United Kingdom, one of the first pilotless aircraft modified from the airframe of De Havilland DH 82 Tiger Mouth named the Queen Bee, later nicknamed Drone became the beginning of the name used today to refer to the UAVs [2]. The development of UAVs for various functions in military use has continued into the 21st century. In recent years, the effectiveness of UAVs for reconnaissance tasks has become increasingly apparent. In the civilian sector, the use of UAV has been increasingly surging. Various industries have been using the UAV platform for a wide range of applications, including remote sensing [3], agriculture [4], aerial manipulation [5], and the newly emerging race to commercialize air mobility platform [6].

In the field of remote sensing multiple platforms are used for different scales and types of applications. In 3D mapping tasks, UAVs are used for high complexity scenes with the scene size of a few meters up to a few kilometers [3]. The types of UAV include fixed-wing traditional takeoff and landing UAVs, vertical take-off and landing (VTOL) UAVs, and the hybrid fixed-wing VTOL utilizing the advantage of both [7].

Among the categories, multirotor UAVs are one of the most popular platforms among industries, researchers, and consumers alike [8]. The mobility of the multirotor to be able to move in 6-DOF without the need to bank in order to turn, the small size which allows them to operate in small to medium sized areas, and the cost-effectiveness of such platforms allows them to be suitable for various kinds of remote sensing tasks.

Various kinds of sensors have been utilized onboard multirotors for 3D mapping tasks, such as visible spectrum cameras, infrared cameras, light detection and ranging (LiDAR), or depth cameras. The resultant 3D maps are obtained from the fusion of aforementioned sensors with the localization data from the multirotor, such as GNSS, inertial measurement unit (IMU), or external localization devices. The following mapping methods used in such tasks includes:

- photogrammetry and structure from motion (SfM) based methods utilizing monocular camera [9] [10] [11] [12]
- light detection and ranging (LiDAR) based method [13] [14][15]
- visual or visual-inertial simultaneous localization mapping (SLAM) [16] [17] [18]

1.2 Motivation and objectives

Using flexible configuration stereo vision as the mapping methods provides multiple advantages over the use of methods mentioned in chapter 1. The characteristic and limitations of the mentioned methods are as follows:

- SfM based method produces medium to high resolution 3D maps from the large number of geo-tagged images retrieved from each checkpoint from careful path planning [9]. In photogrammetry, a down-facing camera is often used meaning that the multirotor needs to be at a considerable altitude over the object being mapped.
- LiDAR based method produces high accuracy and high resolution 3D maps. Sensors capable of far range (more than 100 m) are often large and heavy,

being a limiting factor of the types of multirotor used and their flight time. Additionally, LiDAR sensors are usually acquirable at high cost [19].

- Visual SLAM based method provides low to medium density map while also providing localization data [20]. SLAM is extremely effective in an indoor or closed environment. The robustness of SLAM methods relies heavily on the frame rate of the camera used, especially in a high vibration system such as a multirotor. Furthermore, the range limitation of SLAM is based on the range of the sensor used, for example the range of the depth camera used.

- Additionally, the common limitations of mentioned methods include the need to fly over the whole area of interest in order to map in order to cover the area, the computation of the final map being processed after the flight meaning lack of real-time capability, and their limited range.

The first limitation of mentioned methods is the range of the system compared to the amount of movement needed to achieve it. The range of the system is limited by the range of the sensor utilized in each method. For example, a visual inertial SLAM based on depth camera method is limited by the range and accuracy of the depth camera. The commonly used depth cameras such as Intel Realsense D435 depth cameras have their estimated effective range of 10 meters and their maximum field of view of 90 degrees. With this limiting factor, in order to cover an area with a wide range, the system needs to move a significant distance to cover the whole area. Additionally, the UAV with the depth camera has to enter and get close to the area in order to map it. This may cause safety problems when used in a dynamically active environment such as in a bustling city with many vehicles, or in a disaster-stricken area where the situation is actively developing. On the other hand, SfM based methods such as photogrammetry can cover a wide area when high altitude is used, but in turn the resolution of 3D mapping would decrease proportionate to the altitude used. However, the limitation of down-facing a camera is the mapping of tall structures which requires significant altitude. For multirotor UAVs which have limited flight time, the amount of movement per area covered is extremely crucial.

Our proposed system has been shown to map an area with a distance up to 400 meters with only 10 meters of movement in simulation at the altitude of 20 meters.

The second limitation the proposed system aims to overcome is the real-time capability. SfM based methods lack the ability to provide sustained depth information in real-time, since depth is estimated from the movement of the UAV. For Visual SLAM and LiDAR based methods, real-time depth estimation is only provided within the range of the sensor used. The advantage of using stereo vision is the ability to provide sustained depth information in real-time. By changing the baseline distance, the depth estimation range can be extended such that a large area can be monitored in real-time. Additionally, if more than two viewpoints are used, multiple ranges can be monitored at the same time [21]. For instance, a short baseline can be used to monitor near field objects while a large baseline can be used for far field objects.

1.3 Outline

The concept of the main 3D reconstruction method, flexible configuration stereo vision, as well as its advantages in comparison to traditional stereo vision and other existing methods is described in Chapter 2. In Chapter 3, the simulation showing the characteristic of flexible configuration stereo vision is shown along with the conclusion drawn from the simulation result as a foundation to construct the proposed point cloud fusion algorithm and horizontal and vertical setup fusion algorithm. Chapter 4, Chapter 5, and Chapter 6 describe the implementation of the proposed system physically, based on three different tracking approaches and the lesson learned from each implementation used for the succeeding implementation. Finally, in Chapter 7, the summary of the thesis as well as the future work and improvement proposals are discussed.

Chapter 2

Concept of flexible configuration stereo vision

2.1 Introduction to stereo vision

The 3D mapping system utilizing stereo vision as the 3D reconstruction method is proposed. According to Bradski [22], stereo vision is the use of two viewpoints from a pair of often identical cameras, traditionally visible spectrum cameras. Each camera is placed next to each other horizontally at a specified distance, the distance between the two cameras is called baseline distance. The left and right cameras are faced in the same direction such that the image plane of the cameras are in a parallel plane. In the two images of the scene obtained from the same instance taken by both cameras are called image pairs. Objects reflected in the image of the left camera seem to dispace compared to the same object reflected in the right camera. Objects that are close to the camera appear to displace further than far objects. This displacement is called disparity. The disparity of each pixel is calculated by stereo matching algorithms, which match the features from each frame and calculate their disparity. The resulting disparity of all pixels are represented in the form of a disparity image, with each pixel containing the amount of disparity in pixels. The disparity image is then reprojected into a 3D point cloud using the information such as the relative position of the camera, and intrinsic parameters of the camera such as

focal length to calculate the 3D position of each pixel in the real 3D space. Figure 2.1 shows the geometry of the depth estimation using stereo vision and Figure 2.2 shows the overall process of stereo vision depth estimation.

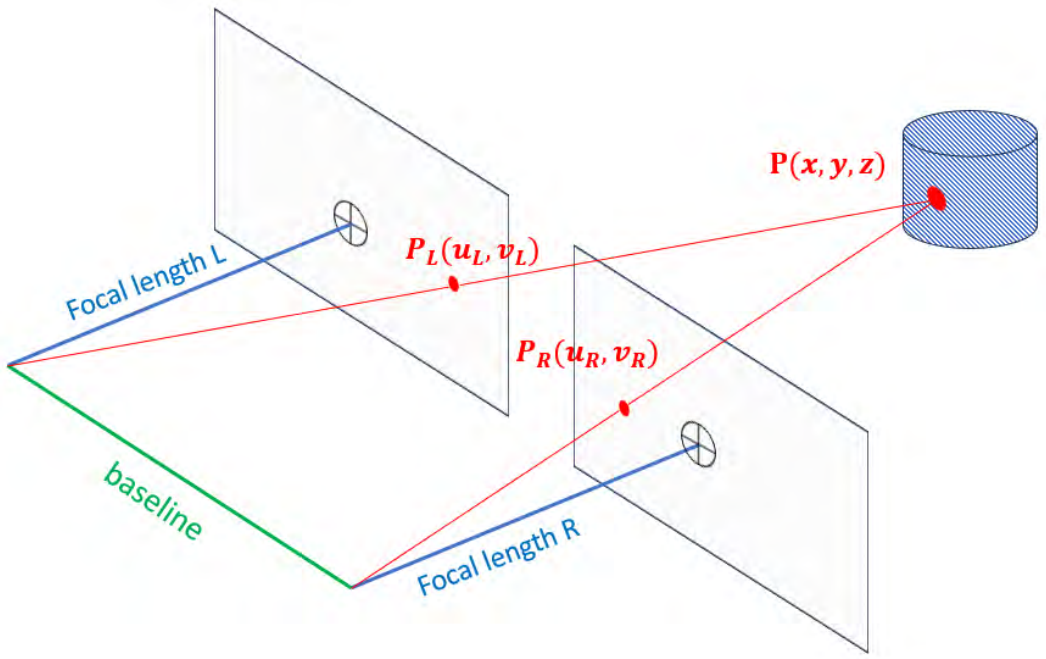

Figure 2.1: Geometry of stereo vision depth estimation

In traditional stereo cameras, the position and orientation of the left and right cameras are fixed, such that the geometry transformation of the cameras are always constant and known. Camera parameters are acquired from the camera calibration process [23]. Normally, a pattern such as a checker board or similar pattern with known physical dimensions is shown in front of the two cameras. The stereo camera parameters are then calculated from the appearance of the pattern in the image . The important parameters in stereo vision that need to be acquired before use are called intrinsic parameters, and extrinsic parameters. Intrinsic parameters describe the optical characteristic of an individual camera. Equation 2.1 shows the two intrinsic matrices, K camera matrix, and Equation 2.2 shows D distortion matrix. Camera matrix is composed of focal length f in pixels, horizontal center of image c_x in pixels, and vertical center of the image c_y in pixels. To simplify, focal length

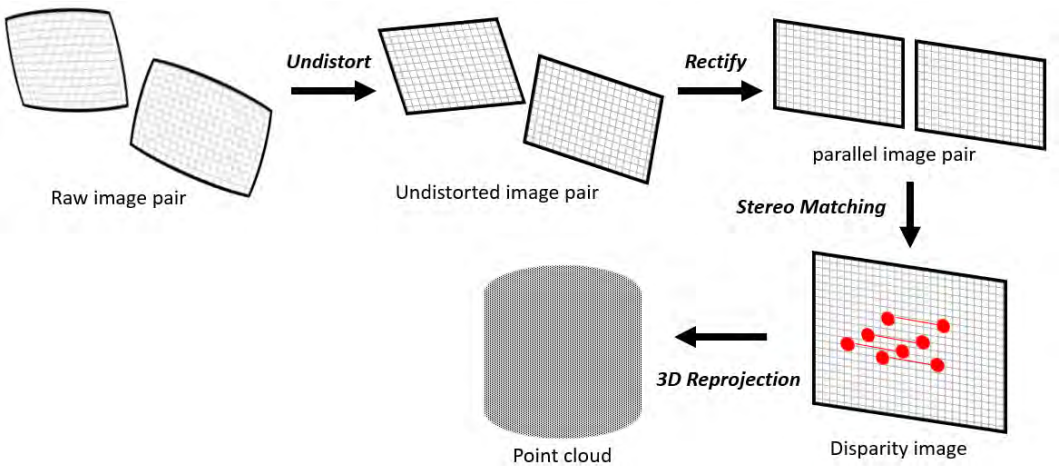

Figure 2.2: Flowchart representing processes in stereo vision depth estimation

of both horizontal and vertical axis of the image plane are assumed to be the same. Distortion D matrices used in this case are composed of tangential distortion d_{pn} and radial distortion coefficients d_{kn} describing the distortion of the image appearing in the camera. Each set of intrinsic matrices exists for each camera, therefore the final intrinsic matrices are K_1, K_2, D_1, and D_2.

$$K_1 = \begin{bmatrix} f_1 & 0 & c_{x_1} \\ 0 & f_1 & c_{y_1} \\ 0 & 0 & 1 \end{bmatrix}$$
$$K_2 = \begin{bmatrix} f_2 & 0 & c_{x_2} \\ 0 & f_2 & c_{y_2} \\ 0 & 0 & 1 \end{bmatrix} \tag{2.1}$$

$$D_1 = \begin{bmatrix} d_{k1_1} & d_{k2_1} & d_{p1_1} & d_{p2_1} & d_{k3_1} \end{bmatrix}$$
$$D_2 = \begin{bmatrix} d_{k1_2} & d_{k2_2} & d_{p1_2} & d_{p2_2} & d_{k3_2} \end{bmatrix} \tag{2.2}$$

Extrinsic matrices are described in equation 2.3 composing of a reprojection matrix Q and rectification matrices R_1, R_2. Extrinsic matrices describe the physical

relationship between the two cameras, including relative position and orientation. Q is used for reprojection of disparity images into a 3D point cloud. The calculation of reprojection matrices are information from the intrinsic matrices from the main camera (typically left camera) and the baseline distance b in meters. Rectification matrices R_1 and R_2 are used to fix the unparalleled orientation of the camera by rotating the image virtually to match the relative orientation of the camera in reality.

$$
\begin{aligned}
Q &= \begin{bmatrix} 1 & 0 & 0 & -c_x \\ 0 & 1 & 0 & -c_y \\ 0 & 0 & 0 & f \\ 0 & 0 & -1/b & 0 \end{bmatrix} \\
R_1 &= \begin{bmatrix} r_{1_{11}} & r_{1_{12}} & r_{1_{13}} \\ r_{1_{21}} & r_{1_{22}} & r_{1_{23}} \\ r_{1_{31}} & r_{1_{32}} & r_{1_{33}} \end{bmatrix} \\
R_2 &= \begin{bmatrix} r_{2_{11}} & r_{2_{12}} & r_{2_{13}} \\ r_{2_{21}} & r_{2_{22}} & r_{2_{23}} \\ r_{2_{31}} & r_{2_{32}} & r_{2_{33}} \end{bmatrix}
\end{aligned}
\tag{2.3}
$$

The pixel position of each feature u, v is projected to the image plane in the 3D space using:

$$
\begin{aligned}
\begin{bmatrix} x_L \\ y_L \\ 1 \end{bmatrix} &= {K_1}^{-1} \begin{bmatrix} u_L \\ v_L \\ 1 \end{bmatrix} \\
\begin{bmatrix} x_R \\ y_R \\ 1 \end{bmatrix} &= {K_2}^{-1} \begin{bmatrix} u_R \\ v_R \\ 1 \end{bmatrix}
\end{aligned}
\tag{2.4}
$$

3D position of each pixel can be estimated by solving geometry calculation using:

$$\begin{aligned} x &= \frac{b(x_L + x_R)}{2d} \\ y &= \frac{by_L}{d} \\ z &= \frac{bf}{d} \end{aligned} \tag{2.5}$$

where b is baseline distance, f is focal length, and d is disparity calculated using stereo matching algorithms and represented in the form of disparity image with the same dimension as the image from the left camera.

The Bouguet's algorithm [24] to calculate the rectification matrices for both cameras are as follows:

1. Compose e_1 using normalized translation vector from camera 1 to camera 2 using:

$$e_1 = \frac{T}{|T|} \tag{2.6}$$

   where T is the 3D translation vector from camera 1 to camera 2

$$T = \begin{bmatrix} T_x \\ T_y \\ T_z \end{bmatrix} \tag{2.7}$$

2. Compose e_2 using the following equation:

$$e_2 = \frac{\begin{bmatrix} -T_y & T_x & 0 \end{bmatrix}^T}{\sqrt{T_x^2 + T_y^2}} \tag{2.8}$$

3. Compose e_3 using the following equation:

$$e_3 = e_1 \times e_2 \tag{2.9}$$

4. Compose a shard rectification matrix R_{rect} using the following equation:

$$R_{rect} = \begin{bmatrix} e_1^T \\ e_2^T \\ e_3^T \end{bmatrix} \tag{2.10}$$

5. Calculate each individual camera's rectification matrix using:

$$\begin{aligned} R_1 &= R_{rect}\ r_1 \\ R_2 &= R_{rect}\ r_2 \end{aligned} \tag{2.11}$$

where r_1 and r_2 is the relative orientation of each camera w.r.t. an origin frame.

2.2 Limitations of stereo vision

The principle limitations of stereo vision have challenged the use of stereo vision as a 3D reconstruction method for 3D mapping. A fixed viewpoint system where two cameras of the stereo vision system are fixed at a specific baseline and axis is referred to as traditional stereo vision [25]. In traditional stereo vision, the principal limitation is the depth estimation error that increases quadratically to the target distance. Depth estimation error is estimated using the following equation from [26]:

$$\epsilon_z = \frac{z^2}{bf} \cdot \epsilon_d \tag{2.12}$$

Where ϵ_z is the depth estimation error in meters, z is the target distance in meters, b is the baseline distance in meters, f is the focal length in pixels, and ϵ_d is the disparity error in pixels. Estimation error ϵ_z is observed to increase at the power of two of the target distance z when other variables are constant. Figure 2.3 illustrates the estimation error of a stereo vision system with different baseline distances. The error of the depth estimation is estimated to be less than or equal to the colored lines plotted from equation 2.12. The error of the stereo vision system with wider baseline

can be seen to be smaller than narrower baselines. A conferred effect from another viewpoint is that, the confidence of the depth estimation of a given distance is higher when larger baseline is used. Therefore, with this information, logically using a large baseline is more effective since the estimation error can be kept lower.

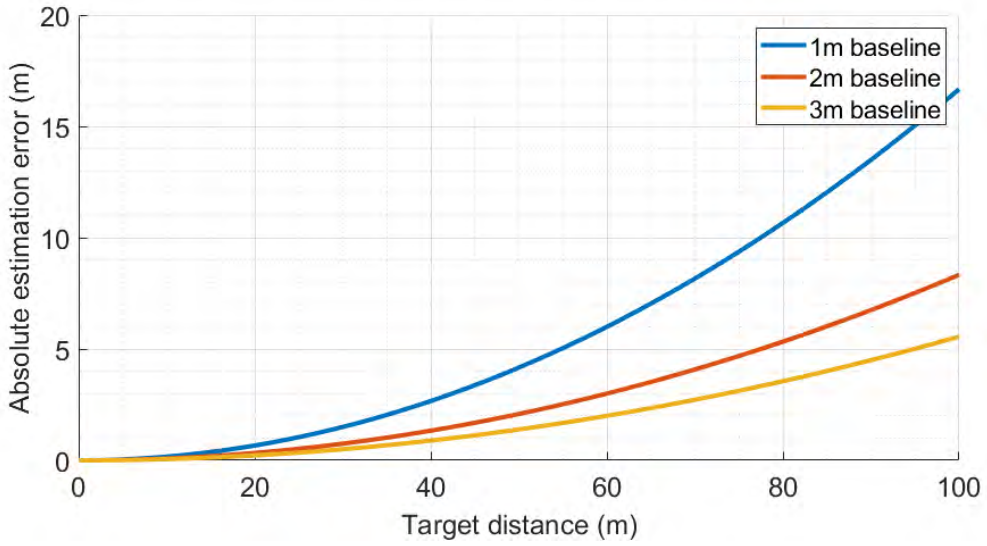

Figure 2.3: Depth estimation error (x) vs target depth (y) using three different baselines.

Having said that, in practice a larger baseline, even though it reduces the estimation error, it promotes occlusion [27]. When the distance of baseline used is too wide, occlusion occurs to areas close to the camera such that stereo matching becomes difficult. The completeness and correctness of disparity images produced from stereo matching with said baseline is significantly reduced. This phenomenon is called by this work as an excessive baseline. Figure 2.4 shows the effects of using excessive baseline. Noise is generated in the disparity image as false matches are computed by the stereo matching algorithm. When projected into a point cloud, the noise directly creates fuzziness in the point cloud resulting in an incorrect point cloud.

Finally, another limitation of stereo vision is occlusion as an effect from the geometry of viewpoints. Several kinds of occlusions are considered including:

1. Occlusion from non-overlapped field of view (FOV) area.

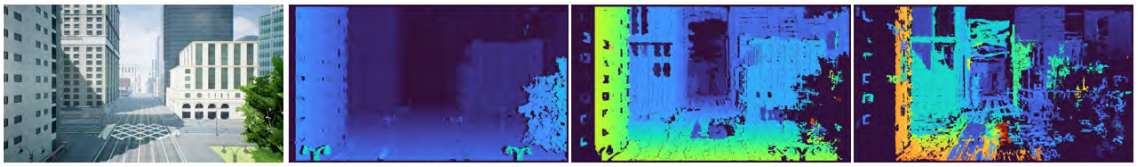

Figure 2.4: Image of (a) reference frame and disparity images of (b) 2 meters baseline, (c) 6 meters baseline, and (d) 12 meters baseline.

2. Occlusion behind objects.

Figure 2.5 shows the geometries of the cameras that create non-overlapped areas. In traditional stereo vision, stereo matching of objects that are closer than the closest intersection of the two cameras' field of view is difficult [28]. Therefore, the closer range cannot be reconstructed, especially when a wide baseline or narrow field cameras are used and such non-overlapped area becomes larger. Figure 2.6 shows the effects of occlusion from non-overlapped areas in disparity images. In horizontal stereo, the non-overlapped area can be observed in the left side of the disparity images. The area grows larger as a larger baseline is used. This phenomenon limits the FOV of the stereo cameras with large baseline or narrow FOV. On the other hand, if an extremely wide FOV lens is used, the non-overlapped area can be reduced. However, the accuracy of depth estimation is also reduced due to the distortion of the lenses. Additionally, since the pixel-per-angle ratio increases in wide FOV lenses compared to narrow FOV lenses, the depth estimation becomes much more inaccurate [29]. Figure 2.7 shows the effects of using wide FOV where pixel-per-angle becomes a problem.

Occlusion behind objects is also a major issue in stereo vision. Since the camera cannot see behind the object [30], depth estimation by stereo vision alone cannot cover these areas [31]. Figure 2.8 shows the geometric illustration of the area behind objects. The shadow projected from the object by the camera shows the blind area in disparity images, the disparity values of these areas cannot be calculated. Usually, in order to cover these blind spots, depth estimation from stereo vision is combined in a fusion algorithm like visual SLAM such as [18] while the whole stereo system moves around the object in order to cover all the area around it. Alternatively, multiple

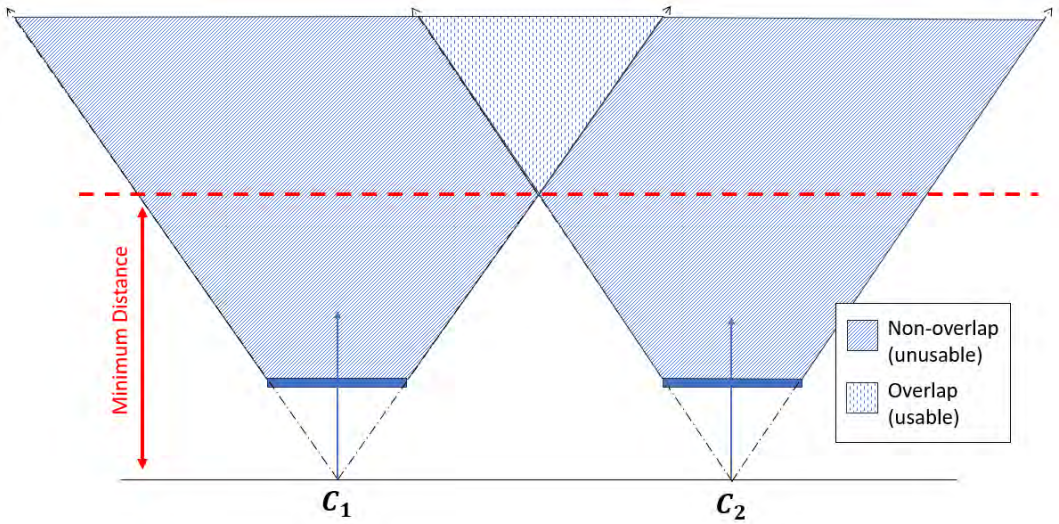

Figure 2.5: Geometry of overlapped area in stereo cameras.

viewpoints [32] can be used in the system such that all the viewpoints cover the whole area around the object.

2.3 Advantages of flexible configuration stereo

In this work, the main concept of the system is to overcome the limitations of traditional stereo vision. The method to achieve this is called by this work as flexible configuration stereo vision [33]. In contrast to traditional stereo vision where two viewpoints of the stereo vision system are fixed in place, in flexible configuration stereo the viewpoints are distributed. The two viewpoints can be freely moved in 6-DOF such that the relative pose of the two viewpoints, namely the configuration of the stereo vision, can be flexibly adjusted. In this work, the adjustment of three parameters of the stereo configuration are explored including, baseline distance, horizontal or vertical setup, and inward tilt angle.

The adjustment of baseline length, called variable baseline stereo [25], provides a means to overcome the principal limitation of stereo vision, the estimation error per target distance limitation described in section 2.2. The baseline of the stereo vision system can be adjusted to fit the scale and distance of the mapping area in order to

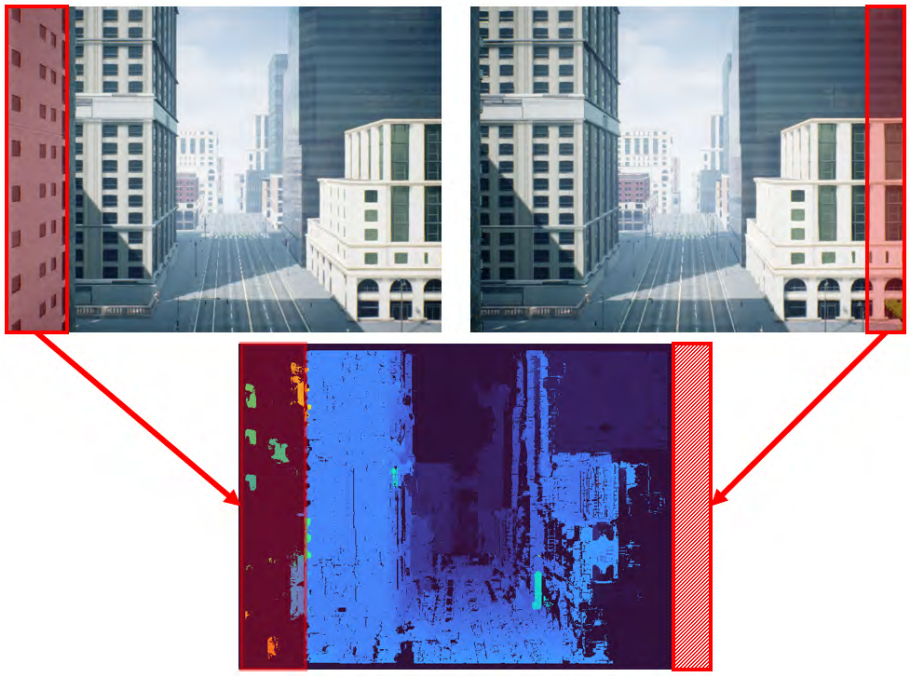

Figure 2.6: Effects of non-overlapped area in computed disparity.

minimize error. Additionally, the baseline can be reduced to an optimal baseline to balance the minimum estimation error while keeping the baseline distance from being too excessive. Furthermore, multiple baseline distances can be used to map a single area [34]. The use of multiple baselines can extend the range of the mapping system to cover a wider range of distance.

Selectability between horizontal stereo and vertical stereo is essential for completeness of the resultant map. Since horizontal and vertical stereo has its own effectiveness in each situation, such as in terms of occlusion axis and shapes of objects. By being able to switch between the two setups, an ideal setup can be chosen for different kinds of scenes. Furthermore, a combination of depth estimation from horizontal and vertical stereo has shown improvement in depth estimation completeness as well as accuracy [35]. In a simulation experiment in chapter 3 has proven the advantages in using each configuration as well as the combination of both.

The inward tilt angle of the cameras, namely the inward tilt of the yaw angle of the horizontal setup, or the inward tilt of the pitch angle of the vertical setup

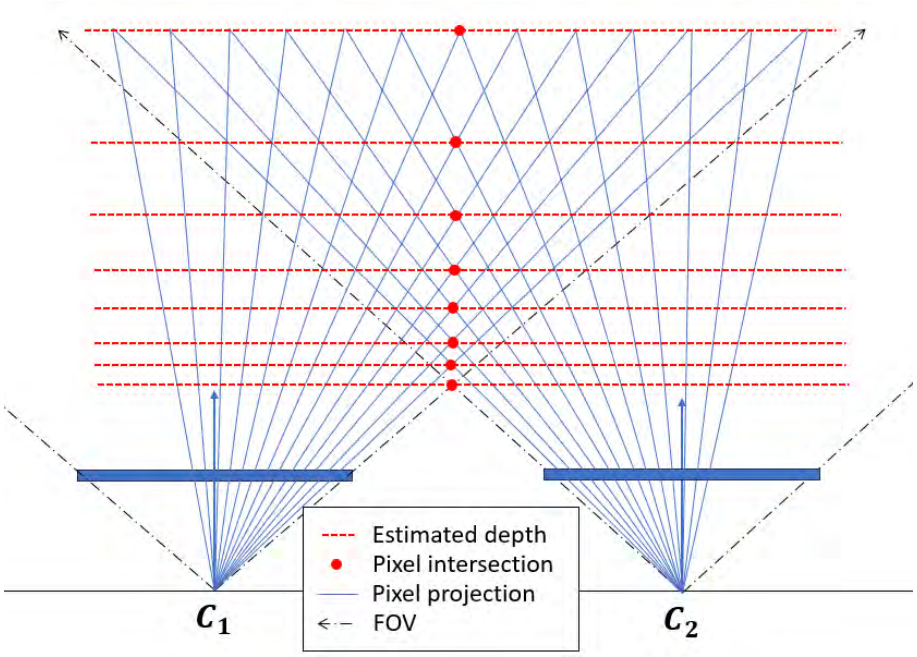

Figure 2.7: Ray projection from the center of the camera to each of its pixels and their estimated depth.

can be adjusted in this work. The advantage of being able to adjust the inward tilt angle is the reduction of the non-overlapped area of the two cameras' field of view [36]. The non-overlapped area on the movement axis of the camera can be reduced. Furthermore, when a large baseline is used, inward tilt can be applied in order to improve the completeness of depth estimation of near field objects. In chapter 3, the simulation experiment result has shown that by applying inward tilt, stereo matching of near field objects can be seen improving. By applying this, the range of a large baseline can be extended to include near field objects as well, and reduce the effects of noise generated by excessive baseline.

Finally, in order to realize flexible configuration stereo vision, the viewpoints are distributed on each different UAV. In a two viewpoints system, two multirotor UAVs are used with left and right viewpoints on two different drones. The movement of viewpoints is greatly facilitated by the multirotor UAVs, due to their ability to move in 6-DOF and ability to hover at a fixed position and level orientation, contrast to

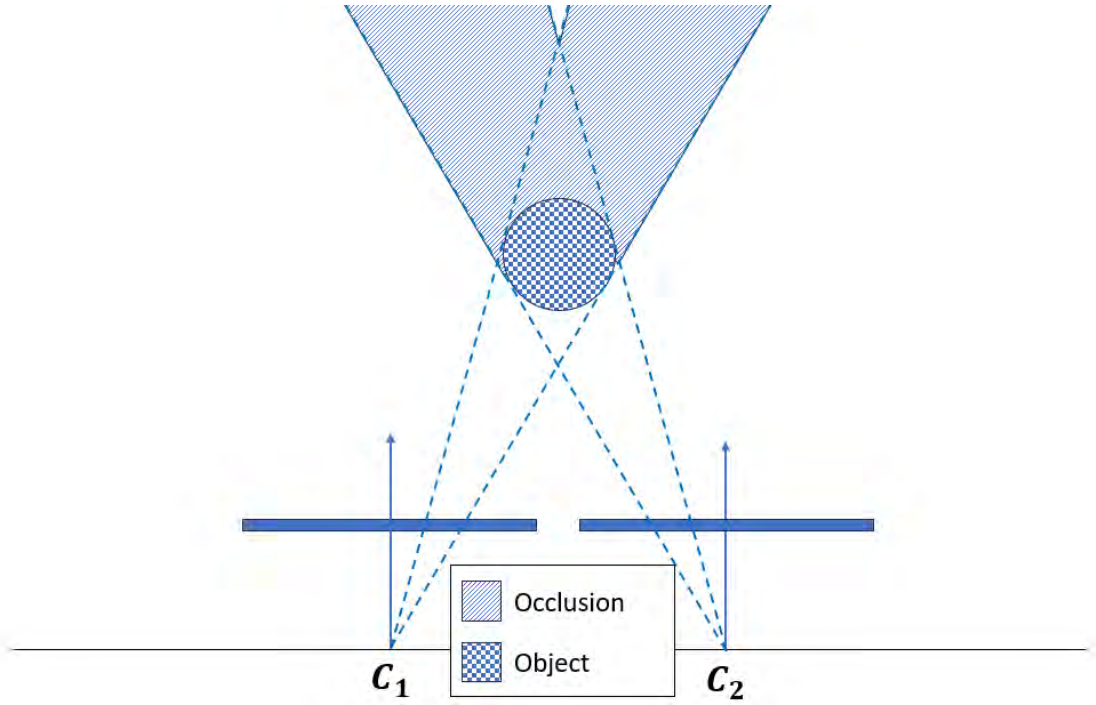

Figure 2.8: Occlusion ray projected from the two cameras caused by an object.

fixed-wing UAVs which need to maintain airspeed. This especially holds true when large baselines, such as more than 3 meters, where otherwise the difficulty persists in positioning of the viewpoints that has to be done by unmanned ground vehicles (UGVs) [37] or manually moved by human operators. Additionally, by increasing the number of viewpoints, for example three UAVs are used as three viewpoints, multiple configurations can be utilized simultaneously. The 3D reconstruction from each configuration can be combined into a more accurate and higher completeness 3D map, as explored by [38]. In chapter 3, the utilization of multiple configurations and its advantage is discussed.

2.4 Challenges in the design

Although stereo vision provides several advantages over existing methods, several challenges are to be addressed in order to achieve effective depth estimation. The first and likely the most crucial point in the design is the means to accurately obtain

relative pose of all the cameras in real-time, including both position and orientation. The accuracy relative pose of the cameras directly affects the efficacy of stereo matching as well as the accuracy of depth estimation. Relative orientation of the cameras is used in the image rectification process in order to align the images obtained from non-parallel cameras to align in the same plane. Since stereo matching methods look for the features mainly in the same epipolar line, the accuracy of ratification to align the features on the same plane is crucial for the stereo matching in order to find a matching feature on both frames. Furthermore, even if the features are matched, the error in rectification results in the error computed disparity values, therefore the accuracy of depth estimation. Especially, for long range depth estimation where the error of only one pixel could mean a large proportion of error in depth estimation. Relative position of the cameras are used in the point cloud reprojection process where the disparity image is projected into a point cloud. The measurement of relative position to align the two cameras such that they align on the same axis affects the efficacy of the stereo matching where rectification cannot fix. Furthermore, the accuracy of baseline distance affects the calculation of depth estimation directly. Hence, the accuracy of relative pose measurement is extremely crucial for the overall success of the system.

The second challenge is tracking and localization of each individual UAV. In order to achieve high accuracy and high frequency relative pose, each of the UAVs' own localization is used to aid the calculation of relative pose. The pose odometry obtained from localization is coupled to the images obtained from the cameras are coupled based on timestamp. Furthermore, pose obtained from localization is used to control the UAVs to position itself at the correct desired position and orientation. The position of the drones to be correct at the time of image acquisition is extremely important because error is extremely difficult to fix using software algorithms afterward. Additionally, when multiple baselines are used and the point cloud projected from each baseline is fused into a resultant point cloud, an external point of reference outside of the drone is required. The points needs to be rotated and translated such that all point clouds are in the same coordinate system. In contrast, if only the rela-

tive position of the two drones are used, if the reference drone moves between taking of the images at each baselines, the movement and the relative position at each baseline would be unknown without a localization method. The merging of point clouds without the external reference point would result in the point clouds being in different coordinates. Therefore, an accurate, high frequency, and low latency tracking method is required.

The third challenge is the robustness of the stereo matching algorithm in case of not perfectly rectified images. In the case of the proposed system, the chance is very unlikely that stereo image pairs obtained from the UAV are perfectly rectified. Several unpredictable factors, such as tracking drift, error in pose calculation, or camera vibration can cause the image to be slightly misaligned. Therefore, a stereo matching algorithm that can effectively match the features even if the images are perfectly rectified is needed. In case of this work, three stereo matching algorithms, Block Matching (BM), Semi-global Block Matching (SGBM) [39] and Quasi-Dense Stereo Matching [40] are evaluated. Figure 2.9 shows the performance of each stereo matching algorithm processing the same non-parallel image pairs. Quasi-dense matching algorithm shows significantly better performance than the other two with the cost of much more processing time. Since the completeness of the depth estimation is prioritized in this work, Quasi-Dense stereo matching algorithm is used in this work as the primary matching algorithm.

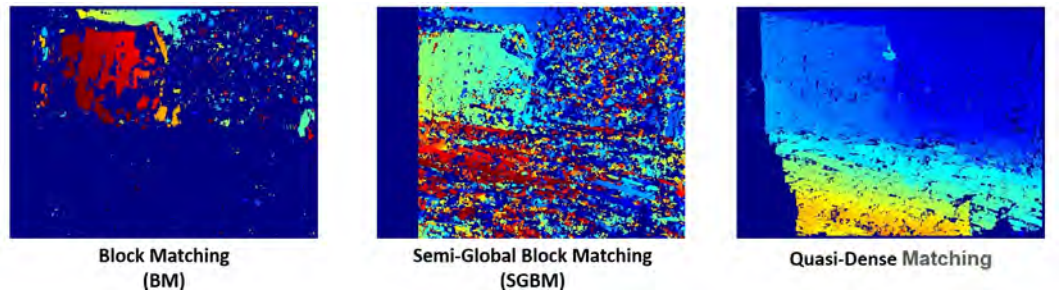

Figure 2.9: Comparison of Block Matching (BM), Semi-global Block Matching (SGBM) and Quasi-Dense Stereo Matching

The last challenge is the effective simultaneous use of multiple baseline distances.

As mentioned in section 2.2, the range that a baseline distance is capable of reliably providing depth information is limited. When too small a baseline is used, the estimation error grows, on the other hand, if an excessive baseline is used stereo matching would result in a noisy image. Therefore, a baseline fusion algorithm is developed to determine the estimated effective range of a baseline distance. The baseline fusion algorithm is described in detail in section 3.2.

2.5 System applications

An example use case is using the system as a reconnaissance and guidance system for another robot. The UAVs in the system act as the mapping agent, creating a rough map of an area of operation. After getting an approximate idea of the area of operation's 3D map, another robot such as an AGV or another UAV can navigate the area without the need to map the area by itself. Figure 2.10 shows the illustrated use of the idea. A ground robot can navigate a maze using the proposed system to create a map of the maze that is not in the line-of-sight of the ground robot. Furthermore, Figure 2.11 shows the utilization of the real-time mapping capability of the proposed system. The real-time mapping capability can aid a ground robot in a dynamically changing environment by determining the obstacles in the path. An obstacle can be detected from the map, and a new path can be promptly plotted in such a case.

The real-world scenario application is the use of the system to map a disaster-stricken area where the situation is unpredictable and time is of essence, but directly sending in a rescue robot is deemed risky. The proposed system can create an approximate map of the area to determine a safe path and continue to guide the rescue robot from a distance to the target. Another use case is the map creation of a metropolis where there are a large number of vehicles and pedestrians in the area such that sending in a UAV into the area would be dangerous. The proposed system can create a long range map from a distance without the need to get into the city and without the need to fly at high altitude which would disrupt the operation of aircrafts.

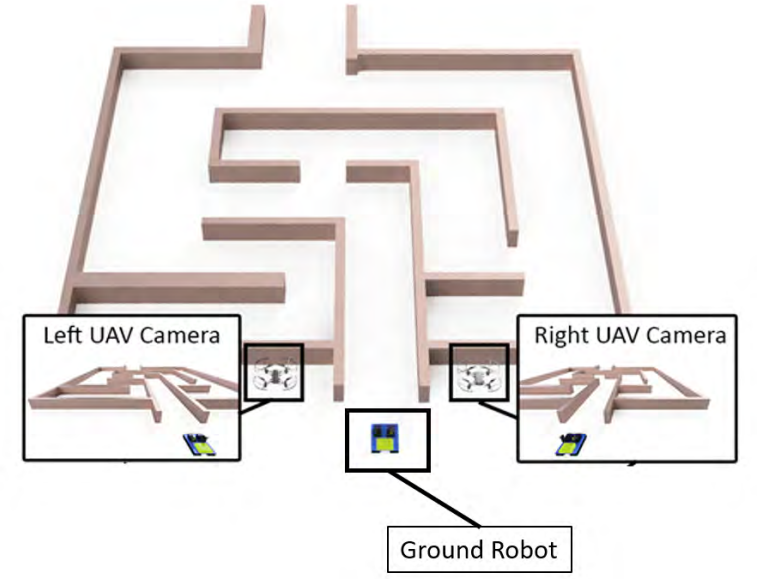

Figure 2.10: Two UAVs as mapping agents guiding a ground robot through a maze.

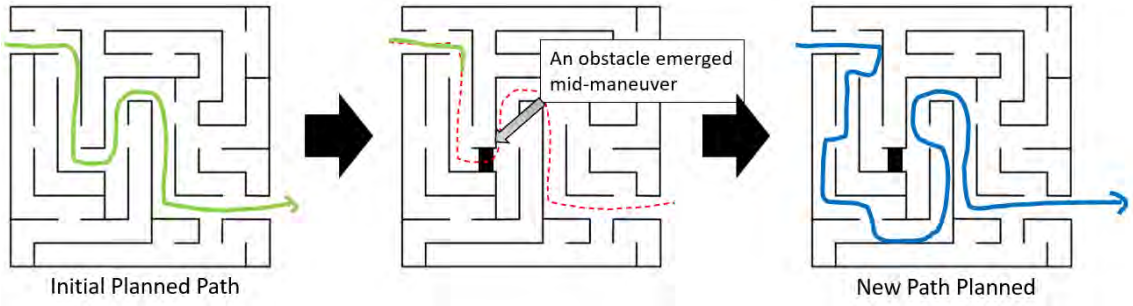

Figure 2.11: Use case idea of real-time path planning with the real-time map creation capability.

Chapter 3

Algorithm and simulations

3.1 Characteristic of variable baseline

In order to explore and build understanding on the characteristics of flexible configuration stereo, various simulation scenarios are conducted. The simulations are conducted in a realistic Unreal Engine based simulation software AirSim [41].

The first simulation involves the evaluation of depth estimation for increasing baseline distance. The images are taken in several image pairs each with different baseline distance, and their corresponding disparity image and point clouds are evaluated. In Airsim, the camera is moved to 6 different baseline distances from 3 to 8 meters in a mater increment step. Their corresponding image pairs are retrieved. Figure 3.1 shows the retrieved images. The obtained images from the simulation have no distortion and since the image pairs are perfectly parallel, the undistortion and rectification steps can be skipped.

Figure 3.2 shows the disparity image calculated by Quasi-Dense stereo matching algorithm corresponding to each baseline. The disparity has been filtered to be in the range of 11 to 256 pixels. The effects of increasing baseline can be seen in the disparity images. The first point that can be noticed in the disparity is the non-overlap area on the right side of the disparity image that increases as the baseline increases. The second point that can be noticed is the deterioration of disparity image of the near field objects as the baseline increases, but the improvement can be seen for far field

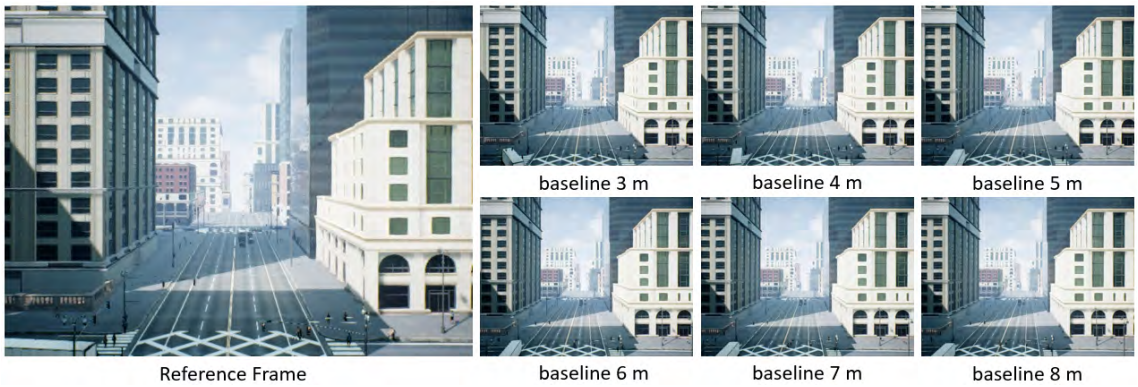

Figure 3.1: Color image obtained from AirSim of 6 different baseline distances.

objects.

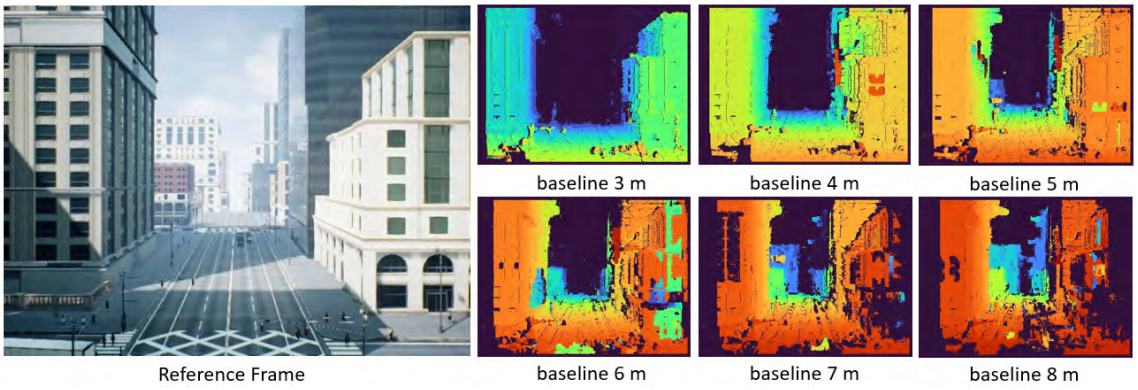

Figure 3.2: Disparity image obtained from AirSim of 6 different baseline distances.

Figure 3.3 shows the point cloud projected from each baseline. The first effect of baseline increment, from the top-down view, the range of depth estimation can be seen extending as the baseline increases. For larger baselines, buildings that are far away can be seen. The second effect that can be seen is the distance between each row of depth estimation. The distance between the rows at smaller baselines at any given range is much larger than the larger baselines. The distance between each row is the estimation error per pixel given by equation 2.12. This is the visualization of the depth estimation error which falls in the range of equation 2.12 as the upper bound. In our published work [42], this effect has been proven with the data. The third effect is the accuracy and precision of the depth estimation. For close objects

with small baselines, the object's depth estimation can be seen larger than the ground truth. Their depth estimation approaches the correct value as the baseline increases.

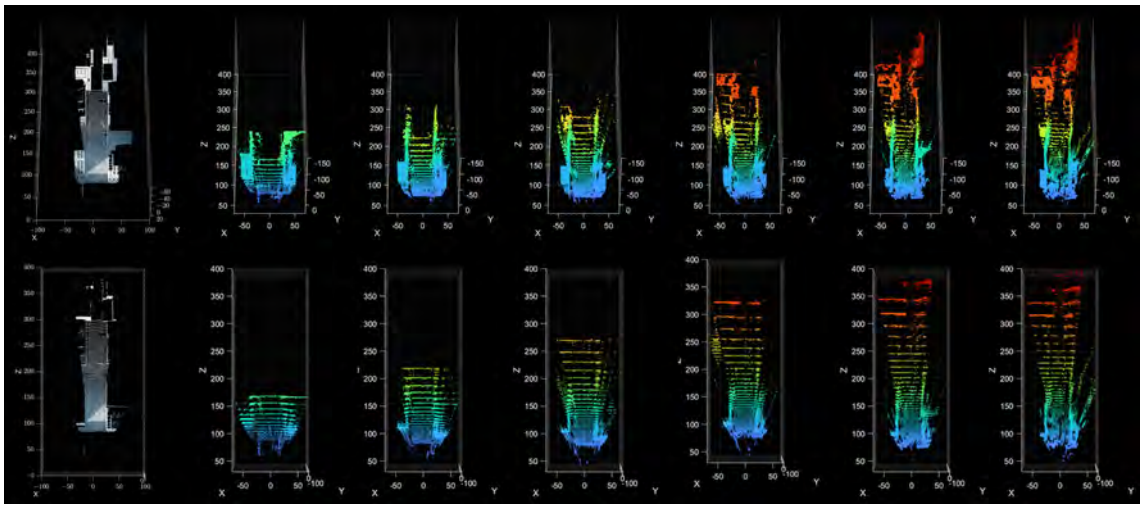

Figure 3.3: Point cloud projected from each corresponding baselines from a top-down view (below) and 60 degrees pitch down view (above).

The estimation error evaluation has been shown in [42]. This is the visualization of the depth estimation error which falls in the range of equation 2.12 as the upper bound. The evaluation scene is shown in Figure 3.4 where depth estimation of a house is evaluated. Point clouds projected from this experiment are shown in Figure 3.5, and their estimation error compared to ground truth is shown in figure 3.6. The estimation error can be seen mostly below the theoretical value given by equation 2.12 plotted in red dashed line.

The conclusion that can be concluded in this simulation is that each baseline distance has their own effective range. The use of small baseline has their range limited and error of far objects being high. On the other hand, the use of excessive baseline results in the deterioration of depth estimation for near field objects.

Figure 3.4: Reference frame house.

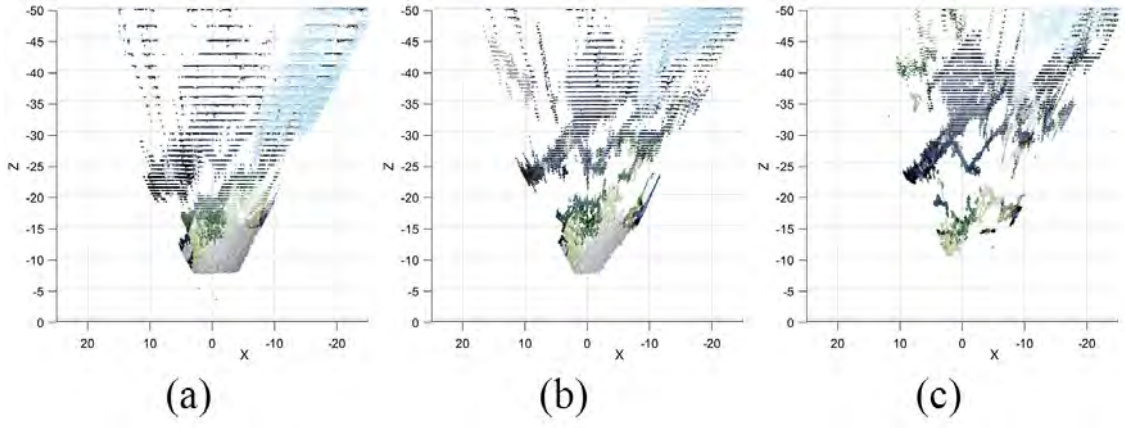

Figure 3.5: Point cloud of the house from (a) 1 meter baseline, (b) 2 meter baseline, (c) 3 meter baseline.

3.2 Baseline fusion algorithm

Based on the conclusion drawn from section 3.1, the baseline distance has their own effective range. Therefore when multiple baseline distances are used, in order to extract only useful depth estimation from their effective range, a baseline fusion algorithm is developed and proposed in this work as well as the published work [42].

The baseline algorithm estimates the usable range of the point cloud of a specific

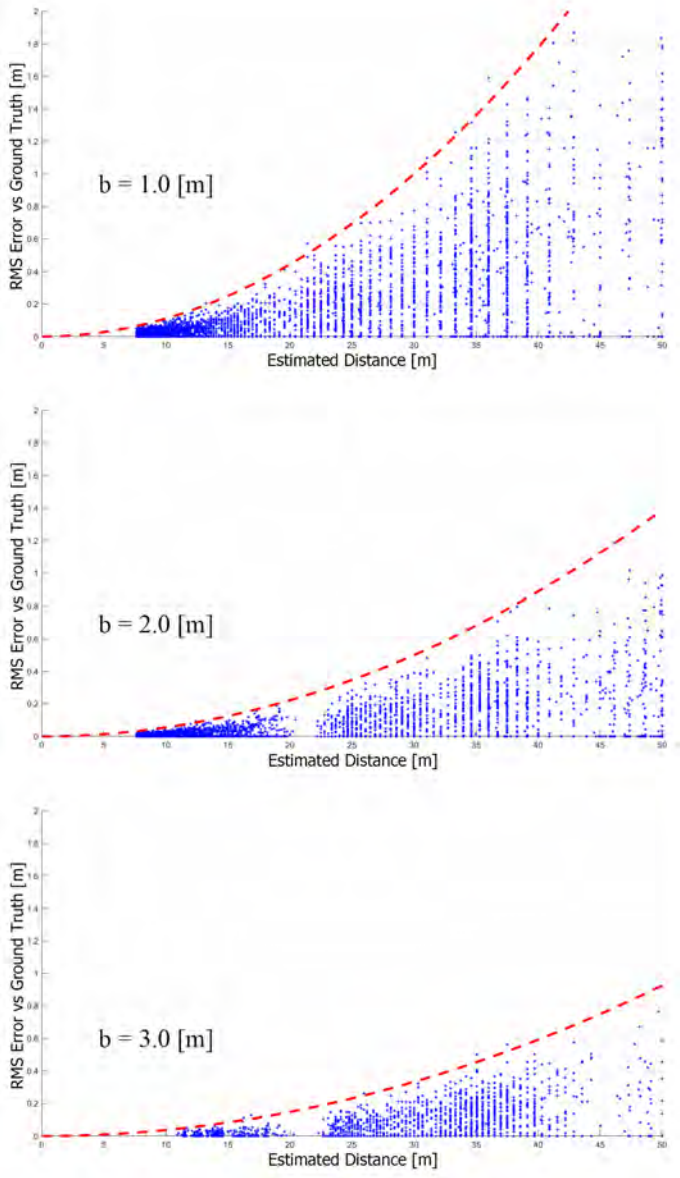

Figure 3.6: Estimation error of the point cloud vs ground truth (blue dot), compared to the theoretical error value (red dashed line) with 1 meter baseline (top), 2 meter baseline (middle) and 3 meter baseline (bottom).

baseline such that only the accurate part is used and the noise is excluded as much as possible. The algorithm is developed based on the estimation error upper bound equation 2.12. The foremost information that is used to construct the algorithm is the

prediction of maximum range that can be used for a baseline distance. The maximum range of a baseline distance can be given by the following equation:

$$z_c = \sqrt{\frac{\epsilon_c \ b \ f}{\epsilon_d}} \tag{3.1}$$

Where z_c is the maximum distance, ϵ_c is the desired error threshold. Given a threshold, the maximum distance that the threshold will hold for this specific baseline can be calculated. In case of using only a single baseline, this equation can estimate the maximum distance that this baseline is deemed effective, meaning that the estimation error ϵ_z is below or equal to the error threshold ϵ_c.

However, even though the maximum effective distance can be calculated, the minimum effective distance cannot be calculated. Obviously, as discussed in section 2.2, the minimum distance can be calculated from the non-overlapped area of the FOV using simple trigonometry. Still, the minimum distance that the depth estimation of near field objects becomes noise as the result of excessive baseline cannot be calculated. The factors that affect the determination of the minimum distance, in addition to the baseline distance and the object distance, is the shape, size, texture quality, planar position of the object. In the current state, it is impossible to obtain such information using existing methods, therefore the only sign that indicates that the baseline distance is too large is when the quality of disparity image of the object of interest starts to deteriorate. Which can only be observed in the real scene when the system is actually used.

The next step is when multiple baseline is used. In order to determine the distance of the baseline used as well as their effective range, the user needs to specify the following parameters:

- n_B: The number of baselines to be used.
- ϵ_c: The maximum error threshold.
- z_{max}: The maximum distance that the ϵ_c should hold.
- (optional) z_{min}: The minimum distance of the area of interest.

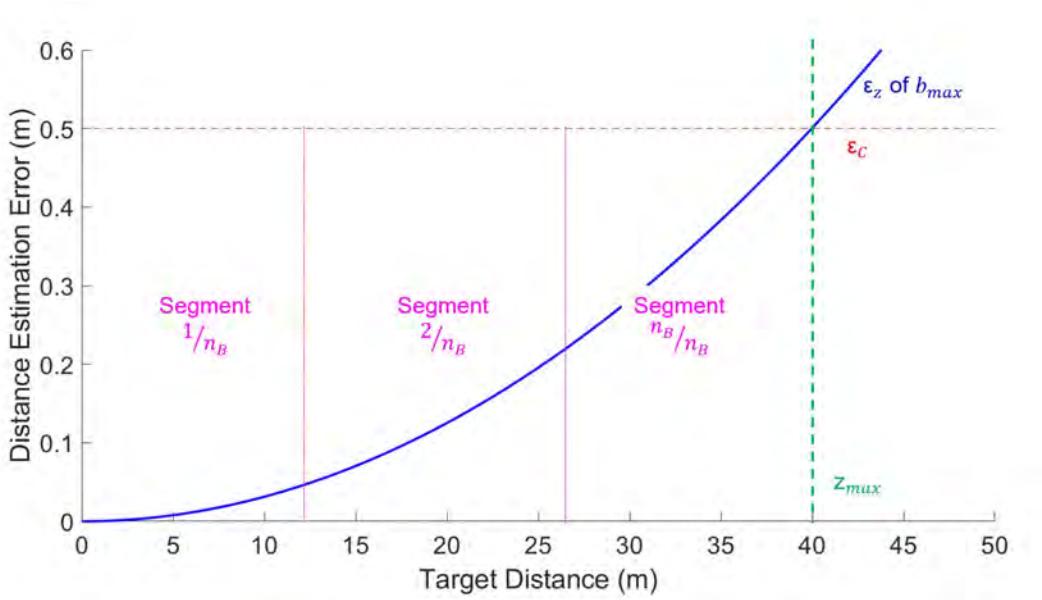

Figure 3.7: Example case of the baseline fusion algorithm where the given conditions are $n_B = 3$, $\epsilon_c = 0.5$ m, $z_{max} = 40$ m, and $z_{min} = 0$ m.

From the parameters provided, a set of baseline distances that should be used to satisfy the condition of the parameters, as well as their effective range can be calculated. Figure3.7 shows an example case that provides the steps of the proposed algorithm:

1. Calculate the largest baseline needed to satisfy the condition with the following equation:

$$b_{max} = \frac{z_{max}^2}{\epsilon_c f}\epsilon_d \tag{3.2}$$

2. The range between z_{max} and z_{min} is divided into equal segments. The number of segments corresponds to n_B. The size of each segments is calculated by:

$$z_{seg} = \frac{z_{max} - z_{min}}{n_B} \tag{3.3}$$

3. After getting the size of the segment, the range of each segment can be given by an equal distance, for example, in the example case in Figure3.7, the range of each corresponding segment is, segment 1 s_1, [0, 13.33], segment 2 s_2, [13.33, 26.66] and segment 3 s_3, [26.66, 40.0].

4. At each intersection point between the maximum range of each segment and ϵ_c, baseline distance used for each segment can be calculated using equation 3.2 where z_{max} is replaced by the maximum range of each segment and b_{max} is replaced by the baseline distance that will be used for each sector.

5. In total, one segment has its own baseline distance. The effective range of each obtained baseline is equal to the range of each segment. For example, baseline b_1 is used for range 0 to 13.33 meters, baseline b_2 is used for 13.33 to 26.66 meters and baseline b_3 is used for 26.66 to 40.0 meters.

6. The system obtains image pairs at each calculated baseline and point cloud of each baseline is reprojected into each corresponding point cloud.

7. The depth axis of the point cloud is trimmed to the range of each segment specified in step 5.

8. The trimmed point clouds are fused into a resultant point cloud

Using the developed algorithm, we can get the baseline distances that need to be used and their effective range. Then the resultant point cloud can be obtained from fusion of the point clouds. In this fashion, the estimation error ϵ_z is limited to the specified error threshold ϵ_c.

In order to confirm the effectiveness of the algorithm, a large area mapping experiment is carried out and the effects of the proposed system in a large area mapping is discussed. A cityscape of an area up to 400 meters is used for the experiment. Figure 3.8 shows the used area, and Figure 3.9 shows the ground truth of the area, while the previously shown Figure 3.3 shows the point clouds projected from each baseline.

Figure 3.10 shows the disparity image produced from each baseline. Noise can be observed in excessive baseline cases where closer objects become noise. For simplicity, the size of baseline is decided beforehand to be of whole numbers such that it is easier to understand and easier to verify. Using the fusion algorithm, the error threshold is set to be 10 percent of the estimated distance, and the trimming point for each

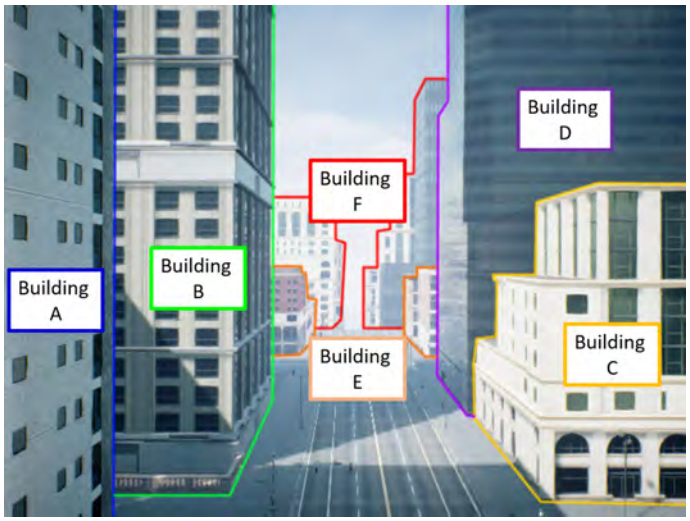

Figure 3.8: Reference frame with area of interests marked in color

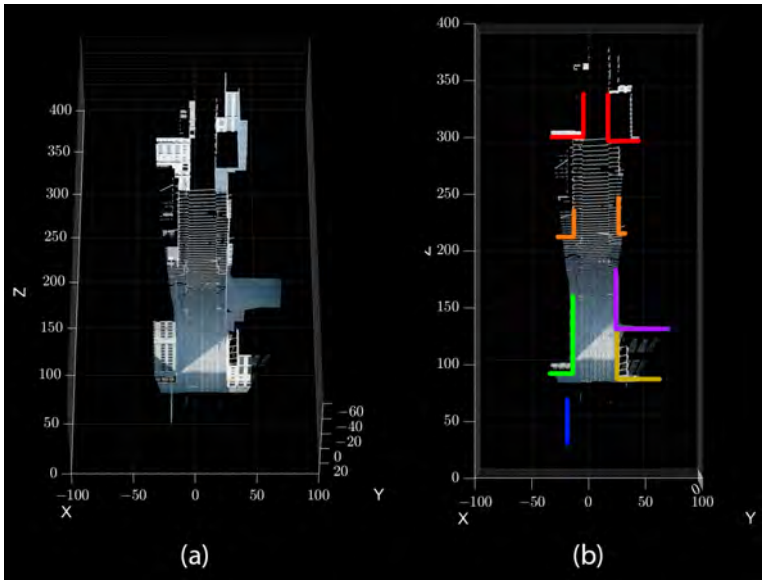

Figure 3.9: Ground truth of the experiment area viewed from (a) 45 degrees pitch, and (b) directly above with each area of interest marked in colored lines.

baseline is 120, 240, 360, 480 m respectively, where the depth limit is set to be between 0 and 400 m.

Figure 3.11 shows the combination of all point clouds from all baselines. Image (*b*) shows the resultant point cloud generated from the baseline fusion algorithm,

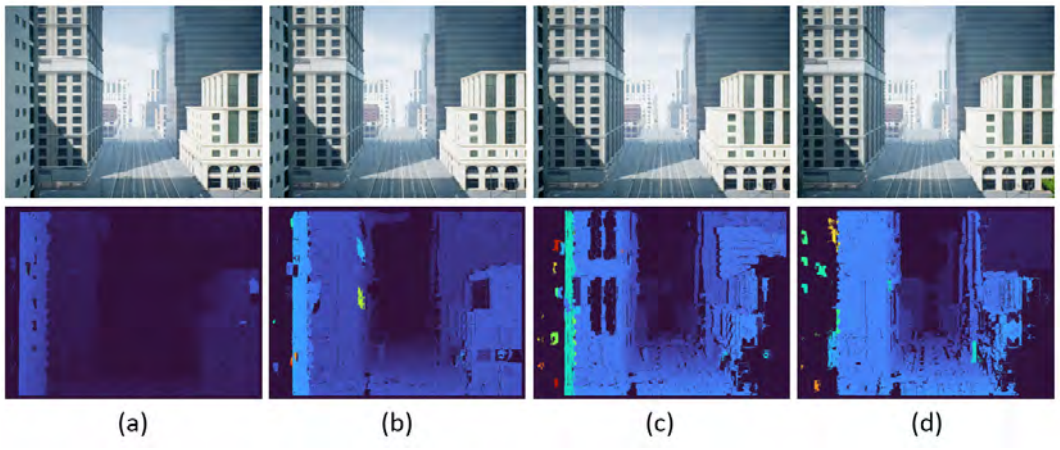

Figure 3.10: Image frame from baseline (a) 2.0 m, (b) 4.0 m, (c) 6.0 m and (d) 8.0 m, and their respective disparity image.

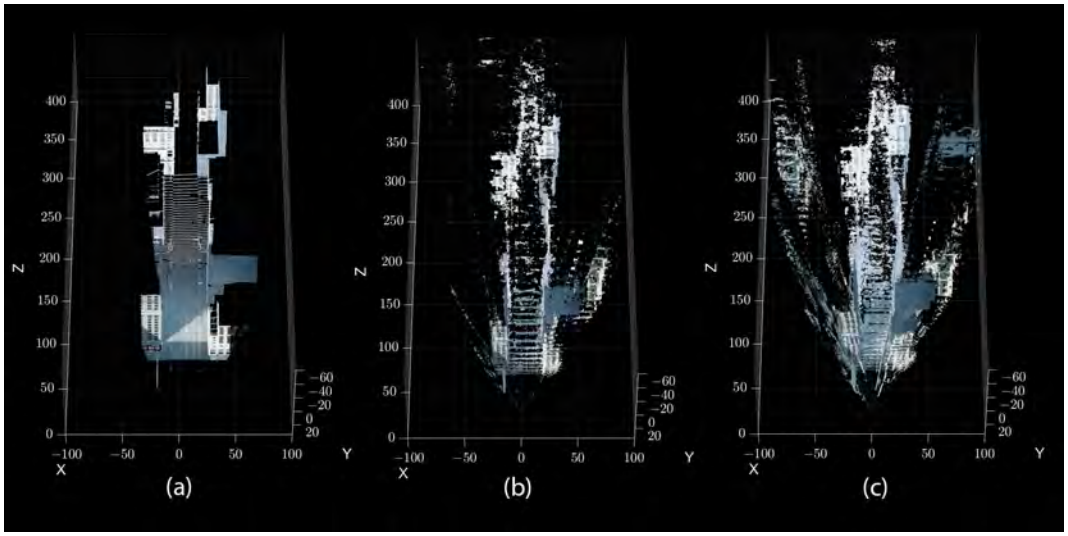

Figure 3.11: Resultant point cloud maps of (a) ground truth, (b) trimmed point cloud, and (c) non-trimmed pointcloud.

while image (*c*) shows the resultant point cloud from the direct combination without trimming. Noise in image (*b*) can be seen much lower than image (*c*). This can be concluded that the baseline fusion algorithm greatly limits the amount of noise by filtering the non-proportionate baseline distance to depth range. Furthermore, the algorithm reduces the amount of redundant points. For example, the points of near-

field objects which reflect in many baselines, if the baseline fusion algorithm is not used, the points from all baselines are used. Resulting in the near-field points to be abnormally and redundantly dense. By cutting these points from other baselines, this problem can be countered.

3.3 Characteristic of horizontal and vertical setup

The effects of using horizontal setup and vertical setup stereo varies. In the simulation, the characteristic of each setup is explored and evaluated. Figure 3.12 shows the disparity image from the same scene from a horizontal setup and vertical setup.

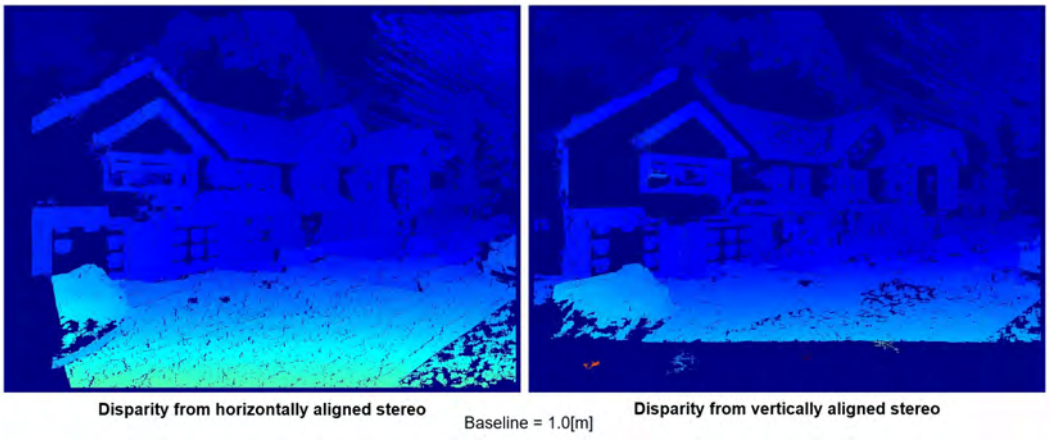

Figure 3.12: Comparison of disparity image obtained from vertical stereo and horizontal stereo of a house scene.

The first noticeable difference is the non-overlap area axis. In horizontal setup, the occlusion from non-overlap area begins on the left side of the disparity image and increases as the baseline increases. This is because the primary frame is the left frame, and the right frame is the secondary frame. The area that can be seen on the primary frame but cannot be seen in the secondary frame then appears as the void in the disparity image. On the other hand, in the vertical case, the primary frame is the camera below and the secondary frame is the camera above. Therefore, the non-overlap area begins from below the disparity image.

The second difference is the object occlusion axis. As described in section 2.1, the occlusion of the area behind the objects cannot be seen by both cameras, therefore appears as occlusion. The axis of those occlusions depends on the axis of the setup. In horizontal setup, the occlusion usually occurs at the area to the left of the object, on the other hand, in vertical setup the occlusion occurs below the object. The size of occlusion depends on the distance and position of the object. The closer the object, the larger the occlusion.

The third difference is the robustness against different shapes of the object. Since the stereo matching algorithm works well on recognizing distinct features, therefore some of the easiest features to detect are the edges of the objects. At the edges of the object, distinct features can be easily detected from the contrast in color for color images, and the brightness in grayscale images. Therefore, horizontal setup usually works better with vertical shaped objects such as light poles or pillars, whereas vertical stereo works better with horizontal shaped objects such as bridges.

The final difference is the types of scenes for each case. In the horizontal case, an open scene where the area is wide open is fitted. The reason is the movement space for the UAVs which has to move sideways, therefore an open area is required for such movement. Furthermore, the sideward movement can cause the object near the camera to become occlusion for farther objects, so a scene with no obstacles and open area such as a field or a far view of a village is more effective for horizontal setup. On the other hand, vertical setup is more fitted for scenes with limited line of sight, such as an alleyway with buildings on two sides, or a city scene. The movement axis of the mapping process is vertically, meaning upward, therefore not much space sideways is required. Additionally, by translating the camera vertically, the objects close to the camera will not become occlusion in such a case, because the object does not translate horizontally like in a horizontal case.

3.4 Fusion of horizontal and vertical stereo

As reported by [35], the combination of horizontal and vertical stereo yields improvement in depth estimation. In the proposed system, a simple and comprehensive horizontal and vertical stereo fusion algorithm is proposed. The algorithm can be described by the following steps:

1. Two image pairs are obtained from horizontal and vertical setup of the same baseline. The two pairs can be obtained either simultaneously or sequentially. Different baseline can also be used but the same baseline is recommended so the effective range of the disparity matches between two setups. Otherwise, the baseline fusion algorithm may need to be used.

2. All four images from two image pairs are rectified such that all the images are level to the horizontal ground and the same yaw. The yaw of the primary frame of the vertical setup is recommended to be used as the reference yaw.

3. Stereo matching each image pair separately. Resulting in horizontal disparity and vertical disparity.

4. Project the disparity image to 3D point clouds, resulting in horizontal point clouds P_H and vertical point clouds P_V. Combine two point clouds using this condition:

$$P_R(u,v) = \begin{cases} null & \text{if } P_H(u,v) = null \text{ and } P_V(u,v) = null \\ P_H(u,v) & \text{if } P_V(u,v) = null \\ P_V(u,v) & \text{if } P_H(u,v) = null \\ average(P_H(u,v),\ P_V(u,v)) & \text{otherwise} \end{cases}$$

   Where P_R is the resulting point cloud and (u, v) represent the pixel coordinate of each pixel in the primary image.

5. The resulting point cloud P_R is obtained.

Figure 3.13 shows the implementation of the proposed algorithm. The occlusion axis differences can be noticed in the depth image as voids around the objects. The combination of horizontal and vertical stereo fills up many of the voids as they are filled with points from the other setup. This results in a more complete point cloud compared to using only one setup.

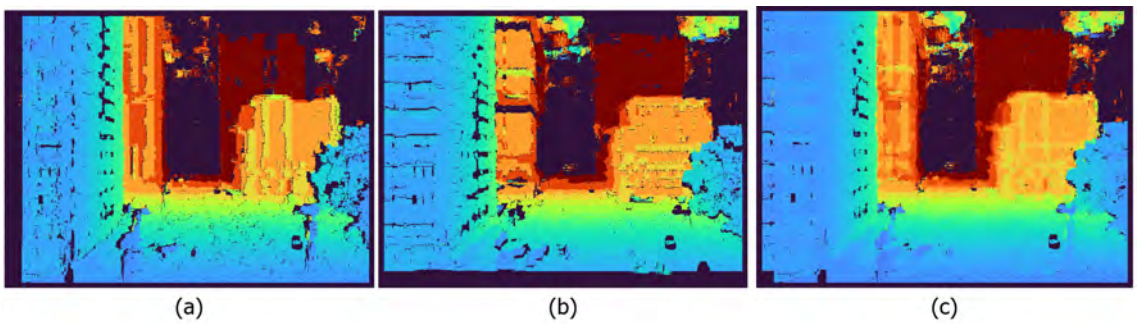

Figure 3.13: Comparison of depth image obtained from horizontal stereo (a) and depth image obtained from vertical stereo (b) and the fused image between both (c).

3.5 Characteristic of tilting stereo

The issues of excessive baseline has been observed in section 3.1. As the baseline increases, the stereo matching of near field objects becomes more difficult. At the point where baseline is excessive, the stereo matching of these objects fails resulting in noises. The utilization of tilt angle has been explored by [43] for wide baseline stereo. In the proposed system, the utilization of tilting cameras inward towards each other is further explored in a flexible configuration system.

Figure 3.14 shows the images obtained from an experiment. In the experiment, an increasing baseline is used from 2 meters, 4 meters, 6 meters, and 8 meters. Horizontal setup is used. In the experiment, the first case is a parallel stereo with both cameras facing parallel at all baselines. In the second case, the secondary camera is tilted inward toward the primary camera 1 degree for every 1 meter of baseline. The disparity image of the same baseline both cases are compared.

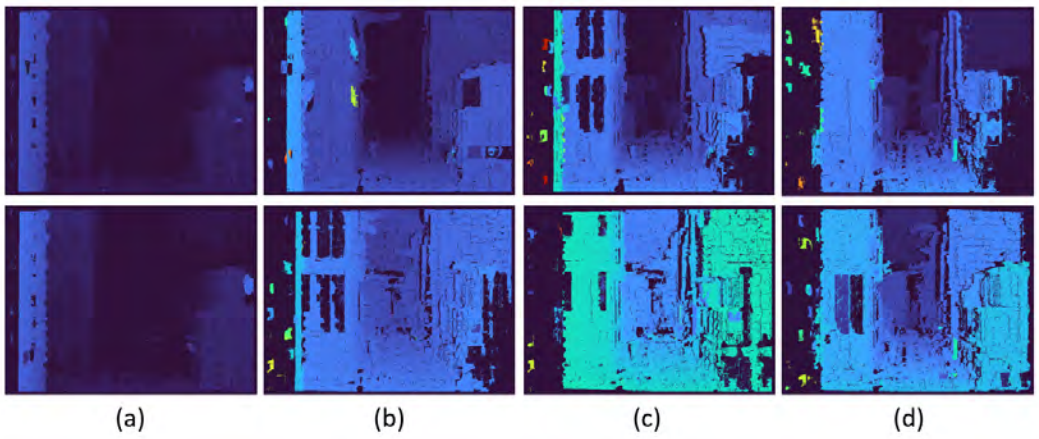

Figure 3.14: Disparity image from baseline (a) 2.0 m, (b) 4.0 m, (c) 6.0 m and (d) 8.0 m. The row above is computed from non-tilting stereo pairs and the row below is tilting stereo pairs.

At large baselines, improvement in the completeness of the near-field objects can be observed in the disparity images. By tilting the stereo cameras inward, the depth estimation of near field objects can be retained even in excessive baselines. The disadvantage of using the tilt is the depth estimation of the sky. In the parallel case, most of the sky can be seen excluded from the disparity images due to the minimum disparity limit at 32 pixels. However, in the tilted case, the disparity of the sky is included even with the filter. This effect is caused by the sky being rectified to match the parallel axis. Therefore, in the view of the stereo matching algorithm, the sky seems to have such that it has disparity in the image. If the tilt is too large, that disparity exceeds the minimum disparity and therefore included in the disparity image. This problem happens for very far objects, far exceeding the range of the baseline such as the. Therefore, in most cases, this problem can be countered by excluding the sky from depth estimation by either masking or color filtering.

3.6 Simultaneous use of multiple configuration

The advantage and effectiveness evaluation of using more than two UAVs has always been a focus in the proposed work. The possibility of using multiple configurations simultaneously is explored in this simulation experiment. In the aforementioned cases, two UAVs are used to collect images. However, when only two UAVs are used, only a single stereo configuration can be used in an instance. Though the two drones can sequentially acquire images from a set of configurations and fuse them later, the issues remain in the real-time capability of the system.

By using more than two UAVs, the advantages of a multiple viewpoints system is the ability to use multiple configurations simultaneously. Firstly, multiple setup, both horizontal and vertical setups can be used at the same time. The images acquired from both setups can be processed with the proposed fusion algorithm for horizontal and vertical stereo. Therefore, the effectiveness of depth estimation can be increased.

The second possible use is the use of multiple baseline simultaneously. By having multiple baseline at a time, multiple depth ranges can be monitored at a time. Figure 3.15 shows the use of a two baseline system to monitor an area with large depth variability. In use with the baseline fusion algorithm proposed in section 3.2, an effective range of a baseline can be determined. Therefore, the depth estimation of a baseline can be trimmed to a specific range as seen in the figure. By keeping the UAVs at this configuration, real-time depth monitoring of the area is possible. This extends the range of the monitoring which is the unique advantage in the proposed system.

The final possible use is the use of mixed configurations. For example, in an area with different kinds of scenes, the appropriate setup can be used for a specific part of the area, and another in other areas. As discussed in section 3.3, each specific setup has their own scene which is matched for their characteristic. A combination of different setup at different baseline can be used to map a specific part of the area with an appropriate configuration.

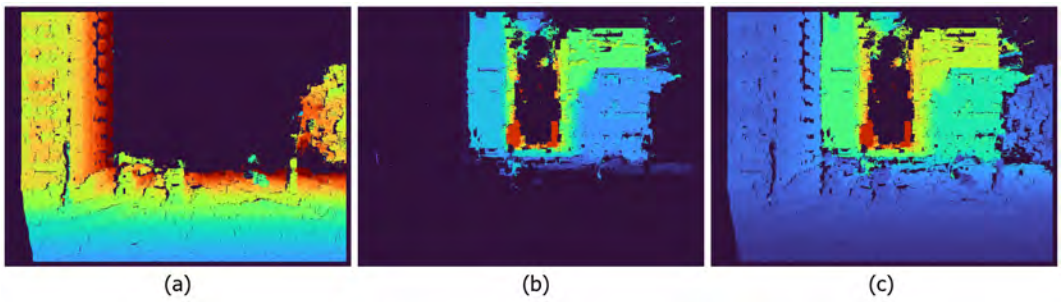

Figure 3.15: Normalized depth image of a two configurations and two baselines system with (a) horizontal setup 2 m baseline, (b) vertical setup 4 m baseline, and (c) combined depth image of both configurations.

Chapter 4

Implementation with monocular camera based tracking

4.1 System overview

Physical system prototypes have been developed. In total, separated by major changes, three system prototypes have been developed. The latter is the improvement of the earlier using the advantages and flaws from the previous system. The major change of the system is the tracking method, such that chapter 4, chapter 5, and chapter 6 are separated by the tracking method. In this chapter, the first system implementation is discussed.

In the first system design, the first consideration is the type of UAVs used in the system. In this implementation, small UAVs are chosen since the size of the UAV means the smaller the size of the minimum baseline that can be used without the UAVs crashing into each other or having their thrust interfere with each other. Two small-sized UAV, DJI Tello, are used in this implementation. Figure 4.1 shows the DJI Tello used for the implementation. Table 4.1 shows the general specification of DJI Tello.

Figure 4.2 shows the overview of the system design. A ground robot is used

Figure 4.1: DJI Tello used in the first implementation mounted with two AR markers.

Table 4.1: General characteristic of DJI Tello.

Camera Resolution:	960 × 720 pixel
Dimensions:	98 × 92 × 41 mm
FOV:	82.5 deg
Weight:	80 g
Payload (centered):	80 g

as a reference point on the ground. On the ground robot, two sets of gimbals and a monocular camera are mounted on the ground robot. These cameras are used for tracking the UAVs. The method of tracking is AR marker pose estimation. A number of AR marker [44] is attached on each side of the UAV such that it can be seen from the camera from various different angles.

The design goal of this implementation is to use the proposed system for a ground robot guidance system similar to [45] and [46] where the UAVs creates the map of the area around the ground robot. The ground robot can localize itself based on the UAVs tracking position such that the ground robot does not need additional sensors for localization.

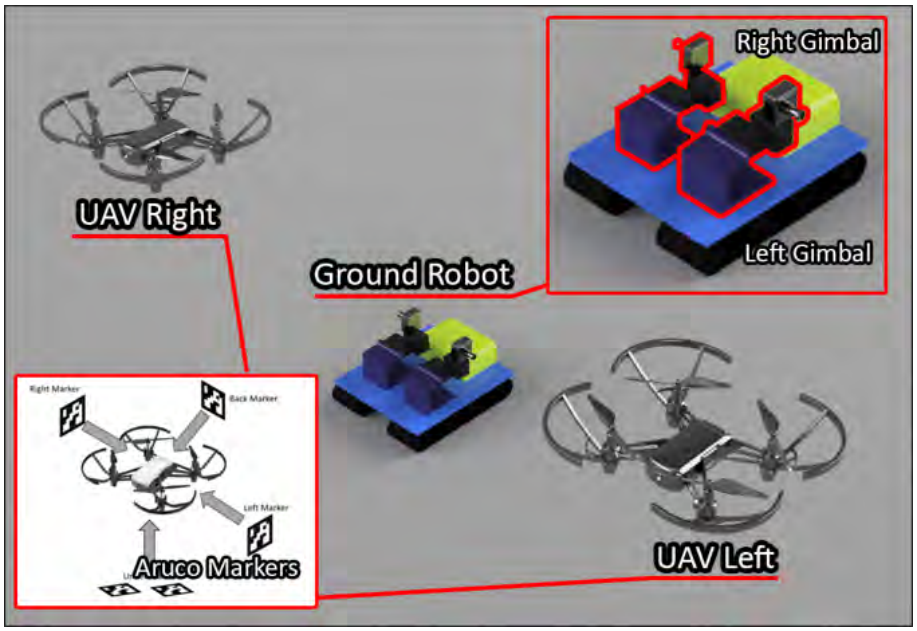

Figure 4.2: System overview of the first implementation.

4.2 Tracking method

The tracking method for this implementation relies mainly on the camera mounted on the gimbal on the ground robot. The camera and the gimbal will follow the UAVs as the UAVs move around. Pose of the UAV can be calculated with AR markers attached on several sides of the UAVs. The specific type of AR marker used is AprilTag 2 [47].

Figure 4.3 shows the used gimbal and camera. The gimbal is a 2-DOF gimbal, where yaw and pitch of the camera is controlled. The servo motors used in the implementation are Robotis MX-12W, a 360 degree continuous servo with the torque of 0.2 N·m and the resolution of 0.09 degrees. A high resolution servo is chosen in this case to maximize the accuracy of tracking by reducing error from the servo command. The camera used in this implementation is the ELP Webcam camera. Table 4.2 shows the specification of the camera.

A specific lens of the camera is chosen to use a very narrow field of view camera. The reason for the narrow field of view is to maximize the range of the AR marker

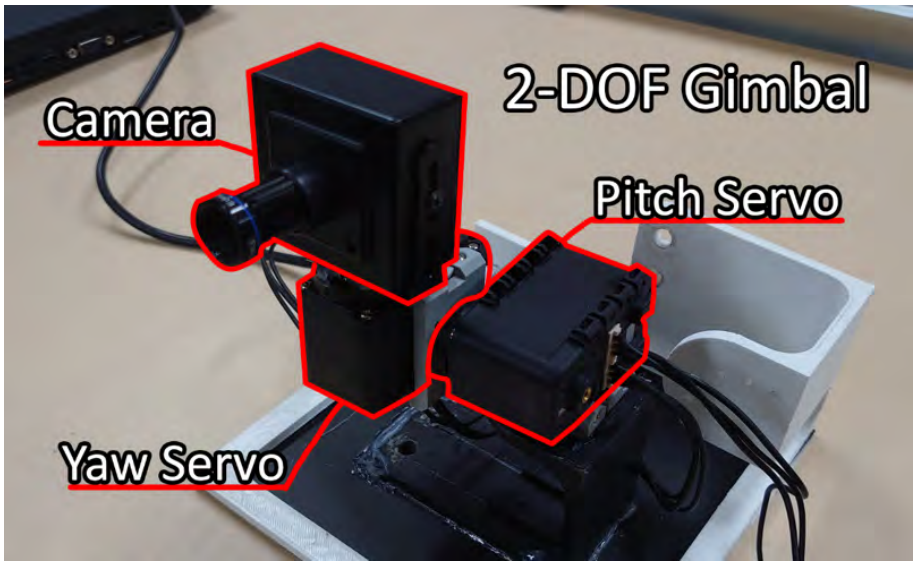

Figure 4.3: Camera and gimbal used in the tracking system of the implementation.

Table 4.2: General characteristic of Monocular Camera used for Tracking.

Camera Resolution:	1920 × 1080 pixel
FPS:	60 FPS
FOV:	21.5 deg

pose estimation. Since the resolution that can be used is limited, and the increase in resolution would worsen the performance of the image processing algorithm used for the marker pose estimation, a narrow field of view camera is the viable choice. The achieved maximum detection range using the proposed setup with the maker size of 8 centimeters is around 4 meters away from the camera when the marker is exactly perpendicular to the camera. Therefore, according to the system setup, two cameras are used for the tracking, each having 4 meters tolerance, the maximum of 8 meters baseline can be used.

In this implementation, the gimbal of the camera needs to see the UAVs at all times, keeping the UAVs in their field of view. In order for the gimbal to be able to track the UAVs at all times, the gimbal needs to be controlled such that the UAV

is always in the frame. However, the solution is not as straightforward as just to rotate the gimbal such that it keeps the UAV at the dead center of the camera. Since the image acquisition has a time delay, the same goes for the servo command. When the camera is actively trying to keep the drones at the center, the delay makes it impossible. Furthermore, as the gimbal moves the camera, the frame becomes blurred due to motion blur. Such that it is proven difficult to implement a system to keep the drone at the dead center of the frame.

Therefore, a simpler gimbal control algorithm is implemented in this system. Figure 4.4 shows the illustration of the algorithm. The red area shown in the figure is a deadzone set in the image frame, and the green dot represents the UAVs position in the image frame. In image (a), the UAV's position is within the deadzone. Therefore, in this zone the UAV is guaranteed to be visible in the camera therefore no gimbal movement is needed as long as it stays in the deadzone. In image (b), the image shows the position of the UAV leaving the deadzone. In this case means that the UAV will soon leave the FOV of the camera. The last known distance of the UAV, indicated in image (c) as d, and the angle of the UAV θ can be obtained from its position in the image frame. From this information, the angle needed for the gimbal to rotate such that the UAV remains at the center of the camera again can be calculated. Finally, in image (d), the gimbal rotates such that the UAV is again at the deadcenter of the camera and within the deadzone. The steps repeat. In this fashion, the movement of the camera can be minimized and theUAV can be kept within the FOV of the camera.

4.3 Control method

In order to control the setpoints and pose of the UAVs, a user interface program is implemented in Unity. Using the concept of augmented reality to facilitate the control of the UAV, the position of the UAV and the ground robot is embedded in the software real-time virtual projection of the scene. A similar approach has been explored by [48]. In this control method, the user can directly move the virtual UAVs in the software and the software sends the setpoint to the UAV controller, which then

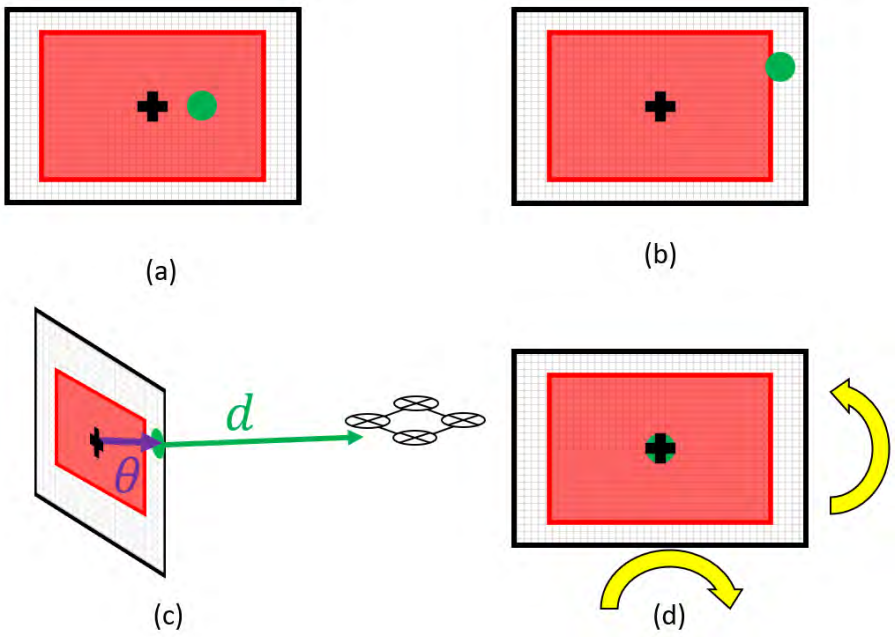

Figure 4.4: Figure showing gimbal control algorithm.

follows the position of the virtual UAV in real-time.

The control of the UAV is implemented inside the Unity program using Tello SDK, a DJI official control API for controlling DJI Tello via Wifi. DJI Tello is connected to the PC using Wifi protocol where each Tello hosts their own access point (AP). On the PC side, two Wifi USB devices are used, each connected to each AP of each Tello. Tello SDK allows the control of the UAV using the attitude thrust command. The control packet includes roll, pitch, yaw and throttle. In this case, throttle refers to the altitude control where Tello would hold their altitude on its own if the throttle is idle. Roll pitch and yaw is used for controlling the position of the UAV. The position control is implemented in the software as a PID controller with the position and rotation from the AR marker as the feedback and the output is the roll, pitch and yaw and throttle. Note that in the Tello SDK, position control is also available but with extremely limited resolution at 10 cm, therefore it is not usable in this system.

4.4 Image Processing

The image processing method in this implementation varis from the algorithm described in chapter 2 and chapter 3, since this is the first experimental system. This implementation utilizes the standard stereo image processing procedure. Each of the cameras are calibrated on the ground pre-flight using [23] checkerboard calibration method. Intrinsic matrices K_1, K_2, D_1 and D_2 are obtained. The two intrinsic matrices are used for undistortion of the images.

Next, extrinsic matrices are calculated based on the steps explained in section 2.1. The information needed for the calculation includes relative translation from camera 1 to camera 2 $T_{1\to2}$, and relative orientation from camera 1 to camera 2 $R_{1\to2}$. Figure 4.5 shows the diagram of the translation and rotation vectors. The two information can be calculated from the following equations:

$$\begin{aligned} T_{1\to2} &= R_{D1}(T_{C1\to C2} + R_{C2}T_{A2} - R_{C1}T_{A1}) \\ R_{1\to2} &= R_{D1}^T\ R_{C1}^T\ R_{C2}\ R_{D2} \end{aligned} \tag{4.1}$$

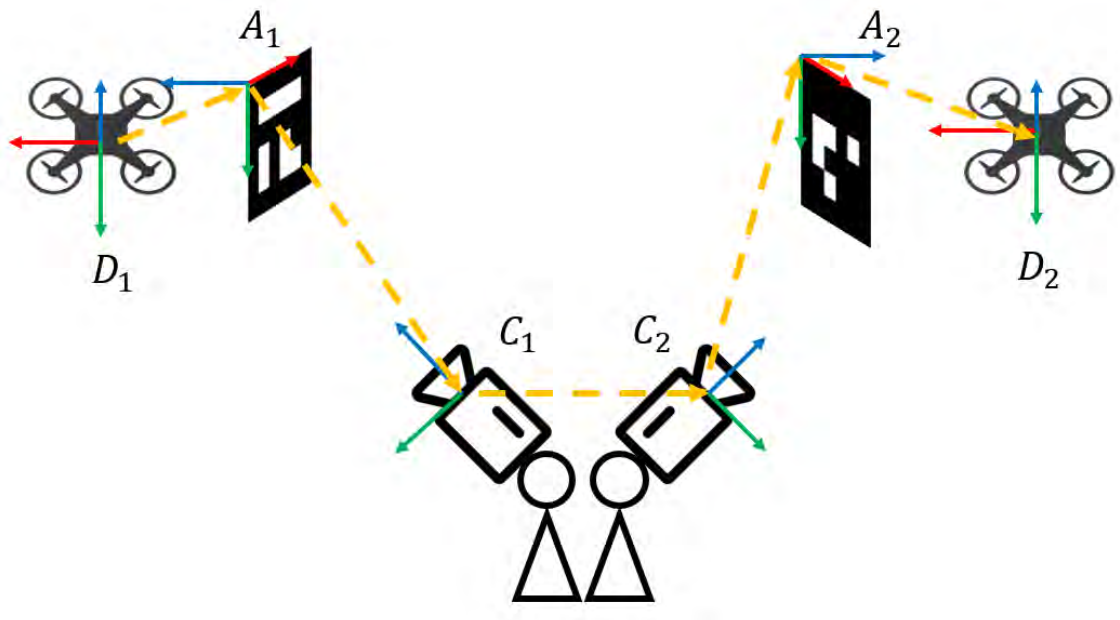

Figure 4.5: Notation of coordinate frames in the system.

Using $R_{1\to2}$ and $T_{1\to2}$, the extrinsic parameters can be calculated as follows:

$$b = |T_{1\to2}| \tag{4.2}$$

$$Q = \begin{bmatrix} 1 & 0 & 0 & -c_x \\ 0 & 1 & 0 & -c_y \\ 0 & 0 & 0 & f \\ 0 & 0 & -1/b & 0 \end{bmatrix} \tag{4.3}$$

Where f, c_x, c_y are obtained from the intrinsic matrices. Consequently, R_{rect} can be calculated using Bouguet's algorithm procedure in section 2.1 resulting in rectification matrices R_1 and R_2.

The obtained rectification matrices are then used for rectification of the image pairs obtained from the UAVs. After rectification, the rectified image pairs are then stereo matched using SGBM matching algorithm. The obtained disparity image are then reprojected into a point cloud using the projection matrix. In this implementation, the reprojected point cloud is projected from the camera frame of camera 1. Therefore, in order for the ground robot to use the point cloud in its own coordinate system, the point cloud can be translated back to the ground robot using the tracking information.

4.5 Lesson learned and issues of the system

Several issues exist in this implementation such that the system is not fully operational due to several issues faced during the development. Therefore, further development of the system is halted considering that the process to fully finish this system would result in a incompetent system. However, the process of developing this implementation gives valuable insight into what the system needs to be in the succeeding implementation.

The first issue of the system is the limitation for the camera that needs to follow the drones at all times in order to track the drone as well as using the position feedback to control the movement of the drone. The loss of tracking may result in

faulty command and may risk safety of the system. Therefore, this issue is a major issue that leads to the use of other alternative tracking systems for the drones that prevents this issue from happening in the next implementation.

The second issue is the computational power used for the AR marker pose estimation. Since the cameras need to see the drones at all times and the position tracking is used for the control, the AR marker needs to be able to be calculated at high frequency. In this implementation, the calculation of one AR marker obtains around 25-30 Hz of pose data. The process of two AR markers reduces the frequency to around 18-20 HZ. The frequency is slightly too low for the stable control of the drones, especially in the fact that image processing is not yet put into account for further frame rate reduction. Therefore, this issue raised a concern that another alternative tracking system needs to be implemented.

The third issue is the problem with the rectification algorithm. In Bouguet's algorithm, joint rectification matrices R_{rect} is computed from both $R_{1\to2}$ and $T_{1\to2}$. In the first step of the algorithm, e_1 was constructed from the normalized $T_{1\to2}$. The problem in this step is that, if translation in $T_{1\to2}$ in the y-axis and z-axis is used to rotate the image, even though the two cameras are perfectly parallel. This caused the image to rotate in the axis that it is not supposed to, causing the stereo matching to be much more difficult. This algorithm is more fitted for a perfectly aligned stereo camera, which in the case of the proposed work is almost impossible to align the two cameras perfectly. Therefore, a rectification algorithm that is used specifically for this application needs to be revised.

The fourth issue is the control software implemented in Unity. In this implementation, all the software are implemented in C# in Unity. Even though the software greatly facilitates the control and visualization, running Unity on the control PC is resource intensive. The processing resource is better used in the processing part than the visualization part. Furthermore, Unity integration with ROS at the time was extremely limited. In order to scale up the system and organize the software, ROS is necessary for the system. Therefore, the control software needs to be reimplemented in Ubuntu ROS.

The last issue is the availability of the stereo processing algorithm in Unity. In order to use OpenCV in Unity, OpenCV for Unity package is used. In this package, the implementation of the stereo processing algorithm was limited to the early versions of OpenCV 3. The available stereo matching algorithm was only BM and SGBM. However, in the newer OpenCV version, a new stereo matching algorithm is added, Quasi-Dense stereo matching algorithm that has better performance than SGBM. This is another issue to move the system to be fully operational in Unity to use the Quasi-Dense matching algorithm.

Chapter 5

Implementation with stereo camera based tracking

5.1 System Overview

In this implementation, several major improvements are introduced as a counter-measures for the issues faced in the previous implementation. This implementation corresponds to the implementation in the publication [42]. The major improvement is the major hardware changes in the system. Figure 5.1 shows the equipment used in this implementation.

The UAVs used are changed from normal DJI Tello to DJI Tello EDU. The reason for the hardware change is the more availability in the control protocol which allows more control over the UAVs' movement. An unofficial Tello control library Tellopy by hanyazou is used in this implementation.

The tracking hardware used in this implementation is the stereo infrared camera motion capture (mocap) Optitrack V120 Duo camera. The specification of the camera is shown in Table 5.1. The camera used along with Motif software, provides 6-DOF tracking information. Motif software can be set to broadcast the tracking data of the selected object via multicast protocol. A ROS package mocap_optitrack is used for the message relay from the multicast and relay the tracking information to ROS topics. The rate of publication is 200 Hz, which is almost ten times faster than the

Figure 5.1: (**a**) DJI Tello EDU micro UAVs with motion capture markers attached. (**b**) Portable dual-lenses motion capture device Optitrack V120 Duo.

previous implementation. In contrast to the normal motion capture system where each individual camera is placed separately, the V120 Duo has a fixed position to the two motion capture cameras. Therefore, the process such as wanding and ground plane calibration that is required in the normal mocap system is not required in this camera. The camera can be used directly without the calibration since the factory calibration parameters are saved within the camera itself.

Table 5.1: General characteristic of Optitrack V120 Duo.

Resolution:	640 × 480 pixel (×2)
Dimensions:	41 × 279 × 51 mm
FOV (HxV):	47 × 43 deg
Frame Rate:	120 fps

The control software as well as all the processes are moved to a python implementation in Ubuntu 20.04 using ROS Noetic. In this version of ROS and Ubuntu, The use of Python 3 as well as the latest version of OpenCV 4 can be used. Newer algorithms and optimization, as well as future algorithms can be updated in this

implementation.

The main focus of the system is largely changed from a ground robot guidance to a general purpose rapid 3D reconstruction algorithm. The development of this system aims to develop a system that can rapidly create 3D maps of a large area beyond the range of normal stereo cameras or depth cameras. Furthermore, another aim of the system is to maximize the movement distance of the UAV per distance mapped. The focus becomes the development of algorithms to realize the goal mentioned.

5.2 Tracking Method

The tracking method in this implementation uses the mocap camera to acquire the pose of the UAVs. On the UAVs in Figure 5.1, the mocap markers are placed on the top of each UAV in a non-uniform pattern. These patterns are used for tracking of the UAVs via the motion capture camera. Figure 5.2 shows the configuration of the mocap and the UAVs.

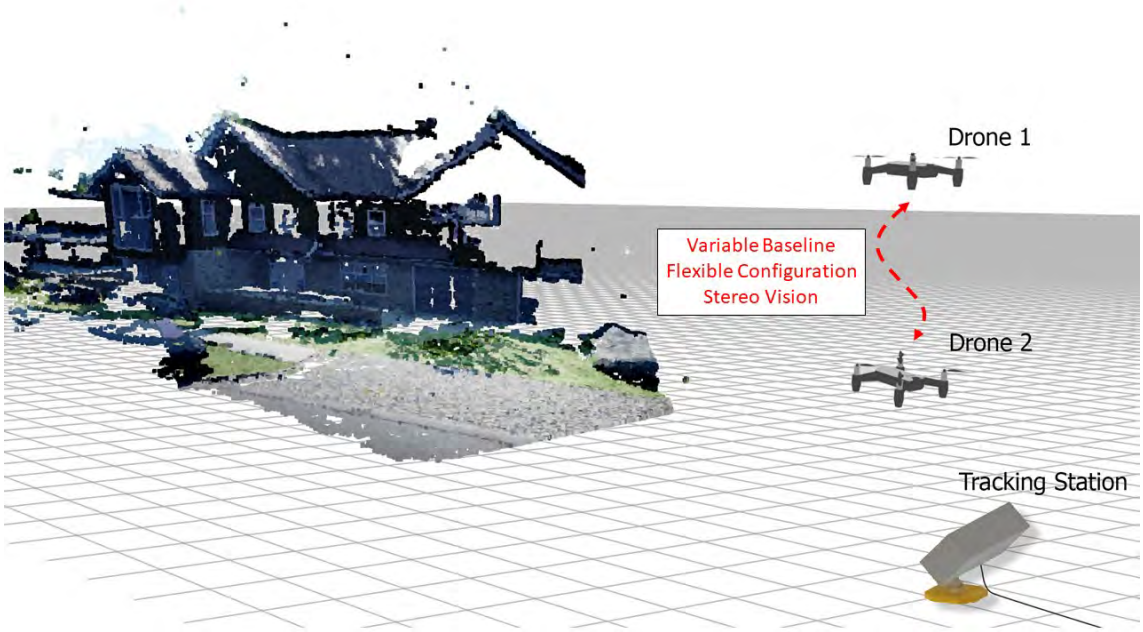

Figure 5.2: Illustrated overview of the system implementation.

However, the field of view of the mocap is fairly narrow. The two UAVs can hardly

stay in the frame simultaneously or remain in the frame as it operates. Therefore, the use of UAVs' own odometry in the tracking process is introduced.

In the new control protocol, TelloPy, a larger set of information can be obtained from the Tello. The data includes the state of the UAV, the raw IMU data, as well as the odometry value of the UAV. These data were not available in the official Tello SDK in the previous implementation. Even though the flight control software of the DJI Tello remains a black box, these data provide the information that in some way, the DJI Tello is producing its own odometry data. The method of odometry remains unknown, but below the Tello, there is an infrared-emitting LED, as well as a dual-lense camera-like component. These sensors are thought to be responsible for the height measurement as well as the odometry of the Tello, working in a similar way as a combination of a range sensor and an optical flow sensor.

The accuracy of the odometry of the Tello is evaluated. The evaluation method is comparing the trajectory of the odometry of the UAV with the trajectory measured by a 4-camera full sized mocap system. Figure 5.3 shows the compared odometry. The unit of measurement is in millimeters. The drones are moved manually in a random trajectory until the accumulated distance of movement measured by the odometry reaches 20 meters. At the final point of movement, the pose from the odometry and the pose from the mocap is compared. The final drift calculation indicates the odometry to be drifted by 0.374 meters. This can roughly be calculated as the error rate of less than 2% by the distance traveled. At this error rate, the error is considered low enough for the system implementation. Since the movement required for the mapping process is less than 5 meters, the final odometry drift is around 0.1 meter, which is considered neglectible error. Furthermore, the position is corrected by the measurement from the mocap ground tracking station as well, therefore the final drift would be less than that.

Therefore, the tracking system in this implementation is developed based on the combination of both the UAVs' odometry, IMU, and the tracking information from the mocap camera.Figure 5.4 shows the tracking flowchart. The case of the tracking, an Extended Kalman Filter (EKF) is used to combine the tracking information and

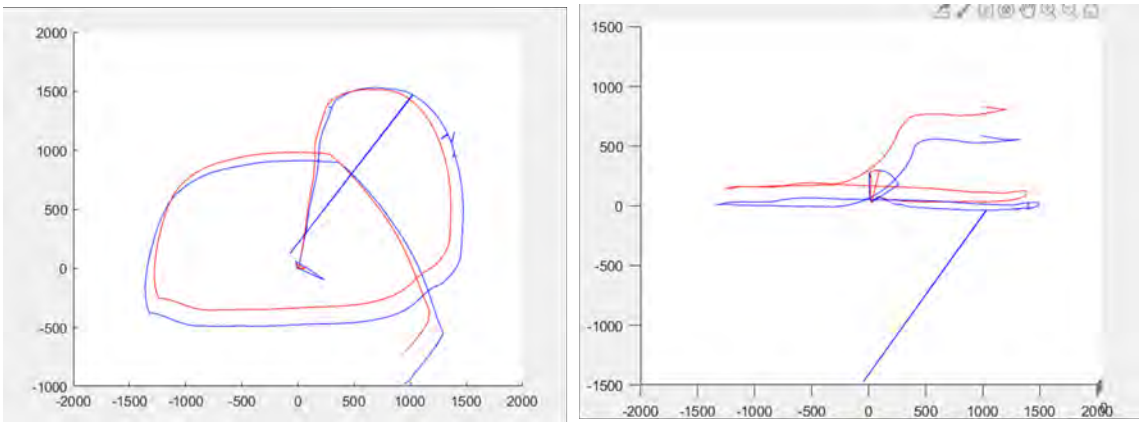

Figure 5.3: Evaluation of odometry data from DJI Tello where blue line is the ground truth trajectory from mocap and red line is the odometry trajectory.

the IMU. Using this tracking scheme, the UAV has to appear in the FOV of the mocap at least once for the tracking to be operational. During the UAV's stay in the FOV, the UAV's odometry pose and the tracking pose from the mocap is compared. The offset of the UAV's pose and the mocap pose is then saved in the software. This pose offset translates the pose of the UAV odometry to match with the pose of the UAV in the mocap coordinate system. The pose that is used for the control in this case is the pose from the mocap itself. When the UAV leaves the FOV, the mocap pose becomes unavailable. Therefore, during this time, the UAV's own odometry is used for tracking. The UAV's odometry is offset by the offset pose saved in the software such that its relative pose to the last known pose can be calculated. Therefore, the pose of the UAV can be kept track even when outside of the FOV of the mocap.

Another advantage of using the UAV's odometry is the ability to extrapolate the UAV's odometry's orientation for a level horizon plane. Since the UAV's odometry pitch and roll value always based on the level horizon, when the level horizon is compared with the orientation measured by mocap, the level horizon can be calculated

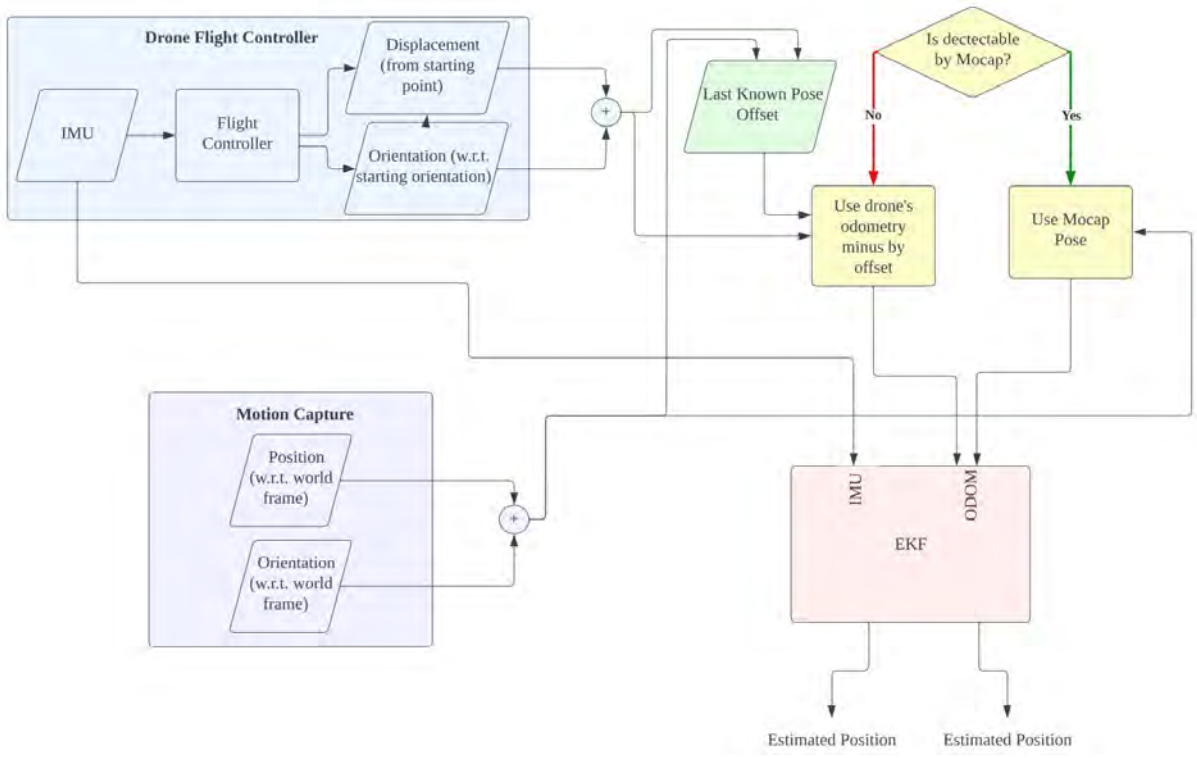

Figure 5.4: Data flow chart of the tracking method.

with the following equation:

$$\begin{aligned} R_{M\to W_1} &= R_{D1}\ {R_{M\to D1}}' \\ R_{M\to W_2} &= R_{D2}\ {R_{M\to D2}}' \\ R_{M\to W} &= average(R_{M\to W_1}, R_{M\to W_2}) \end{aligned} \tag{5.1}$$

Where $R_{M\to W}$ is the rotation matrix from the mocap coordinate system to the level horizon. R_{D1} and R_{D2} are the orientation of the UAVs in their own odometry frame. $R_{M\to D1}$ and $R_{M\to D2}$ are the orientation of each UAV measured by mocap. Supposed that the two UAVs' odometry starts at the same axis, the yaw orientation of the two UAVs are roughly equal, the yaw of the level horizon also represents the average yaw of the two UAVs. Consequently, the pose offset of each drone can be calculated using equation 5.2.

$$
\begin{aligned}
T_{O1} &= T_{M \to D1} - R_{W \to M}\ T_{D1} \\
T_{O2} &= T_{M \to D2} - R_{W \to M}\ T_{D2}
\end{aligned}
\tag{5.2}
$$

During the time both of the UAVs are in the mocap frame, the relative pose calculation of the two UAVs in the mocap frame can simply be calculated using the following equation:

$$
\begin{aligned}
R_{D1 \to D2} &= R_{M \to D2}\ {R_{M \to D1}}' \\
T_{D1 \to D2} &= T_{M \to D2} - R_{D1 \to D2}\ T_{M \to D1}
\end{aligned}
\tag{5.3}
$$

When one or more of the UAVs are not present in the mocap frame, the relative pose can be calculated using the IMU based odometry of the UAV with the following equation:

$$
\begin{aligned}
R_{D1 \to D2} &= R_{D2}\ {R_{D1}}' \\
T_{D1 \to D2} &= T_{D2} - T_{O2} - R_{W \to M}\ R_{D1 \to D2}\ (T_{W \to D1} - T_{O1})
\end{aligned}
\tag{5.4}
$$

5.3 Control method

The control method in this implementation is designed to be lightweight, but still keep the user interface function which aids the ease of control and the monitoring of the overall system. All of the nodes including control nodes and process nodes are implemented in ROS using Python. Figure 5.5 shows the overall nodes and communication in the system.

The control node for each drone is implemented in Python using TelloPy. The control nodes are implemented independently of the UI node and the image processing node such that the control nodes always keep running even if there are errors or crashes in other nodes. The function of control nodes is to receive goal position commands from the UI node, send the goal position command to the PID controlling velocity of the UAV in north-east-down (NED) axis as well as the yaw angular velocity. The feedback of the position is the position calculated from the previous section. Additionally, the control nodes receives the pose information from mocap in order

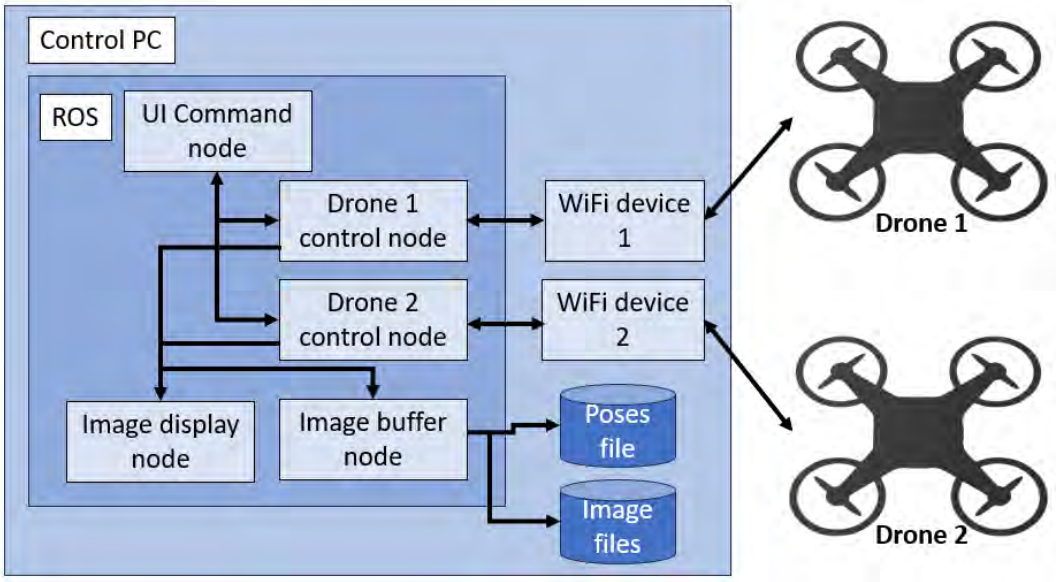

Figure 5.5: Flowchart of nodes and devices in the system.

to use them to calculate the pose offset T_{O1} and T_{O2}, as well as doing all the pose calculation such that the pose control of the UAV is possible. Each individual node sends the command to the UAV via a dedicated WiFi USB device, one for each UAV.

Another control element is the UI node that is used for monitoring the position of the UAVs as well as issuing commands. The control UI includes (1) the position control panel. While operational, the position of the UAV is marked with a colored circle with a line projecting from the center to its current forward facing yaw. The control panel also marks the FOV of the mocap camera. The height of the UAV is indicated at the bar gauge on the right most side of the panel. Furthermore, height can be controlled using the middle mouse wheel. (2) and (4) are the position monitoring gauge. While operational, a middle gray line representing the planar goal position is indicated in the middle, and the current UAV's planar position is represented in a colored line. (3) is the yaw monitoring and setting wheel. The gray line indicates the yaw goal. While operational, the current yaw of the UAV is indicated by a colored line. The goal yaw can be set by clicking on the wheel and dragging the goal yaw line to the desired position. (5) is the general information panel indicating battery level, WiFi strength, connection status in red or green rectangle, and the current status of

the UAV such as landed, or armed. Additionally, this node relays the message from the UAV, such as state, odometry, and IMU, to the control UI node, and relays the image from the front camera to the image processing node.

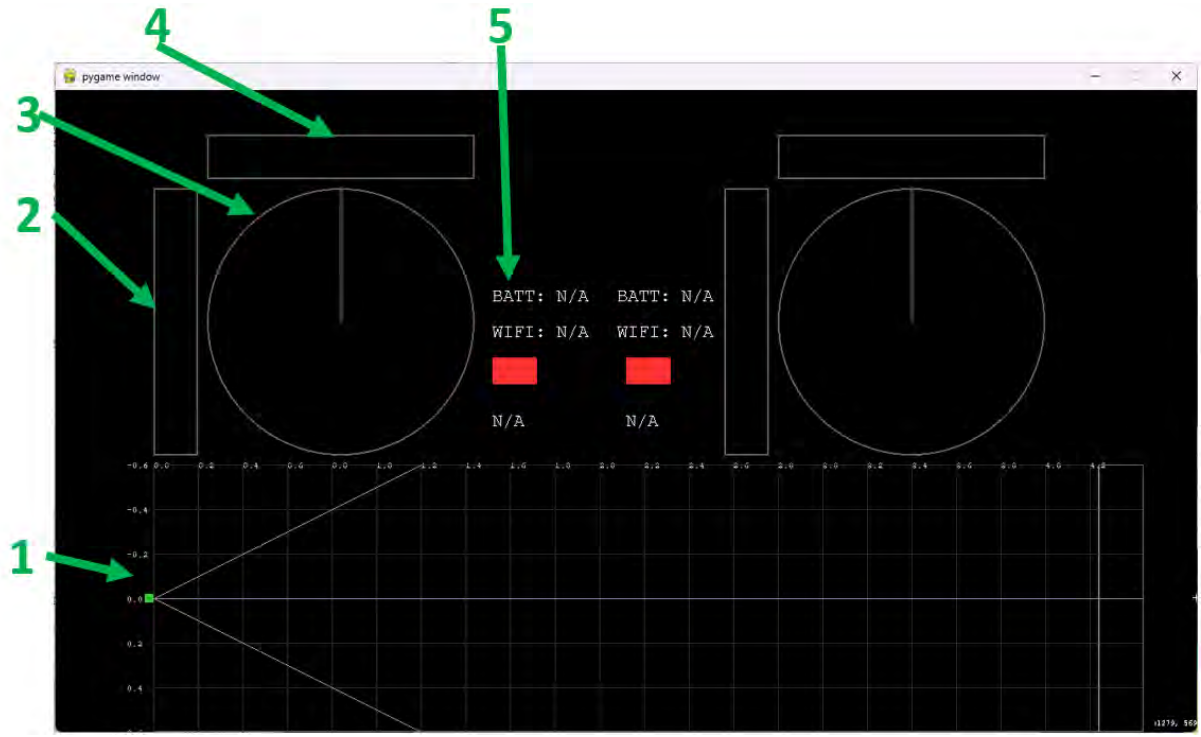

Figure 5.6: The control node user interface.

5.4 Image acquisition and delay handling

Image acquisition is done via the drone control nodes. Since the image from the camera is received along with the odometry data and the IMU data, the rough estimate can be made that those messages are time-synced to a degree. The delay between these information is small such that it is neglectable. However, the delay of the message received from the UAV and the delay of the mocap pose message exists. There is a certain amount of delay. Therefore, in order to compensate for this delay, a buffer is implemented. In this system, the images acquired from the UAVs are not processed in real-time, but processed later after the image acquisition process for simplicity to

implement the delay countermeasure algorithm.

During the image acquisition process, the UAVs are set to go to each check point, corresponding to each baseline needed for the mapping process. Once both of the UAV is within the vicinity of the checkpoint, the image buffering node saves the image from both UAVs, along with their tracking pose as files in the control PC. The image buffering node keeps the images as well as the pose for 5 seconds after the UAVs enter the vicinity for the first time, or until at least 30 images are taken before stopping. Repeat these steps until all the checkpoints are reached and the image acquisition process ends.

The purpose of saving multiple images and multiple poses is to compensate for the delay. Since the delay between the pose and the image is unknown or a vibration could happen such that the pose is not accurate, multiple images obtained are all computed. After the image acquisition, during the image processing process, the image of camera 1 and the relative pose at the middle of the 3 seconds time saved is selected. After that, the images from camera 2 in the -0.5 and 0.5 second bracket around this selected image are all fed to the stereo matching algorithm to be processed into disparity images. After the process, the success match percentage of all the images are calculated from the non-null disparity value. The disparity image with the highest match percentage is then used for the point cloud reprojection. This method is a roundabout way of solving the delay issue, however with the current implementation this is the most effective way to counter the delay issue.

5.5 Rectification algorithm

As stated in the previous chapter, the Bouguet's rectification algorithm has its disadvantage when used with a non-perfectly aligned stereo system because of the rectification matrix that depends on the relative translation vector. In this work, we propose a rectification algorithm specifically made for the case of this system.

The proposed rectification matrix calculation method decouple the relative translation vector completely from the rectification matrix such that the rectification ma-

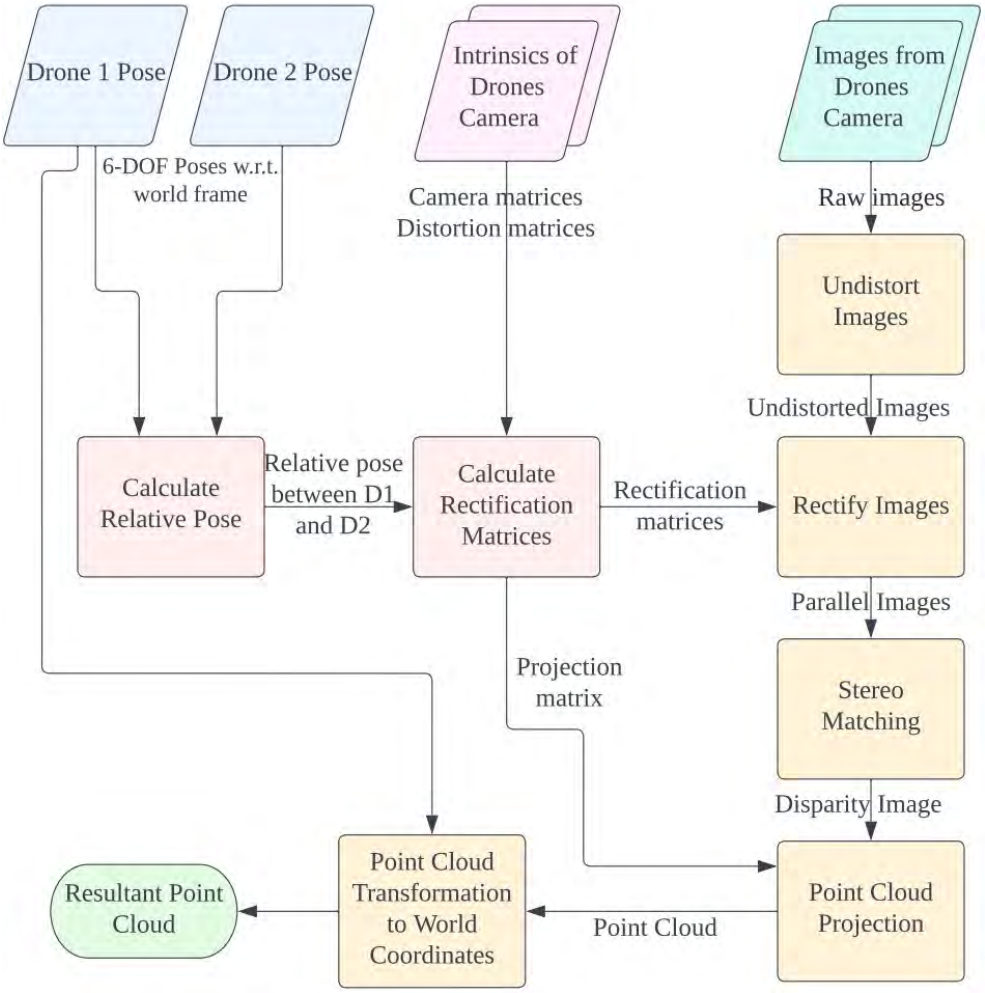

Figure 5.7: Flowchat of the overall image processing processes in this implementation.

trix only considers the relative orientation of the two cameras. By using the rotation of camera 1 as the reference point. Therefore, the image frame of camera 1 is used as is without any rectification. Rectification matrix of camera 1 can be easily indicated as follows:

$$R_1 = I \tag{5.5}$$

Consequently, the rectification matrix of camera 2 is computed such that the image is aligned in parallel with camera 1. Therefore, the relative orientation $R_{D1 \to D2}$ is directly used as the rectification matrix as:

$$R_2 = R_{D1 \to D2} \tag{5.6}$$

Additionally, the calculation of baseline distance for use in the Q projection matrix is purely calculated from only the relative translation in the baseline axis, namely the x-axis, in contrast to the standard algorithm which uses the magnitude of the relative translation vector using equation 5.7. This, too, is done in an assumption that the translation in non-baseline axes are small enough to be neglectable.

$$b = T_{D1 \rightarrow D2\,X} \tag{5.7}$$

5.6 Stereo matching, point cloud projection and baseline fusion

The use of SGBM stereo matching algorithm has been proposed in the earlier implementation. However, as stated in section 2.4, the robustness of SGM in non-perfectly parallel images is limited. In this implementation, the system has been implemented in a Ubuntu 20.04 OS and enables the use of the latest OpenCV 4 with Python 3. An additional stereo processing algorithm, Quasi-Dense Stereo becomes available. In this implementation, the stereo matching algorithm used is then changed to Quasi-Dense Stereo, which gives much better results compared to SGBM. Quasi-Dense stereo detects matches in a 2-D axis, meaning that the matches are searched throughout the image. This of course results in more false matches generated, however, this also enables the algorithm to match more points in non-perfectly parallel image pairs, such as the case of this work.

The point cloud projection in this implementation is implemented in a different mindset compared to the previous implementation. In the previous implementation, the point clouds are projected from the image frame of camera 1, namely UAV 1. Therefore, the origin of the points are at the image origin of camera 1. Since Bouguet's rectification algorithm cause the image rectification to rotate in roll axis as the result of using relative translation vector, the rectified images can be seen largely rotated sometimes. In such cases, the disparity image resulting from these rectified images are also rotated. This makes the monitoring of the quality of disparity becomes

extremely difficult without rotating the disparity image back to the original position. Additionally, the point cloud projected from the said disparity would also be rotated and needs to be rotated back to the level horizon using the rectification matrix of camera 1. In the current implementation, the images are projected from camera 1, but the points are translated and rotated such that the origin is level to the level horizon and the yaw zero is facing the same way as the yaw zero of the ground tracking station. In this fashion, the point clouds acquired from different configurations are translated to the same point of reference on the ground, namely the ground tracking station. The combination of point clouds from many different configurations becomes possible.

In this implementation, the baseline fusion algorithm is proposed and utilized for the first time, as described in the publication [42]. Since the point cloud projection of this implementation are translated and rotated to the origin point of the ground level. The advantage of this system is, even if the UAV 1 moves to some other place during the image acquisition process, the point clouds generated from those positions can still be combined with other point clouds by translating them to the same coordinate system.

5.7 Mapping Experiment

Using the proposed system in this implementation, an outdoor mapping experiment is carried out. The area of interest that is used in this experiment is shown in Figure 5.8. The area has several buildings on both sides and has a depth range of around 10 to 60 meters. In this experiment a set of three baselines are used, 1, 2, and 3 meters baselines. The setup used in this experiment is the vertical setup since it would include the area of interest without losing the left side of the area to the occlusion. Furthermore, the axis of occlusion in vertical setup is from below of the image. In this specific area of interest, the area on the below side of the image is the area close to the camera. Therefore, in a larger baseline, the area that disappears due to the occlusion is the area closer to the camera which will later be trimmed by the baseline fusion

algorithm anyway. This reduces the loss of information due to occlusion compared to horizontal setup. The last reason for using the vertical setup is that there is less space on the side of the area since it is a small alleyway, making it hard to move the UAVs on the horizontal axis. In this experiment, the ground tracking station is placed static on the ground facing 45 degrees upward.

Figure 5.8: The image of the area of interest.

The images obtained from the experiment are shown in Figure 5.9. The rectified image pairs show the movement of the UAV 2 on the vertical axis holding position at baseline 1, 2, and 3 meters. At each baseline the images are taken and saved on the disk. After the experiment, the images are processed and the selected disparity images are shown in the figure. In this experiment, a total of 3 baselines are used. However, judging from the quality of the disparity image, disparity from 3 meters baseline shows the sign of the excessive baseline. Most of the area of interest becomes noise. Therefore, the disparity from 3 meters baseline is excluded from further process.

Figure 5.10 shows the point cloud projected from the two remaining baseline from top-down view. The high error of the white warehouse in the point cloud generated from the 1 m baseline can be observed. On the other hand, the lack of the details on the road closer to the camera can be seen in the 2 m baseline. Therefore, both

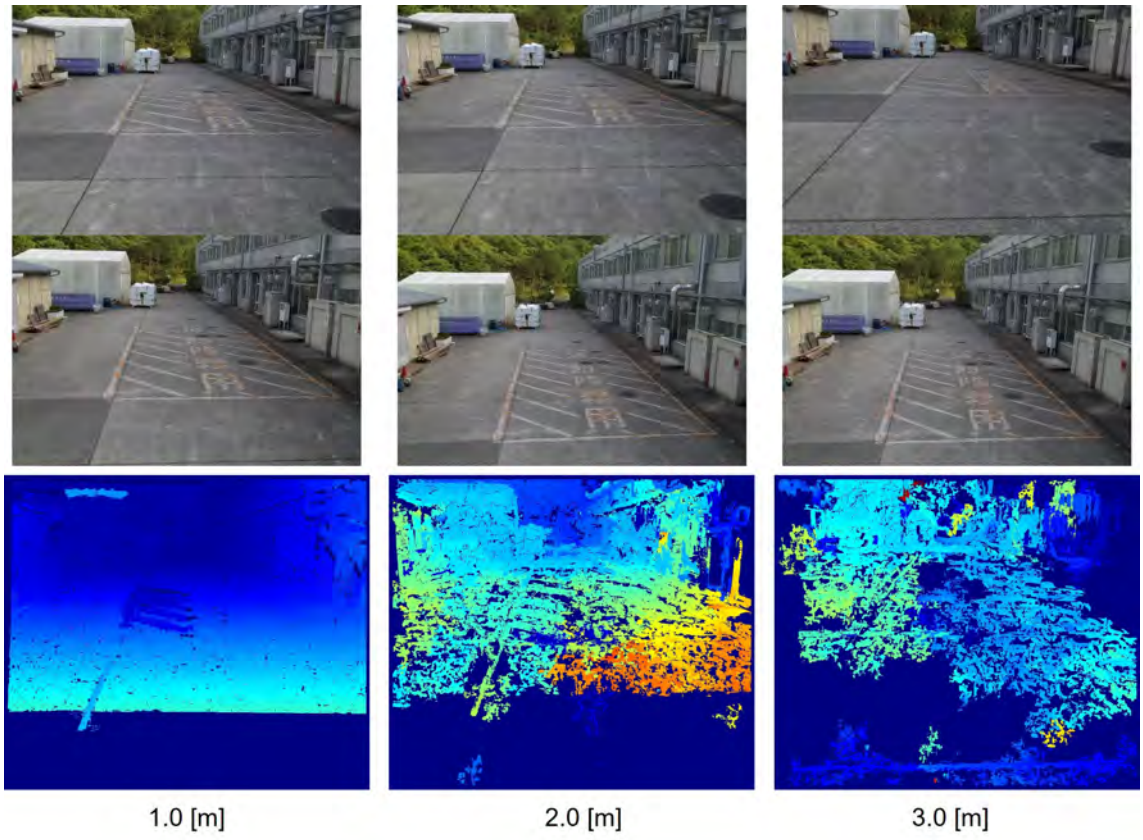

Figure 5.9: Rectified color image from each baseline and their corresponding disparity images.

baselines are fused using a trimming point at 20 m, the middle point of the range of the area of interest. The baseline fusion algorithm is used for trimming the point cloud.

The resultant point cloud can be seen in Figure 5.11. The trimming point can be clearly seen at 20 meters where the depth resolution of the point cloud changes. Figure 5.12 shows the comparison of the resultant point cloud to the ground truth obtained from Google Maps [49]. Red lines are drawn on the key points in both maps to compare the position. The position of the key points can be seen to fairly match the ground truth. Although, the range estimation of the proposed system of 20 meters is slightly closer than the ground truth. This phenomenon is seen in the simulation as well, that the depth estimation error tends to estimate the object to be

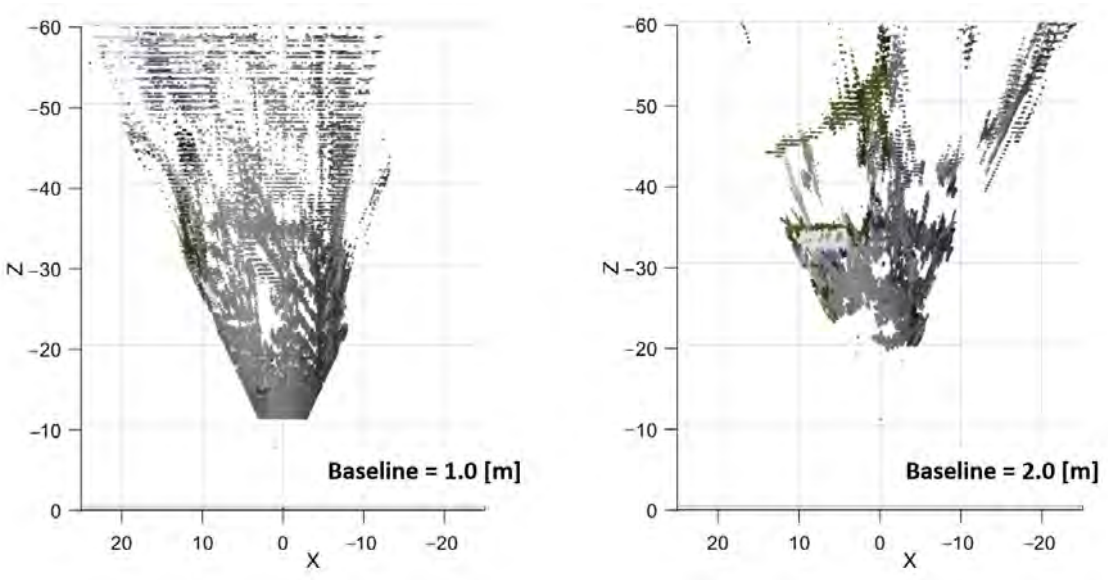

Figure 5.10: Point cloud projected from each baseline distance.

closer than reality.

5.8 Lesson learned and issues of the system

The first point of consideration in this system is the limitation of the ground tracking station using mocap. The known problem with the mocap is the vulnerability in sun-lit outdoor use. The device's infrared cameras are easily affected by sunlight and their reflection. Therefore, outdoor use, which is the main focus of the system, becomes difficult and the system only works in an ideal condition. For example, in the case of the outdoor experiment the system is operated inside the shade of buildings. In the operational area, there is no direct sunlight or reflection of sunlight in the scene, so the mocap could detect the marker. Another issue with the ground station is the limitation in the area of operation. The area of operation needs to be close to the ground tracking station such that the system can calibrate anytime by returning to the ground tracking station as needed. In this work, the ground tracking station is only used for calculation of relative pose, and acting as a fixed reference point on the ground. In the next implementation, the main goal is to remove the need for the

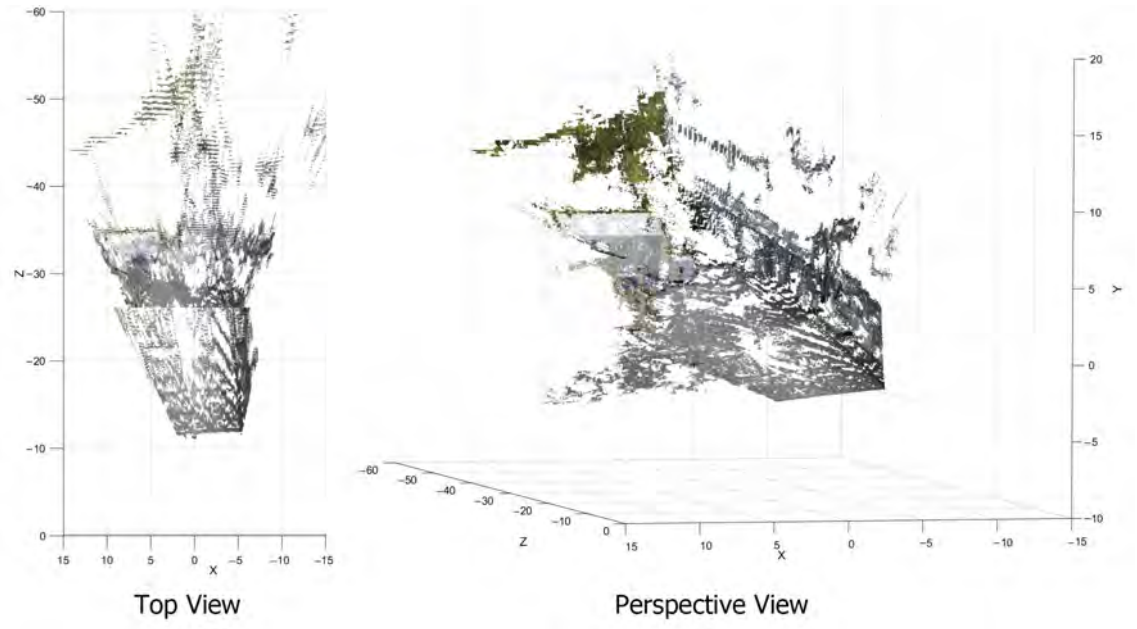

Figure 5.11: Resultant point cloud as a result of baseline fusion.

ground tracking station and implement the whole system using inside-out sensors.

The second point of consideration is the use of DJI Tello EDU UAVs. Tello is a very small UAV, such that the tested successful maximum payload is less than 100 grams, considering that this load is directly in the same vertical axis as the center of mass. This limited payload makes the additional mounting of equipment or sensors extremely difficult. Additionally, even though odometry data from the UAV is accessible, the method of odometry is unknown. No information is provided from the official side, but the speculation is that the Tello utilize optical flow tracking method from the sensors mounted below the chassis. If the speculation is correct, there is presumably no loop-closure detection in the algorithm. Figure 5.3, the odometry data shows no loop-closure adjustment. Compared to other tracking devices utilizing visual SLAM, the loop-closure detection adjustment can be clearly seen in the odometry data as a small jump in the pose data to compensate for the odometry drift. Therefore, further evaluation and utilization is difficult.

The third consideration is the test in an arguably small area. In this implementation, the system is intended to be the proof of concept. Therefore, the design of

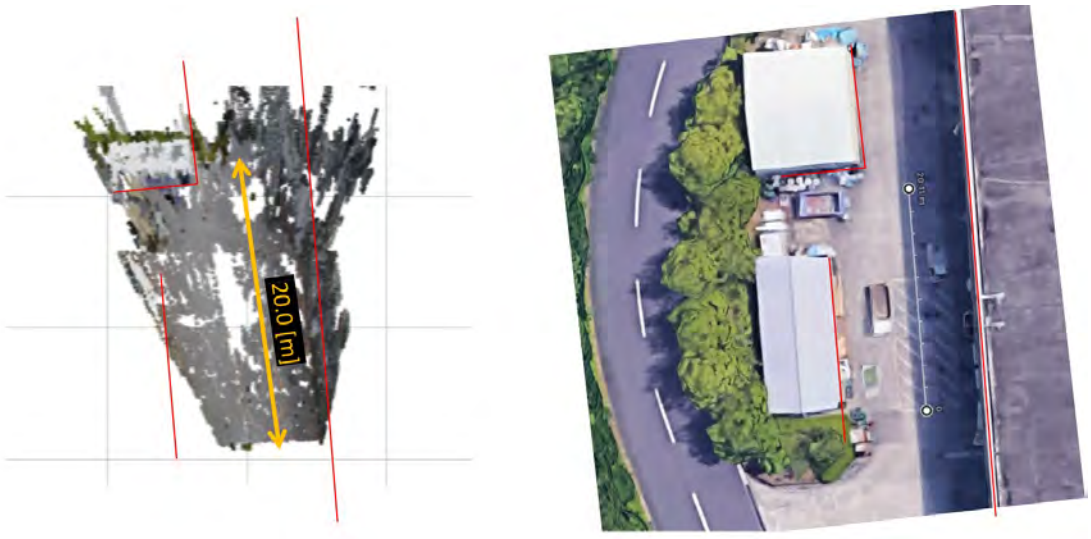

Figure 5.12: Resultant point cloud compared to the ground truth 2D map obtained from Google Maps.

the system is meant for a small area of a distance up to 50 meters. Therefore, a verification of the system in a much larger area, as the original intent of the research, is needed. To that end, a system revision of the system's hardwares is needed. A custom-made UAV platform to realize the scaled-up system is required.

Chapter 6

Implementation with inside-out sensors tracking

6.1 System overview

In this implementation, a final major change to the system is made in order to overcome the limitations of the earlier implementation. A prototype UAV platform is built purposely for the system. Figure 6.1 shows the components of the UAV. Table 6.1 shows the general specifications of the UAV.

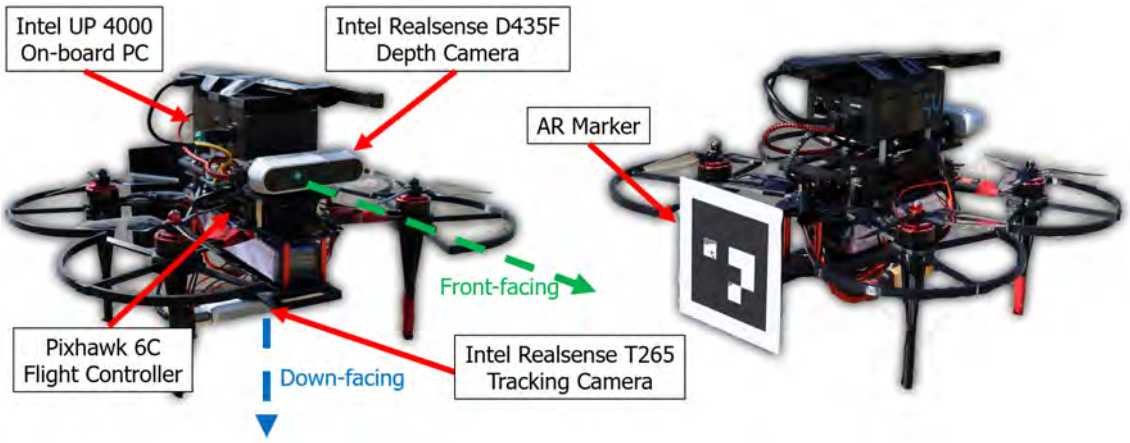

Figure 6.1: Prototype UAV components overview from the front (left) and from behind (right).

Table 6.1: Developed prototype UAV specification

Dimensions:	35.9 × 30.5 × 19 cm	Rotor Span:	25 cm
Flight Controller:	Pixhawk 6C	Firmware:	Ardupilot
Weight w. battery:	1.5 kg	Motor:	ARRIS 2205 (2300KV)
Propeller:	Inverted Tri-wing 5045	Total Thrust:	4 kg
ESC:	Holybro Tekko32 F4	Battery:	3S/4S

The prototype UAV is built based on the QAV250 frame, a small quadrotor frame intended for FPV flight. The reason for choosing the small frame is the same as the previous implementation, to make the smallest baseline as narrow as possible. The QAV250 frame parts are assembled upside down in order to fit more components on the top, and use the lower part to attach the battery. The landing gear, propeller guards, onboard PC mount, and camera mounts are designed and 3D printed to fit the frame. The screws of the frames are changed so that there is as little vibration as possible in the UAV that is caused by loose parts.

The UAV is controlled by a Pixhawk 6C flight controller with Ardupilot flight stack. The version of the flight stack used is Ardupilot 4.3.2. The reason for choosing Pixhawk firmware is their compatibility with ROS protocol having a MAVROS package that can reliably relay information from and to the flight controller out of the box. Another advantage of Ardupilot is the open source and compatibility for multiple kinds of sensors out of the box. For example, the PWM pins of the flight controller can be used as a normal GPIO pin to read data from an ultrasonic range finder. Finally, Pixhawk flight stacks support position control of the UAV using external position feedback. The internal EKF of the flight stack is already implemented to follow position commands. Therefore, the additional implementation of PID position controllers in earlier implementation is no longer required.

The external sensors used in this prototype are Realsense D435F depth camera, and Realsense T265 Tracking camera. The D435F is mounted front-facing and is

mainly used for image acquisition, and can be used for depth acquisition for near field objects of up to 10 meters if needed. The F variant of the D435 is the variant that uses a IR strobe to project an IR pattern, and uses a single IR camera to estimate the depth based on the projected pattern. This variant is optimized for outdoor uses even in the presence of sunlight. The settings for the D435F camera used in this implementation can be seen in Table 6.1. The down-facing T265 camera is a tracking camera with an IMU, which provides accurate odometry data with 1% error per movement distance. The T265 is used for localization of the UAV. The reason for choosing the down-facing orientation for the camera is that the UAVs are planned to be flown in the middle of an open space with no immediate features in the radius of more than 20 meters in any horizontal direction of the UAV. In this case, if the tracking camera is mounted horizontally, the tracking camera may struggle to estimate the depth of the features that are far away in the horizontal direction, and therefore the difficulty may affect the accuracy and robustness of the tracking odometry. Therefore, the T265 is attached to face downward, because the predicted altitude of the UAV will not exceed 20 meters. In this fashion, the features on the ground can be seen by the cameras, closer than other features in the horizontal direction, therefore aids the effectiveness of tracking for the camera.

Table 6.2: Realsense D435F settings and specification

Resolution:	640 × 480 pixels	Field of view:	56 degrees
Frame rate:	30 FPS	Exposure:	Automatic

An onboard PC, Intel UP 4000, is used as the processing unit. The UP 4000 is running Ubuntu 20.04 OS implemented with ROS Noetic. The reason for choosing UP 4000 as the onboard PC is its Intel x64 architecture processor, which provides powerful computational power with software compatibility with libraries and softwares that are implemented in the x64 architecture. The UP 4000 is connected to a USB WiFi device for communication with the control PC of the human operator. Furthermore, the sensors and the flight controllers are connected to the UP 4000. The UP 4000

acts as the data relay unit and onboard processing unit to process all the information needed for the flight of the UAV to be operational.

Furthermore, this implementation has the scalability in mind as the main functionality. The system is designed such that the number of UAVs can easily be increased and the system can operate beyond two UAVs with ease. Compared to the earlier implementation, this implementation uses a modular design such that the number of UAV can be increased with the same set of sensors, but not limited to the same frame size or hardware. As long as the main functionality is realized through the hardware, the system can be easily extended.

6.2 Communication and data flow

Figure 6.2 shows the block diagram of all the components in the system and their data flow. The main three parts of the system are, the control PC, the UAV, and the Operator. The control PC is operated by an operator on the ground. The control PC used in this case is a laptop Asus GL522VW. The control PC utilizes AMD Ryzen 7 6800H, 4.7 GHz, 16 logical core processor, and 16 GB of RAM. The control PC is used for sending commands as well as status monitoring of the UAVs. The control PC is hosting a network time protocol (NTP) server [50] so the UAVs can sync their time with the NTP server. The onboard PC on the UAV is used for communication between components on the UAV as well as relaying the data to the control PC via WiFi. The WiFi protocol used in this case is IEEE 113 802.11ac 5.0GHz connected to a 5.0GHz gigabit router. The operator part is the control using a radio transmitter to control the UAV manually, when the UAV is not in the automatic mode.

The control PC runs the control UI software that is used for command and monitoring of the UAVs in the system. Figure 6.3 shows the interface of the control software. The software is implemented in the PyGame library. The control UI for setpoint control and monitoring are shown in the figure. The first component (1) is the stereo configuration list table, and (3) is the list of all configuration files, each obtaining a set of stereo configurations. When the button in (3) is clicked, the con-

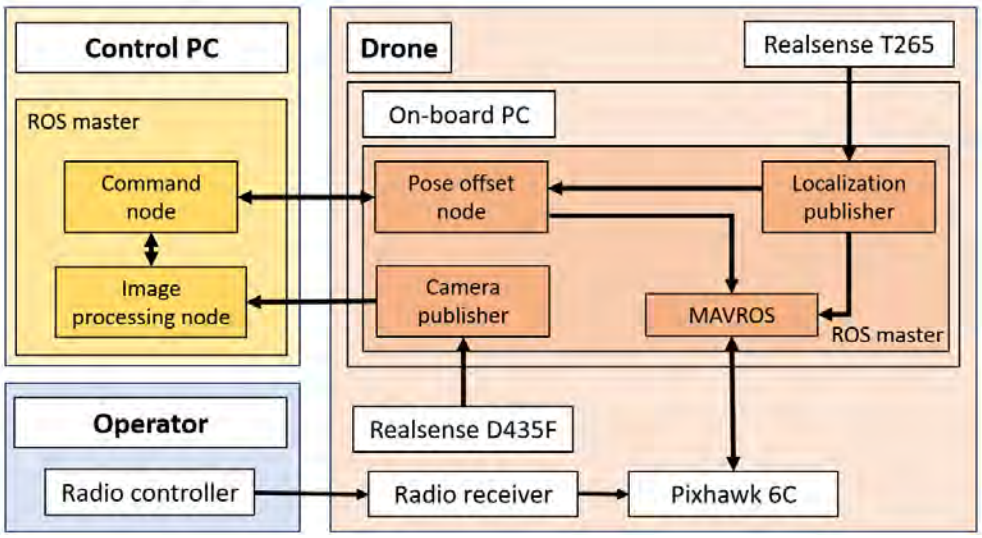

Figure 6.2: Block diagram of the communication and data processing nodes.

figuration file is loaded and the configurations in the file are reflected in (1). Each component of the configuration represents, setup axis, baseline, tilt angle, and image recording time. Component (2) is the configuration selection control. The up and down arrow selects the configuration in the list, when the middle "Execute" button is pressed, the UAV will execute the configuration. Component (4) is the FPS counter of the UI. Component (5) is the connection and general status display. The display includes the callback rate of the pose message from each UAV's T265 camera, arming status, and flight mode. Component (6) is the set home button, used for setting the stereo vision origin point. The detailed use will be described in the next section. Component (7) is the record and calibrate button. The calibrate button is used for initiating pose calibration which is described in detail in the next section. The record button is to start the recording of images and poses retrieved from the UAVs. The record duration is set in the configuration file. Component (8) is the pose data display, which displays the pose of each UAV in different reference frames. This is used for monitoring the pose tracking of the UAV, both individually and w.r.t. to other UAVs in the system. Component (9) shows the image from the front-facing camera and its FPS. Used for monitoring the images taken.

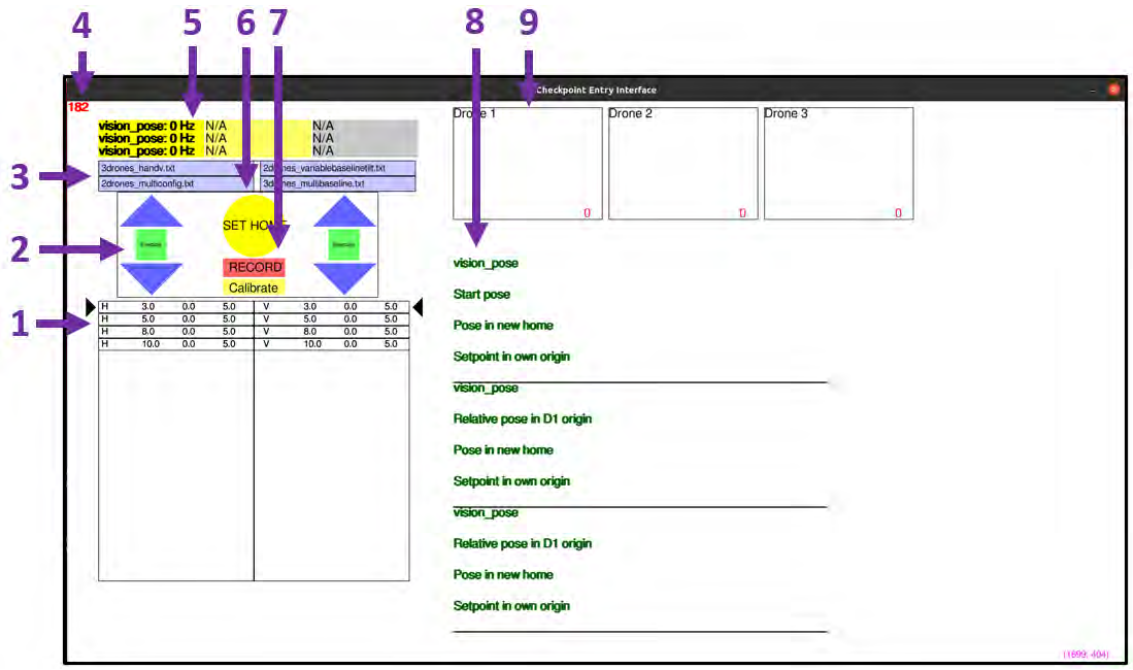

Figure 6.3: Control UI node implemented in this implmentation.

The point of consideration in the system is the ROS master which at each PC in the system runs its own ROS master. In each UAV, the onboard PC runs its own ROS master to relay information between sensors and the flight controller. Likewise, the control PC runs its own ROS master. The crucial reason for each component to have their own ROS master instead of a shared master is safety. Even though it might complicate the overall information communication between the UAVs and the control PC, this implementation ensures that pose data from the T265 tracking camera is always relayed to the flight controller through the local ROS master on the onboard PC. Such that the UAVs can continue to fly manually using a transmitter even if it does not receive commands from the control PC, or the WiFi communication to the control PC is unstable. Each ROS master on each different machine is communicated to each other through a multi master data synchronization package [51]. Only a selection of necessary topics is communicated to reduce the limited bandwidth of data being transferred over WiFi.

6.3 Tracking method

In this implementation, the tracking method aims to rely only on the inside-out sensors on each individual UAVs for tracking and relative pose measurement. This implementation completely removes the need for a ground station and utilizes the tracking scheme based on AR markers and localization odometry of the T265 camera.

The localization of each individual UAV is, in the similar fashion to the previous implementation, based solely on the tracking odometry data from its own T265 tracking camera. The odometry data from the T265 camera is relayed inside the onboard PC through MAVROS to the flight controller, which enables the flight controller to obtain the pose feedback and allows the flight controller to operate in position control modes. This odometry is used for localization of each UAV in their own localization origin. This origin is referred to as O_n, where n is the number of the UAV, such as in a two UAV system, O_1 and O_2 exists for UAV D_1 and D_2.

In order to calculate the relative pose of the UAVs in the system, the AR marker pose estimation is once again utilized. From Figure 6.1, at the back of the UAV, an AR marker can be seen mounted facing backward. A unique AR marker is used for each of the UAV in the system. In order to measure the relative pose of each UAV, the UAVs are commanded to perform a pose calibration process at least once per operation. Pose calibration is performed at least once after take-off using the AR marker attached on the UAV directly in front of it. At the beginning, the UAVs are taken-off manually. The UAVs are then controlled manually to form a column and hold their position, such that the AR marker of the UAV in front is visible in the front-facing camera of the UAV behind. Pose of the AR marker of the UAV in front is acquired by marker pose estimation. The pose obtained is used for the relative pose calibration of the UAVs by calculating the offset of the localization pose origin of the UAVs based on the AR marker pose. The following equation is used for the calculation:

$$\begin{aligned} R_{O_{n-1}\rightarrow O_n} &= R_{D_n}\ R_A\ R_{D_{n-1}}^T \\ T_{O_{n-1}\rightarrow O_n} &= T_{D_n}\ +\ R_{D_n}\ (T_{C_n} - R_A\ (T_A + T_{D_{n-1}})) \end{aligned} \tag{6.1}$$

Where T_{D_n} and R_{D_n} is the position and orientation of the drones obtained from the T265 tracking camera, and Figure 6.4 shows the notation of each coordinate frame.

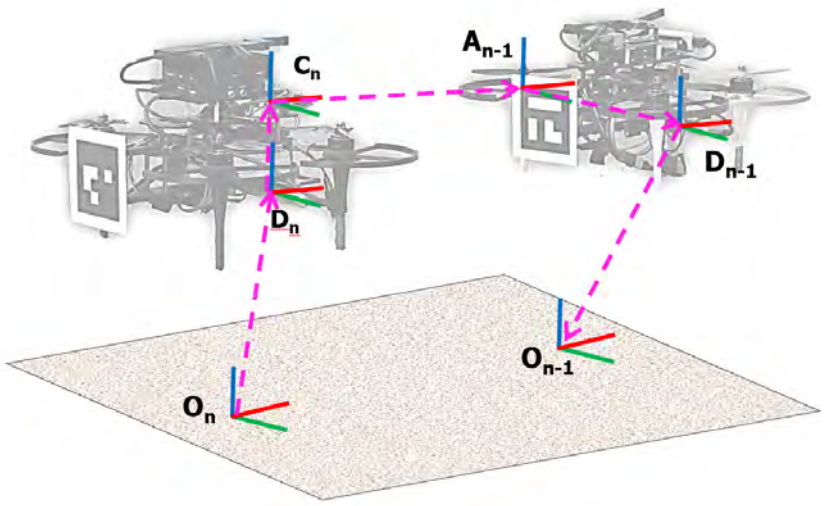

Figure 6.4: Coordinate frames of each UAV in the calibration process.

In this fashion, the $R_{O_{n-1} \to O_n}$ and $T_{O_{n-1} \to O_n}$ can be used to calculate the current pose of each UAV in their preceding UAV's localization frame. This step is carried out for all the UAVs and at the end, the localization origin of UAV 1 is used as the tracking reference point of the UAVs, and thus the pose discussed after this is in the coordinate frame of O_1.

6.4 Control method

The control method of this implementation is divided into multiple stages. The first stage of control is the manual takeoff by the operator. In this case, the operator needed to control the UAVs are equal to the number of the UAV used, for safety reasons each individual is responsible for controlling the UAV until they enter the autonomous flight phase. The UAVs are taken off in Altitude mode. The reason for taking off in Altitude mode instead of Loiter mode is the safety concern. Since the tracking camera is facing downward, there is a chance that when the Loiter

mode is used, the position feedback largely jumps as the UAV lifts off the ground and the altitude measurement from the tracking camera may be incorrect. In our experience, the altitude drift during takeoff has been observed at a maximum of 30 meters, meaning that the measurement of the altitude of the UAV is 30 meters below the actual altitude in the real world. At which point, the UAV would quickly try to climb 30 meters into the air, causing a safety concern.

After the takeoff, the UAVs are then put into Loiter mode. In Loiter mode, the UAV uses the position feedback to hold a position and the orientation feedback to hold the yaw, and the operator uses the transmitter to control the position setpoint in 3-DOF and the yaw setpoint. When the control sticks are centered, the UAV holds its position even if there is an external force such as wind gusts. In contrast to normal Altitude Hold mode, which the UAV only holds its altitude and yaw, but not the planar position. When the control sticks are centered, the UAV holds its attitude at roll and pitch zero. The UAV can still move if there is an external force acting on it or if the balance of the UAV causes the thrust in each motor to be unequal and the thrust pushes the UAV to a direction.

After switching to Loiter mode, the UAVs are controlled manually to as near as possible to the starting point of the image acquisition position. The UAVs are then aligned into a column for pose calibration as discussed in the earlier section. After the alignment, the operator confirms in the screen of the control PC that the AR marker of the UAVs in front is reflected in the image from the frontal camera of all UAVs then the operator initiates the pose calibration process. After calibration, all the UAVs relative pose are known.

After the calibration, UAV 1, the reference UAV is moved to a starting point for image acquisition by the operator. After reaching the starting point where the intended area of interest fully reflects in the image of the UAV 1. The operator then set this position of the UAV 1 as the home position, or the stereo vision origin position. All the movements of the other UAVs in each configuration are then based on this position as the origin point. After that, the image acquisition phase starts.

All the UAVs are then put to Guided mode, the UAV holds the position and yaw

based on the pose feedback like Loiter mode. But in guided mode, the UAV cannot be controlled by the transmitter. The setpoint, in this case position setpoint, is sent to the UAV via MAVROS protocol and the UAV would follow the setpoint and hold at the command setpoint. The setpoint loaded in the control software is executed so the UAVs would move to the image acquisition positions and hold at that position. The images are then acquired in the control software and processed in the image processing node. Furthermore, the images that fit the positional requirement and their respective raw odometry and their position wr.t. to the home point are published in ROS topics. To reduce the processing load during the flight which may cause errors and safety concerns, these data are saved in a ROS bag to be post-processed after the flight.

6.5 Rectification method

The rectification method in this implementation is revised to fit the characteristics of the system. The rectification methods in the earlier implementation are modified. In this implementation, instead of rectifying all the images from the other UAVs to be in the same parallel plane as the camera of UAV 1 at all times, the images are rectified such that it is parallel with the position and yaw of the home point and the roll and pitch of the level horizon. The main reason for this change is the change in camera characteristics in-flight. In this implementation, the front-facing camera used for the image acquisition is largely affected by the vibration of the UAV such that the image frame is always vibrating. It is redundant for the images from other UAVs to be rectified to match the vibration of this camera. Therefore, another given fixed point, which is the home position, is selected as the reference point of the rectification.

With that in consideration, the rectification matrix can be easily calculated using:

$$R_{rectn} = R_n R_{O_h \to O_n} \tag{6.2}$$

Where $R_{O_h \to O_n}$ is the rotation from home pose h to the localization origin O_n of UAV n. This can be calculated using the following equation:

$$R_{O_h \to O_n} = R_{O_{n-1} \to O_n} \; R_{O_{n-2} \to O_{n-1}} \; ... \; R_{O_2}^{O_1} \; R_{O_1}^{O_h} \tag{6.3}$$

Using the calculated rectification matrix, the images are then rectified to match the home position. The vibration of all the cameras can be measured from the IMU of the T265 camera such that most of the vibration is included in the tracking localization data. The rectification based on the tracking data can mostly compensate for the vibration. In contrast to the earlier implementation, the images obtained from the Tello are already stabilized such that the vibration from the UAV is already eliminated, so in the earlier implementation the rectification method is different.

6.6 Stereo matching and point cloud projection

Similar to the previous implementation, the stereo matching algorithm used in this implementation is Quasi-Dense matching algorithm. Most of the image processing processes are the same up until the point cloud processing. The main difference in this implementation is the coordinate system chosen for the point cloud projection. Point clouds, instead of combining at a reference point on the ground, which does not exist in this case, the point clouds are rotated and translated such that they are in the coordinate system of the home position. Alternatively, they can be translated to O_1 if needed, in an ideal case this O_1 is the takeoff point of the UAV 1. However, in reality, as discussed in earlier sections, there is a chance that the odometry of the UAVs drifts when they take off so the origin is not the takeoff point anymore. Therefore, the most effective point that the point clouds are translated to is the only known and confirmable position in the odometry, the home position.

Another change in the image processing part is the baseline fusion algorithm. The baseline fusion algorithm proposed in section 3.2 is the algorithm that determines the length of baseline used as well as determining the cutting point for each baseline. In this implementation, the baseline fusion algorithm is modified such that the baseline size used are whole numbers. The reason for change is to be able to easily monitor the movement of the UAV during their operation compared to a decimal number. Furthermore, the cutting point in this implementation is not limited only to the error threshold, but also depends on the objects in the scene. Since the scene mapped in

this implementation is very big, therefore many objects are included in the scene. If the trimming point for each baseline cuts in the middle of the object of interest, there is a possibility that a discontinuity may arise in the resultant map for that specific object. Therefore, the cutting point in this implementation focuses on the organization to fit each object of interest in the range of one baseline if possible.

6.7 Mapping experiment

In order to verify the usability of the proposed system, the outdoor experiment is carried out to map a large outdoor area using the proposed system. In this experiment, two prototype UAVs are used. Figure 6.5 shows the time lapse of the movement of the UAVs in the experiment. The UAVs are taken off manually and controlled to near the home position. The calibration is carried out mid-air. After the calibration, the home position is set and the Guided mode for both UAVs are activated.

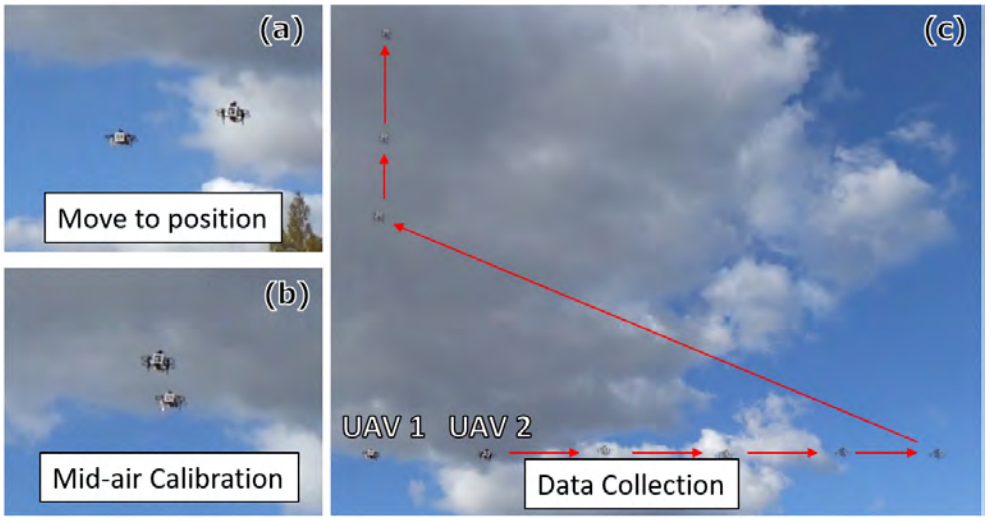

Figure 6.5: A time lapse of mapping process including (a) UAVs moving to a starting point manually, (b) pose calibration using AR marker, and (c) stereo images collection process.

The first UAV holds its position at the home position, the second UAV moves to each configuration pre-defined. The configuration set points include, 2, 4, 6, 8 and 10

meters for horizontal setup, and 5, 7, and 10 meters for vertical setup. UAV 2 holds at each position and collects the images. The images collected and their corresponding disparity image are shown in Figure 6.6 and Figure 6.7.

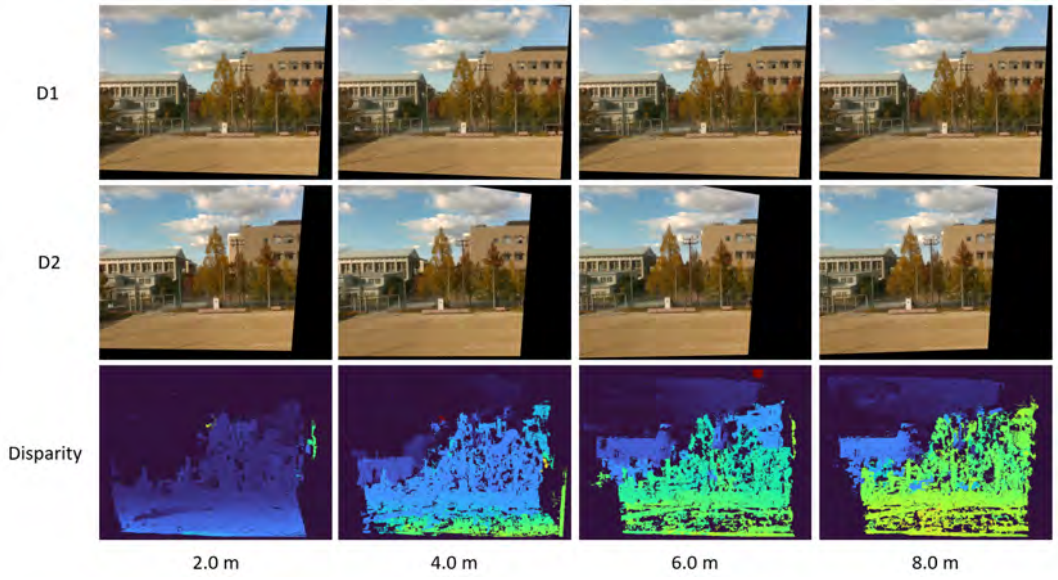

Figure 6.6: Disparity images and their image pairs from each corresponding baseline using horizontal configuration.

Due to the high amount of occlusion and noises in disparity acquired from horizontal setup using 10 m baseline caused by excessive baseline distance, data collected from the configuration is excluded from the result. The shown disparity images are filtered in the range of 16 to 256 pixels. From each disparity, the point clouds are projected.

In order to evaluate the depth estimation accuracy of the algorithm, several points of interest (POIs) are defined in the area of interest as shown in Figure 6.8. At each POI, a square of the size of 160×30 pixels is drawn and its z-axis depth is extracted. Their depth estimations are then statistically evaluated.

Figure 6.9 shows the evaluation result of the POIs. Two parameters are used for the evaluation. The first parameter is the depth estimation of each pixel in the POI, and the second parameter is the number of non-null, namely the successfully matched points, in each sampling area of the POIs from each configuration.

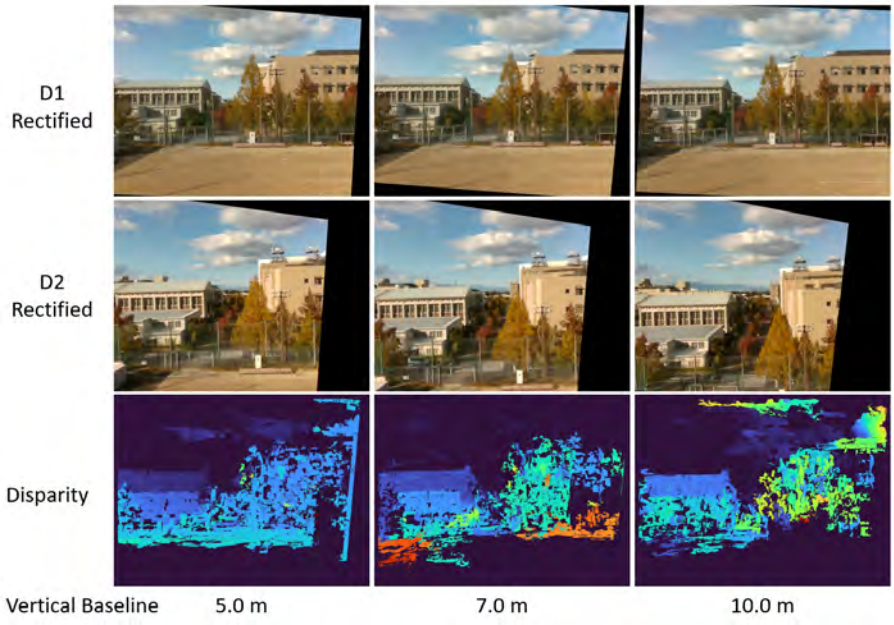

Figure 6.7: Disparity images and their image pairs from each corresponding baseline using vertical configuration.

Starting from the *POI A*, the red colored graph shows the depth estimation of the building. In horizontal setup, the depth estimation from 2 m and 4 m baseline are excluded due to the building being below the disparity threshold of 16 pixels due to the building being too far. The depth estimation of the remaining configurations can be seen approaching 100 meters, which is corresponding to the ground truth seen in Figure 6.10 obtained from [52]. The successfully matched area is considered high, with all exceeding 90%. *POI A* can be an example of a good match where the distribution of depth estimation error is narrow and the match percentage is high.

POI B is the green color in the graphs, the depth evaluation of a tree line. The ground truth distance of the tree line is about 50 meters. The depth estimation of the *POI B* can be seen as slightly more varied compared to *POI A*. The match rate also varies. This is caused by the non-uniform and the shape of the trees that are full of dimples and dents. This made the depth estimation more varied and the successful match rate lower. Another phenomenon in this POI is the match rate that decreases as the baseline increases. This is caused by excessive baseline. As the

Figure 6.8: Rectangles denoting depth sampling area of three areas of interest in the mapping process, POI A (red), POI B (green), and POI C (blue).

baseline gets too large, the match rate reduces. This POI is a good example of the effects of excessive baseline.

POI C is the blue colored graphs. The match rate is extremely low and varies and the depth estimation is even more largely distributed. This is caused by the POI having very low texture. Therefore, depth estimation is extremely difficult. This is an example of an object with low texture and is difficult to process.

In the statistics of all the POIs, the accuracy of the depth estimation can be seen approaching the ground truth value when the baseline distance used is not too wide or too narrow. When the baseline used is too narrow or too wide, the depth estimation can be seen notably lower or higher than the actual ground truth value. With this information, the appropriate point cloud from an appropriate baseline can be estimated for use in the point cloud fusion.

Finally, the point cloud from each baseline and setup are merged together into a resultant point cloud. The trimming distance is based on the estimated distance of the POI. For the horizontal setup of baseline 2, 4, 6, and 8 m, the point cloud of each baseline is trimmed at the distance of 30, 60, 90, and 120 m respectively. For vertical

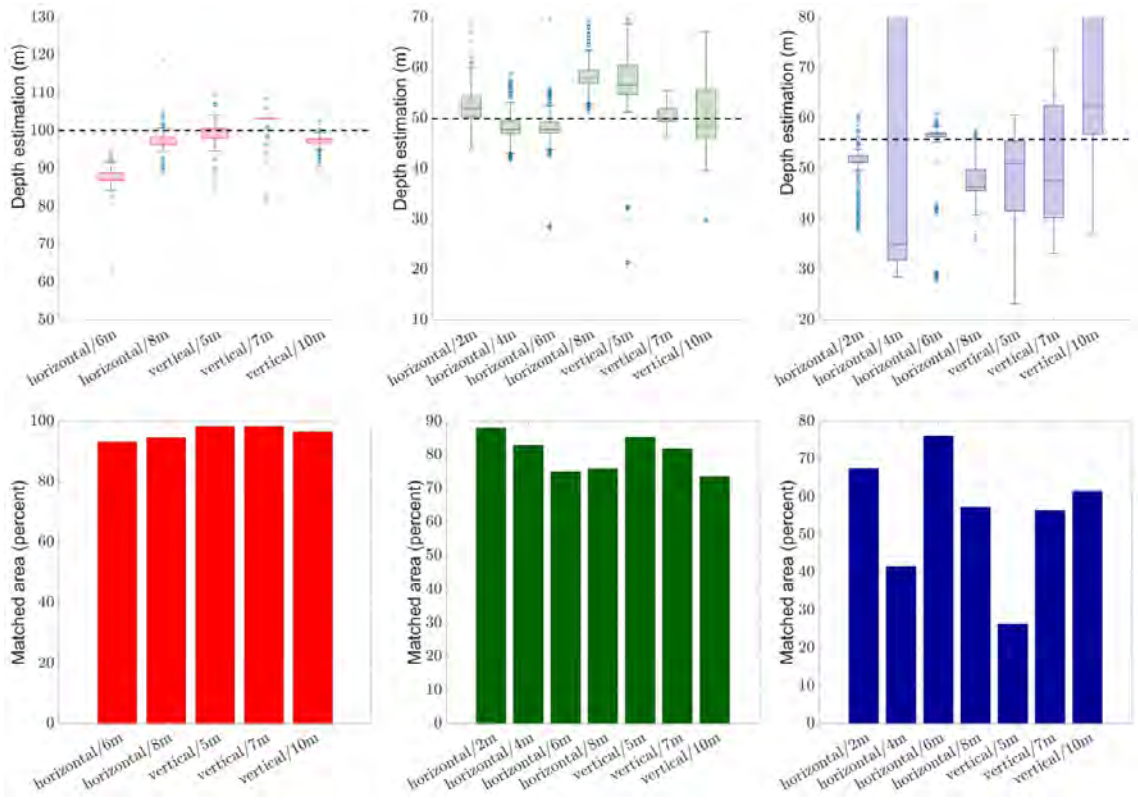

Figure 6.9: Depth estimation and their corresponding match percentage of area of interest POI A (red), POI B (green), and POI C (blue) and their ground truth value in dashed black line.

distance, point clouds of 5, 7 and 10 m baseline are trimmed at the distance of 40, 80, and 120 m respectively. The reason for choosing the specific distance is the distance of POIs. The closest POI is the field, which has the distance of roughly up to 40 m, *POI B* which has the distance of roughly 50 m, *POI C* building which has the distance of roughly 65 m, and the *POI A* of the distance of roughly 100 m. In this experiment, the operator tries to use each baseline for each POI, by specifying the trimming point as shown, the distance of the POIs will fall in the range of different baselines, and not in between.

The resultant point cloud is shown in Figure 6.11. Though the point cloud represented using 2D image is difficult to evaluate, the 3D view of point cloud data is provided in the supplementary video. The evaluation of the resultant point cloud

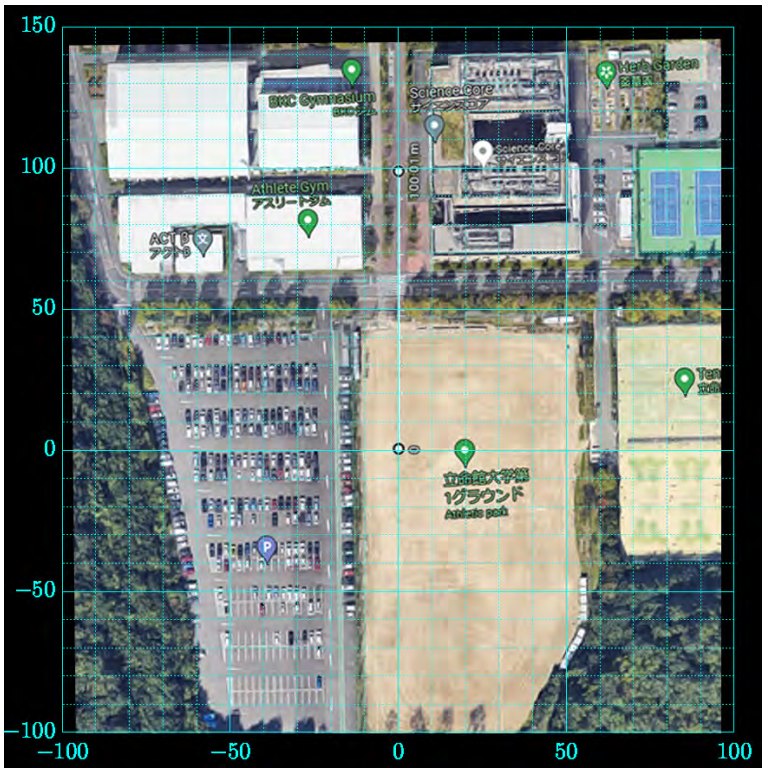

Figure 6.10: Ground truth acquired from Google Maps of the area of mapping fitted with a distance grid.

against the ground truth planar data acquired from Google Maps is shown in Figure 6.12. Colored lines are drawn on the POIs to compare the POIs in ground truth and the resultant point cloud. Depth estimation is observed to be relatively accurate, where there are some positional errors due to the imperfection in the rectification process caused by camera vibration.

6.8 Issues of the system and future work

In the prototype system, one of the most noticeable issues is the communication issue. In this implementation, the communication device used on the onboard PC is the WiFi USB device with the 802.11ac 5.0GHz standard, which has the theoretical speed of 1.3 Gbps. In the testing environment where the onboard PC is next to a

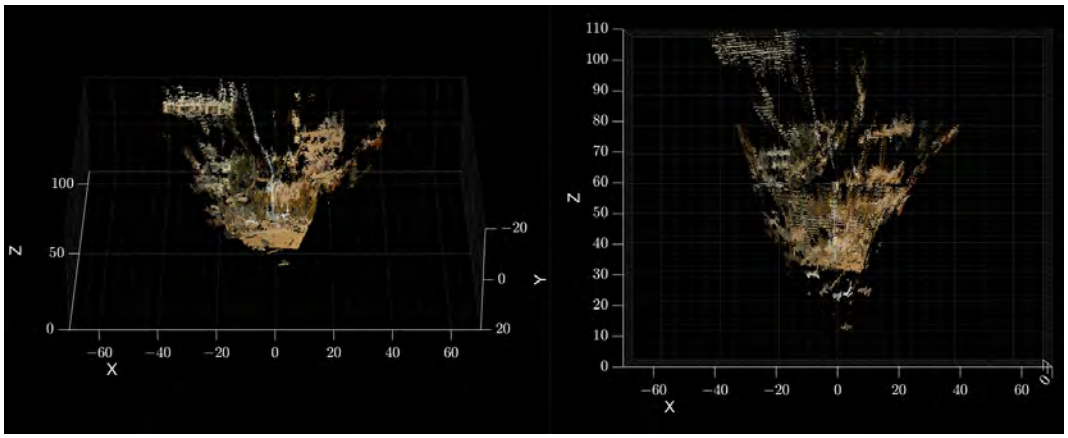

Figure 6.11: Resultant point cloud seeing from pitch minus 60 degrees (left) and directly above (right).

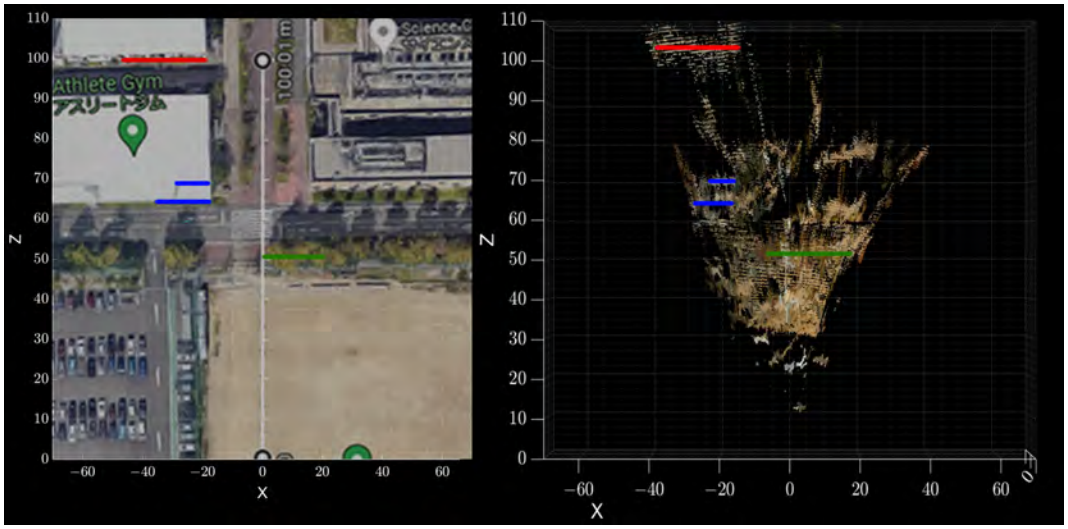

Figure 6.12: Ground truth acquired from Google Maps (left) and resultant point cloud from point cloud fusion (left)

router, the maximum speed obtained from this specific setup is less than 500 Mbps with neglectable packet loss. On the field during the mapping process, the actual speed when the UAVs are operating is less than 300 Mbps with high packet loss such that some data is lost in the communication. Compared to the data being transferred to the control PC, the largest part of the data bandwidth is the raw image data from the front-facing camera, which takes up to about 222 Mbps of bandwidth. The rest

of the communication adds up to less than 5 Mbps. Due to the large bandwidth of the image data and the packet loss, there are moments where image messages are not reaching the control PC in a duration of a few seconds. This problem caused the image display in the control software sometimes unavailable. The future improvement that will potentially aid this problem is either to reduce the amount of data or the change to a better hardware capable of more bandwidth. One approach to reduce the amount of data is to compress the image data into a specific encoding, and the encoding can greatly reduce the bandwidth needed to transfer image data to an estimate of less than 10 Mbps. However, this would add an amount of delay to the data transfer due to the encoding process, and furthermore the image has to be transferred over a protocol other than ROS such as a media streaming protocol such as Real-Time Messaging Protocol (RTMP) . Alternatively, the change in hardware is another option to fix the instability of the communication. However, the alternative hardware for such bandwidth is not common. A dedicated communication module may be needed.

The second issue is the camera vibration caused by the vibration in the UAV. In contrast to DJI Tello, the custom made prototype UAV produces much more vibration due to much higher thrust and much more components attached to the UAV [53]. In the current implementation, most of the vibration is detectable by the T265 camera which contains an IMU and a high frequency odometry. Therefore, the vibration detected by the T265 is used to rectify the images which quite effectively eliminate most of the major vibration. However, some small vibration still exists in the image. In order to completely eliminate vibration, in addition to the odometry data, a software stabilization method can be used for the purpose.

The final future improvement is the use of depth information from the front-facing depth camera. In the current implementation, the depth information of the D435F is not used at all. In the future work, the depth information can be used in combination with a visual inertial SLAM [18] or similar technique to create the map of the near field objects that's closer than the range of the proposed system. For example, the outdoor experiment carried out in this implementation has the minimum range of around 30 meters as observed in the resultant point cloud. The area closer than

that can be mapped using SLAM method and the depth camera during the UAV movement during image acquisition process or after all the images are acquired. The map created from SLAM can be easily integrated into the resultant point cloud using the localization information of the UAV doing the mapping's T265 camera.

Chapter 7

Conclusion

7.1 Summary

This work proposed a rapid and movement efficient 3D mapping and monitoring method using flexible configuration stereo vision with multiple UAVs. The proposed work includes the analysis of using a variable baseline stereo vision system and the use of horizontal and vertical setup, as well as exploring the merit of inward tilting in large baseline stereo vision. This work proposed a baseline fusion algorithm which is effectively used to evaluate the point clouds generated from different baselines, choose the effective range of each baseline, and combine the depth generated from each baselines into a high accuracy and low noise resultant map. An algorithm for depth image fusion of vertical horizontal stereo is proposed, along with the guidelines for obtaining and rectification of the image pairs specifically for the flexible configuration stereo vision system. The simulation result has shown an effective use of inward tilting angle to reduce the occlusion from excessive baseline. Additionally, the simulation has shown a highly accurate long-range map creation of the range up to 400 meters with the movement of only 10 meters and the image acquisition and depth processing time of 1 minutes 46 seconds.

The prototype of the system has been developed in three implementations. The first implementation as a proof of concept of using the proposed system for ground robot guidance application. Two DJI Tello UAVs are used in the implementation.

The tracking system of the first implementation uses two monocular cameras on gimbals to track the UAVs with AR markers. The system opens the perspective of the issues and lessons for further developing the proposed system.

The second implementation utilizes the motion capture camera as the tracking device and the use of UAVs' odometry in combination with the tracking device for relative pose tracking, enables the real-time tracking of the UAVs even when they are out of the field of view of the tracking camera. The second implementation is used for an outdoor mapping experiment which proves the usability of the system in a small area of a range up to 60 meters.

The prototype custom made UAVs are developed for the third implementation. The tracking system is revised to utilize only the inside-out sensors, completely eliminating the requirement for a ground tracking station. The tracking method is based on the odometry of the tracking camera, in combination with calculation of relative pose using AR marker pose estimation. The larger frame used in this prototype enables a larger scale outdoor mapping experiment conduction in Ground 1 in the Ritsumeikan University BKC campus. The outdoor mapping experiment achieved the depth estimation of a range of up to 100 meters with the movement distance of 20 meters and the total process time of 2 minutes and 30 seconds. The final implementation opens the possibility of increasing the number of UAVs and the use of more than two UAVs by the implementation of a modular design.

7.2 Discussions and future work

In the future work, in addition to problem solving mentioned in section 6.8, further use of the flexible configuration stereo vision can be explored. The first possibility of future work is the estimation of excessive baseline. An algorithm for estimating the largest baseline that can be used to map an object at a specific distance is one of the key pieces of information that is unable to be achieved in this work. An attempt to estimate such information has been made by utilizing the inverse function of the estimation error equation 2.12 has been unsuccessful. Another approach such as the

use of neural networks to determine the parameters to calculate for the maximum baseline distance is envisioned.

The implementation of the baseline fusion algorithm is one of the key achievements in the proposed work due to its ability to selectively choose the effective part of the point clouds from different baseline distances. However, the issue of the proposed baseline fusion algorithm is the hard-trimming and discrete fusion property of the algorithm. Each point cloud is cleanly trimmed at the trimming point and the point beyond the trimming point is not at all utilized for the resultant point cloud. Only the selected range is combined into the resultant point cloud which may create a discontinuity. In order to improve the baseline fusion algorithm, a non-discrete method that fuses the point cloud, with some degree a utilization of some part of the previous or the next point cloud. For example, a confidence based method where each data point or depth range is given a confidence value of how accurate this depth estimation is. Based on the confidence value, the points are then combined based on the weight of the confidence value. Although, the missing parameter to implement the confidence based fusion algorithm is the estimation of the excessive baseline mentioned previously. The upper bound of the effective depth range can already be estimated by the proposed baseline fusion algorithm but the lower bound cannot yet be estimated. The proposed baseline fusion algorithm works on an assumption that the upper bound of the previous baseline's effective range is more than or equal to the practical lower bound of the next baseline.

The final future development is the adaptation of the flexible configuration in a micro scale. A prototype of a tilting stereo camera is proposed and shown in Figure 7.1. In contrast to normal fixed stereo cameras, a tilting 1-DOF stereo camera is proposed. The proposed system utilizes the use of inward tilt in order to reduce the minimum depth estimation distance of small objects. The system can potentially be used for small objects detection at close distance utilizing the tilting property that increases the effectiveness of stereo matching of near field objects. The potential application of the proposed prototype is the use for depth estimation of power lines or values. At long range, the stereo camera can operate at a normal configuration

with the two cameras parallel. As the object of interest moves closer to the camera, at one point the object will be occluded due to excessive baseline. At which point, the camera tilts inward and the occlusion reduces, such that depth can be estimated. The advantage of this system is the use of only one camera for depth estimation of both far distance and close distance without the reduction of depth estimation accuracy. Furthermore, instead of changing the baseline which needs to translate the camera, the proposed prototype only needs to rotate the camera on one axis which reduces the moving part in the system.

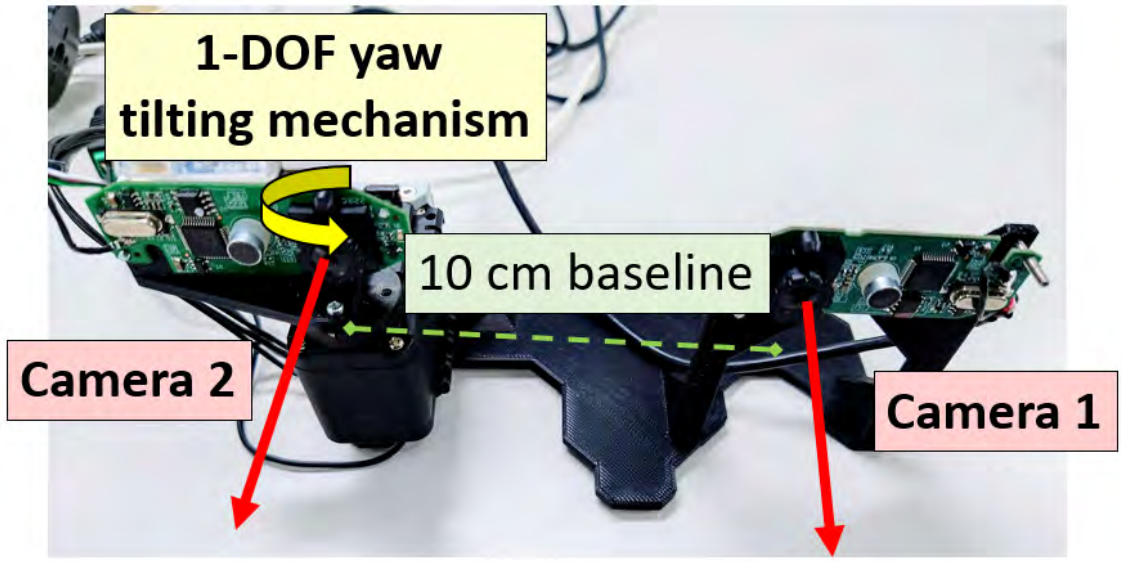

Figure 7.1: Prototype tilting mechanism in a small baseline stereo camera.

References

[1] J. F. Keane and S. S. Carr, "A brief history of early unmanned aircraft," *Johns Hopkins APL technical digest*, vol. 32, no. 3, pp. 558–571, 2013.

[2] P. Mátyás and N. Máté, "Brief history of uav development," *Repüléstudományi Közlemények*, vol. 31, no. 1, pp. 155–166, 2019.

[3] F. Nex and F. Remondino, "Uav for 3d mapping applications: A review," *Applied geomatics*, vol. 6, pp. 1–15, 2014.

[4] D. C. Tsouros, S. Bibi, and P. G. Sarigiannidis, "A review on uav-based applications for precision agriculture," *Information*, vol. 10, no. 11, p. 349, 2019.

[5] F. Ruggiero, V. Lippiello, and A. Ollero, "Aerial manipulation: A literature review," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1957–1964, 2018.

[6] A. Bauranov and J. Rakas, "Designing airspace for urban air mobility: A review of concepts and approaches," *Progress in Aerospace Sciences*, vol. 125, p. 100 726, 2021.

[7] C. Lee, S. Kim, and B. Chu, "A survey: Flight mechanism and mechanical structure of the uav," *International Journal of Precision Engineering and Manufacturing*, vol. 22, no. 4, pp. 719–743, 2021.

[8] M. Sabour, P. Jafary, and S. Nematiyan, "Applications and classifications of unmanned aerial vehicles: A literature review with focus on multi-rotors," *The Aeronautical Journal*, vol. 127, no. 1309, pp. 466–490, 2023.

[9] I. Colomina and P. Molina, "Unmanned aerial systems for photogrammetry and remote sensing: A review," *ISPRS Journal of photogrammetry and remote sensing*, vol. 92, pp. 79–97, 2014.

[10] G. J.-P. Schumann, J. Muhlhausen, and K. M. Andreadis, "Rapid mapping of small-scale river-floodplain environments using uav sfm supports classical theory," *Remote Sensing*, vol. 11, no. 8, p. 982, 2019.

[11] Q. Li, H. Huang, W. Yu, and S. Jiang, "Optimized views photogrammetry: Precision analysis and a large-scale case study in qingdao," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 16, pp. 1144–1159, 2023.

[12] S. Śledź, M. W. Ewertowski, and J. Piekarczyk, "Applications of unmanned aerial vehicle (uav) surveys and structure from motion photogrammetry in glacial and periglacial geomorphology," *Geomorphology*, vol. 378, p. 107 620, 2021.

[13] M. Room and A. Anuar, "Integration of lidar system, mobile laser scanning (mls) and unmanned aerial vehicle system for generation of 3d building model application: A review," in *IOP Conference Series: Earth and Environmental Science*, IOP Publishing, vol. 1064, 2022, p. 012 042.

[14] A. A. Setyawan, M. I. Taftazani, S. Bahri, E. D. Noviana, and M. Faridatunnisa, "Drone lidar application for 3d city model," *Journal of Applied Geospatial Information*, vol. 6, no. 1, pp. 572–576, 2022.

[15] H. Meng, G. Wang, Y. Han, Z. Zhang, Y. Cao, and J. Chen, "A 3d modeling algorithm of ground crop based on light multi-rotor uav lidar remote sensing data," in *2019 IEEE International Conference on Unmanned Systems and Artificial Intelligence (ICUSAI)*, IEEE, 2019, pp. 246–250.

[16] B. Martinez Rocamora Jr, R. R. Lima, K. Samarakoon, J. Rathjen, J. N. Gross, and G. A. Pereira, "Oxpecker: A tethered uav for inspection of stone-mine pillars," *Drones*, vol. 7, no. 2, p. 73, 2023.

[17] M.-A. Leclerc, J. Bass, M. Labbé, *et al.*, "Netherdrone: A tethered and ducted propulsion multirotor drone for complex underground mining stopes inspection," *Drone Systems and Applications*, no. ja, 2023.

[18] M. Labbé and F. Michaud, "Rtab-map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation," *Journal of field robotics*, vol. 36, no. 2, pp. 416–446, 2019.

[19] S. Hummel, A. Hudak, E. Uebler, M. Falkowski, and K. Megown, "A comparison of accuracy and cost of lidar versus stand exam data for landscape management on the malheur national forest," *Journal of forestry*, vol. 109, no. 5, pp. 267–273, 2011.

[20] T. Taketomi, H. Uchiyama, and S. Ikeda, "Visual slam algorithms: A survey from 2010 to 2016," *IPSJ Transactions on Computer Vision and Applications*, vol. 9, no. 1, pp. 1–11, 2017.

[21] M. Okutomi and T. Kanade, "A multiple-baseline stereo," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 15, no. 4, pp. 353–363, 1993.

[22] G. Bradski and A. Kaehler, *Learning OpenCV: Computer vision with the OpenCV library.* " O'Reilly Media, Inc.", 2008.

[23] Z. Zhang, "A flexible new technique for camera calibration," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 22, no. 11, pp. 1330–1334, 2000.

[24] J.-Y. Bouguet and P. Perona, "Camera calibration from points and lines in dual-space geometry," in *Proc. 5th European Conf. on Computer Vision*, Citeseer, 1998, pp. 2–6.

[25] D. Gallup, J.-M. Frahm, P. Mordohai, and M. Pollefeys, "Variable baseline/resolution stereo," in *2008 IEEE conference on computer vision and pattern recognition*, IEEE, 2008, pp. 1–8.

[26] J. J. Rodriguez and J. Aggarwal, "Quantization error in stereo imaging," in *Proceedings CVPR'88: The Computer Society Conference on Computer Vision and Pattern Recognition*, IEEE Computer Society, 1988, pp. 153–154.

[27] C. Strecha, R. Fransens, and L. Van Gool, "Wide-baseline stereo from multiple views: A probabilistic account," in *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, IEEE, vol. 1, 2004, pp. I–I.

[28] J. Wang, C. Zhang, W. Zhu, Z. Zhang, Z. Xiong, and P. A. Chou, "3d scene reconstruction by multiple structured-light based commodity depth cameras," in *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2012, pp. 5429–5432.

[29] S. K. Gehrig and U. Franke, "Improving stereo sub-pixel accuracy for long range stereo," in *2007 IEEE 11th International Conference on Computer Vision*, IEEE, 2007, pp. 1–7.

[30] C. Chang, S. Chatterjee, and P. R. Kube, "On an analysis of static occlusion in stereo vision," in *Proceedings. 1991 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, IEEE, 1991, pp. 722–723.

[31] S. Huq, A. Koschan, and M. Abidi, "Occlusion filling in stereo: Theory and experiments," *Computer Vision and Image Understanding*, vol. 117, no. 6, pp. 688–704, 2013.

[32] B. Hepp, M. Nießner, and O. Hilliges, "Plan3d: Viewpoint and trajectory optimization for aerial multi-view stereo reconstruction," *ACM Transactions on Graphics (TOG)*, vol. 38, no. 1, pp. 1–17, 2018.

[33] B. Sumetheeprasit, R. O. Ladig, and K. Shimonomura, "Variable configuration stereo for aerial survey using two unmanned aerial vehicles," in *2020 IEEE International Conference on Real-time Computing and Robotics (RCAR)*, IEEE, 2020, pp. 268–273.

[34] M. Okutomi and T. Kanade, "A multiple-baseline stereo," in *CVPR*, vol. 93, 1991, pp. 63–69.

[35] J. Kallwies and H.-J. Wuensche, "Effective combination of vertical and horizontal stereo vision," in *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, IEEE, 2018, pp. 1992–2000.

[36] G. D. Hager, W.-C. Chang, and A. S. Morse, "Robot hand-eye coordination based on stereo vision," *IEEE Control Systems Magazine*, vol. 15, no. 1, pp. 30–39, 1995.

[37] K. Konolige, M. Agrawal, R. C. Bolles, C. Cowan, M. Fischler, and B. Gerkey, "Outdoor mapping and navigation using stereo vision," in *Experimental Robotics: The 10th International Symposium on Experimental Robotics*, Springer, 2008, pp. 179–190.

[38] T. Schops, J. L. Schonberger, S. Galliani, *et al.*, "A multi-view stereo benchmark with high-resolution images and multi-camera videos," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 3260–3269.

[39] H. Hirschmuller, "Stereo processing by semiglobal matching and mutual information," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 30, no. 2, pp. 328–341, 2007.

[40] D. Stoyanov, M. V. Scarzanella, P. Pratt, and G.-Z. Yang, "Real-time stereo reconstruction in robotically assisted minimally invasive surgery," in *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2010: 13th International Conference, Beijing, China, September 20-24, 2010, Proceedings, Part I 13*, Springer, 2010, pp. 275–282.

[41] S. Shah, D. Dey, C. Lovett, and A. Kapoor, "Airsim: High-fidelity visual and physical simulation for autonomous vehicles," in *Field and Service Robotics: Results of the 11th International Conference*, Springer, 2018, pp. 621–635.

[42] B. Sumetheeprasit, R. Rosales Martinez, H. Paul, R. Ladig, and K. Shimonomura, "Variable baseline and flexible configuration stereo vision using two aerial robots," *Sensors*, vol. 23, no. 3, p. 1134, 2023.

[43] T. Hinzmann, T. Taubner, and R. Siegwart, "Flexible stereo: Constrained, non-rigid, wide-baseline stereo vision for fixed-wing aerial platforms," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2018, pp. 2550–2557.

[44] S. Garrido-Jurado, R. Muñoz-Salinas, F. J. Madrid-Cuevas, and M. J. Marín-Jiménez, "Automatic generation and detection of highly reliable fiducial markers under occlusion," *Pattern Recognition*, vol. 47, no. 6, pp. 2280–2292, 2014.

[45] J. Y. Chen and B. R. Clark, "Uav-guided navigation for ground robot operations," in *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, SAGE Publications Sage CA: Los Angeles, CA, vol. 52, 2008, pp. 1412–1416.

[46] P. Kim, J. Park, Y. K. Cho, and J. Kang, "Uav-assisted autonomous mobile robot navigation for as-is 3d data collection and registration in cluttered environments," *Automation in Construction*, vol. 106, p. 102 918, 2019.

[47] J. Wang and E. Olson, "AprilTag 2: Efficient and robust fiducial detection," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2016.

[48] R. Ladig and K. Shimonomura, "High precision, intuitive teleoperation of multiple micro aerial vehicles using virtual reality," in *2019 IEEE International Conference on Systems, Man and Cybernetics (SMC)*, IEEE, 2019, pp. 2633–2638.

[49] *Google maps. ritsumeikan university biwako-kusatsu campus.* https://goo.gl/maps/BNZfmefa5A41mT2c7. Accessed 28 May 2022.

[50] D. L. Mills, "Internet time synchronization: The network time protocol," *IEEE Transactions on communications*, vol. 39, no. 10, pp. 1482–1493, 1991.

[51] A. Tiderko, F. Hoeller, and T. Röhling, "The ros multimaster extension for simplified deployment of multi-robot systems," *Robot Operating System (ROS) The Complete Reference (Volume 1)*, pp. 629–650, 2016.

[52] *Google maps. ritsumeikan university biwako-kusatsu campus.* https://goo.gl/maps/BNZfmefa5A41mT2c7. Acessed 6 November 2023., Acessed 6 November 2023.

[53] J. Verbeke and S. Debruyne, "Vibration analysis of a uav multirotor frame," in *Proceedings of isma 2016 international conference on noise and vibration engineering*, 2016, pp. 2401–2409.