ORB-SLAM PERFORMANCE FOR INDOOR ENVIRONMENT USING JACKAL MOBILE ROBOT

by

Tianshu Ruan

A Thesis

Submitted to the Faculty of Purdue University

In Partial Fulfillment of the Requirements for the degree of

Master of Science in Electrical Engineering

Department of Electrical and Computer Engineering

Hammond, Indiana

May 2020

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF COMMITTEE APPROVAL

Dr. Nasser Houshangi, Chair

Department of Electrical and Computer Engineering

Dr. Donald Gray

Department of Electrical and Computer Engineering

Dr. Dave Kozel

Department of Electrical and Computer Engineering

Approved by:

Dr. Xiaoli Yang

Dedicated to my parents Derong Ruan and Huafen He

ACKNOWLEDGMENTS

Here, I would like to express my gratitude to all those who helped me during the writing of this thesis. I gratefully acknowledge the help of my academic advisor, Dr. Nasser Houshangi, who has offered me valuable suggestions in the academic studies. Under his guidance, I have learned much knowledge and many methods for the research. With his patient instruction, insightful criticism and expert guidance, the I complete this thesis successfully.

I also owe a special debt of gratitude to my other committee members, Dr. Donald Gray and Dr. Dave Kozel for their feedbacks. I should finally like to express my gratitude to my group member Amrusha Aryas and Iman Yazdansepas, the ECE workshop faculty Sean M Albano and Clearpath client Nathan Schoenhals in my research for their help and brainstorming.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

SLAM	Simultaneously Localization and Mapping
IMU	Inertial Measurement Units
VO	Visual Odometry
SFM	Structure from Motion
LSD-SLAM	Large Scale Direct - Simultaneously Localization and Mapping
CoSLAM	Collaborative Visual Simultaneously Localization and Mapping
FAST	Features from Accelerated Segment Test
BRIEF	Binary Robust Independent Elementary Features
ORB-SLAM	Oriented Features from Accelerated Segment Test and Rotated Binary Robust Independent Elementary Features - Simultaneously Localization and Mapping
ICP	Iterative Closest Point
PnP	Perspective-n-Point Camera Pose Estimation
RANSAC	Random Sample Consensus
SVD	Singular Value Decomposition
BoW	Bag of Words
BA	Bundle Adjustment
ROS	Robot Operating System

ABSTRACT

Computer vision is a hot topic these days. It has many applications such as object recognitions and navigation. Robot learning is the key ingredient for the future of autonomous robots. Recent trends in robot learning are to use Simultaneous Localization and Mapping (SLAM) technique. The traditional SLAM utilizes IMU, radar and other sensors to find the location and map the environment. In visual SLAM, the camera is the only sensor used and information extracted from the images.

This thesis explains how Oriented FAST and rotated BRIEF SLAM (ORB-SLAM), one of the best visual SLAM solutions, works indoor and evaluates the technique performance for three different cameras: monocular camera, stereo camera and RGB-D camera. Three experiments are designed to find the limitation of the algorithm. From the experiments, the RGB-D SLAM gives the most accurate result for the indoor environment. The monocular SLAM performs better than stereo SLAM on our platform due to limited computation power. It is expected that stereo SLAM provides better results by increasing the experimental platform computational power. The ORB-SLAM results demonstrate the applicability of the approach for the autonomous navigation and future autonomous cars.

1. INTRODUCTION

Mobile robots are increasingly used in daily life. Its core problem is to solve Simultaneous Localization and Mapping (SLAM) [1]. It is a technique for estimating sensor motion and reconstructing structures in an unknown environment. This technique was originally proposed for autonomous robot control. The mobile robot can only navigate autonomously when it knows its position relative to the environment. Therefore, SLAM plays a significant role mobile robot application.

1.1 Introduction of Visual SLAM

In the past few decades, mobile robots and autonomous systems have attracted great attention from researchers around the world, and significant progress and breakthroughs have been made. At present, mobile robots can perform complex tasks autonomously such as delivering a package and autonomous car. In these applications, mobile robots require navigation in complex and dynamic indoor and outdoor environments without human intervention. In order to achieve autonomous navigation and path planning efficiently and safely, robots need to be able to localize themselves in their environment. Therefore, researchers examined the positioning problem in detail and provided various techniques to solve the localization problem.

The SLAM research has been existed for nearly three decades since its first introduction in 1988. It could mainly be divided into two parts according to the whole process: environment detection and optimization. Early SLAM research focused on the use of filter theory to minimize the motion pose and the noise of the landmark points of the map. These works are mostly based on optimization.

In the past, the simplest method of localization is to use odometry based on the encoder of the wheel to measure the amount of wheel rotation of the robot. The measurement of the amount of wheel rotation is combined with the motion model of the robot to find the current position of the robot relative to the global reference frame. The wheel odometry has some major limitations. First of all, vehicles have to move on the ground, and then since the localization data which based on previously estimated positions is incremental, the measurement error will accumulate over time,

causing the estimated robot pose deviating from its actual position. There are many factors leading to errors such as uneven ground and slippery wheels on slippery ground.

To overcome these limitations, other localization methods have been proposed, such as the use of ultrasonic, GPS, Inertial Measurement Units (IMUs) and Lidar. With the deepening of research, Lidar has gradually replaced other sensors and become the most stable mainstream solution. The implementation and difficulty of SLAM vary greatly depending on the type of sensor and the way it is installed. At present, according to the sensor, SLAM is now mainly divided into two categories: laser and vision. Laser SLAM research is earlier, and both theory and engineering are mature. Visual SLAM (VSLAM) is based on visual information which means camera is the only sensor to obtain data. The current visual SLAM is still in laboratory research and can rarely be seen as actual product applications. But in the past ten years, many innovative VSLAM solutions were proposed.

A.J.Davison brought the first monocular SLAM solutions in 2007 "MonoSLAM: real-time single camera SLAM." [2]. It is the first real-time monocular vision SLAM system. Bur the system cannot run online. It relies on the robot to carry the camera to collect data, and then perform localization and mapping offline. At the same time, G. Klein and D. Murray proposed "Parallel Tracking and Mapping" in 2007 [3]. It provides a framework for the current visual SLAM which is discussed in the following subsection. In 2011, "Dense Tracking and Mapping" (DTAM) was introduced by Newcombe, Richard A., Steven J. Lovegrove, and Andrew J. Davison [4]. It is the first visual SLAM solution using direct method which means only the features points are extracted from the image, descriptor of the feature points is not calculated. If the algorithm is based on feature points, the map of visual SLAM is sparse, as feature points only extract from part of the image information. In 2014, Forster, Christian, Matia Pizzoli, and Davide Scaramuzza proposed "SVO: Fast Semi-Direct Monocular Visual Odometry" [5]. It is a VO (visual odometry) algorithm combining feature points and direct method together which leads to less amount of calculation and performs fast. In the same year, "Large-Scale Direct Monocular SLAM" was introduced by J. Engel and T. Schops and D. Cremers [6]. It is based on direct method. Therefore, it can generate a dense map. Mur-Artal, Raul, Jose Maria Martinez Montiel, and Juan D. Tardos proposed "Oriented FAST and rotated BRIEF (ORB) -SLAM" in 2015 [7] and modified it in 2017 with ORB-SLAM2. It is a complete visual SLAM algorithm based on feature points. Unlike other visual SLAM solutions, it can be used on monocular, stereo and RGB-D cameras. At the same time, it

has a flexible usage scenario such as indoor and outdoor environment. So, it is chosen in this project. Both principles and performance are introduced in detail in Chapter 3 and Chapter 4.

1.2 Iconic Solutions of Visual SLAM

After the 21st century, scholars began to learn from SFM (Structure from Motion) to solve visual SLAM problem. This approach has achieved some success and has gained a dominant position in the field of visual SLAM.

1.2.1 Parallel Tracking And Mapping

G. Klein and D. Murray proposed Parallel Tracking and Mapping (PTAM) in 2007. It introduces a keyframe mechanism and extracts features from the image which is shown in Fig 1.1. So, there is no need to deal with each image in detail. Instead, several key images are stringed together and their trajectories and maps are optimized. PTAM is a milestone project in the field of visual SLAM. It was the first to propose and implement parallelization of the tracking and mapping process. It is known that the tracking part needs to respond the image data in real time, and the optimization of the map does not need to be calculated in real time. Optimization can be done slowly in the background (back-end), and the threads are synchronized when necessary. This is the first time that the concept of front-end (tracking and mapping) and back-end (optimization) are distinguished in visual SLAM, leading to the design of many visual SLAM systems. What's more, PTAM is the first to use nonlinear optimization instead of using traditional filters as a back-end solution. Most of the early SLAMs used EKF [8] filters or variants thereof, as well as particle filters [9]. Since people did not realize the sparsity of back-end optimization, they believed that the optimization back-end can't handle such large-scale data in real time, but PTAM is a significant counterexample. After PTAM, visual SLAM research has gradually turned to back-end which is dominated by nonlinear optimization.

PTAM made great progress on monocular SLAM. But it remained the problems of scale ambiguity, which is a common problem of monocular SLAM. It is caused by triangulation calculation when obtaining depth. The solution can be solved by adding other sensors for optimization, such as IMU. Also, when the camera moves rapidly, the image is blurred and fails to continue tracking [3].

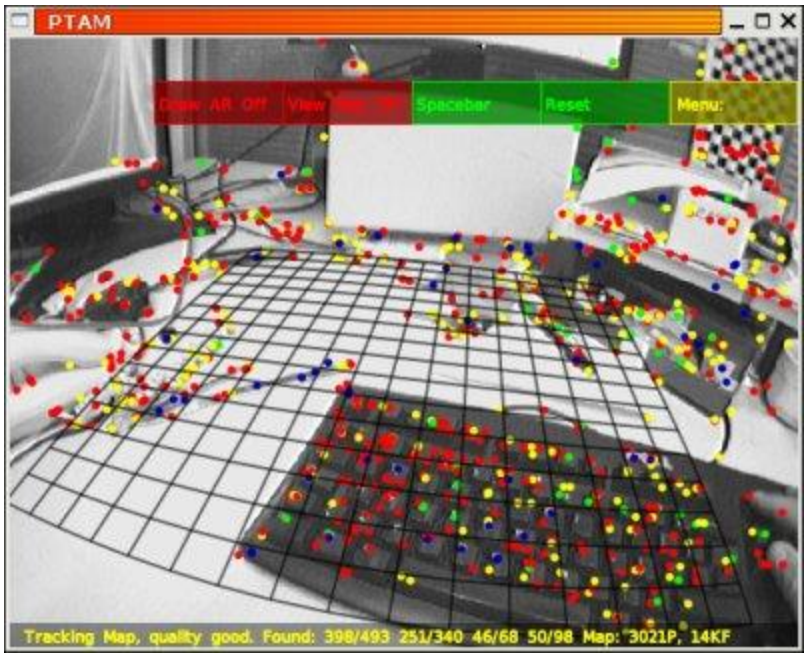


Figure 1.1. PTAM [3] uses "Features from Accelerated Segment Test" (FAST) algorithm to extract feature points from the images shown as dots and builds a plane with its coordinate system based on these feature points.

1.2.2 LSD-SLAM

In 2014, J. Engel and T. Schops and D. Cremers published a paper “LSD-SLAM: Large-Scale Direct Monocular SLAM” [6] which introduced an algorithm for the large-scale environment. It is monocular SLAM algorithm based on direct tracking method. The method only uses the gray level information from the image pixels. It successfully overcomes the limitations of the feature point extraction method which used by PTAM, and can use all the information on the image. This method can achieve high positioning accuracy and robustness in the environment with rare feature points, and provides more environmental geometric information which is shown in Fig 1.2. LSD-SLAM can be divided into three major modules: tracking, depth map estimation, and map optimization. It has the following innovative highlights: LSD-SLAM does not need to extract the feature points from the image, and obtains the transformation between the two frames (R, t) by optimizing the photometric error. Compared with PTAM's method of using feature points, it saves a lot of computational power and time; LSD-SLAM uses high gradient points to calculate the matching, so the final outcome can be a semi-dense map, while PTAM can only generate a sparse point cloud map.

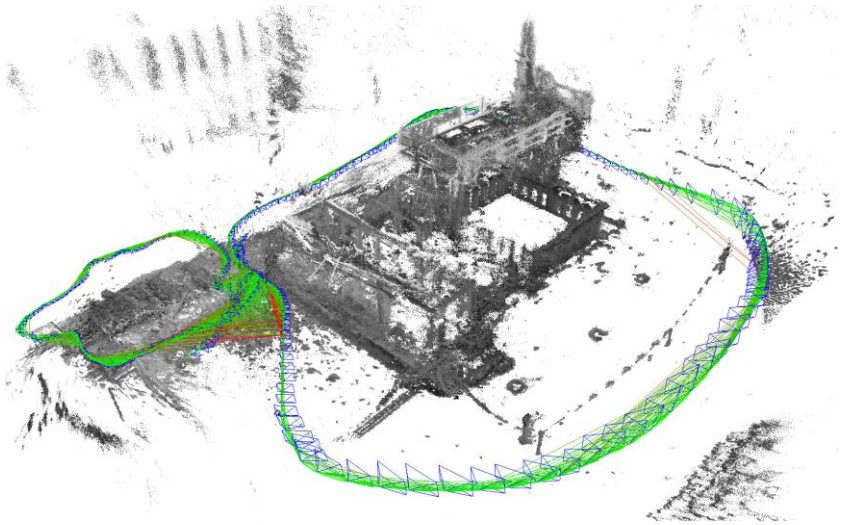


Figure 1.2. Outdoor 3D map built by LSD-SLAM: the green line records the camera trajectory and blue frames refers to the camera poses [6].

1.2.3 CoSLAM

All the solutions mentioned before just utilize one camera as the sensors. Danping Zou and Ping Tan introduced a new way to implement multiple cameras together to conduct visual SLAM. They published a paper "CoSLAM: Collaborative Visual SLAM in Dynamic Environments" in 2013 [11]. It involves two important issues in the field of visual SLAM: dynamic environment and multi-camera collaboration. Each camera can move independently, and the platform is not limited. These cameras will collaboratively build the same global map which frame work is shown in Fig 1.3. The map includes the 3D position of the static point of the environment and the motion track of the dynamic point.

The article introduces how to establish the pose estimation function first for a single camera, mainly to constrain the re-projection error; then, the author proposes a multi-camera collaborative optimization scheme for the dynamic scene. The difference with the former SLAM solution is that the optimization function added the constraint of dynamic points. Each camera independently creates a 3D map of static points. For camera-to-camera collaboration, the author uses Zero-mean Normalized Cross Correlation (ZNCC) to calculate the matching relationship between images, and then uses triangulation to create a map. To enhance the robustness of the system, the uncertainty of the location of each map point is considered. As for the distinction between static points and dynamic points, the author first treats all points as static points and then discriminated according

to the re-projection error: if the re-projection error of a point for a single camera frame exceeds the threshold, the point is marked as "unknown". Then, combined with the information of other cameras, it is possible to determine whether the point is a dynamic point.

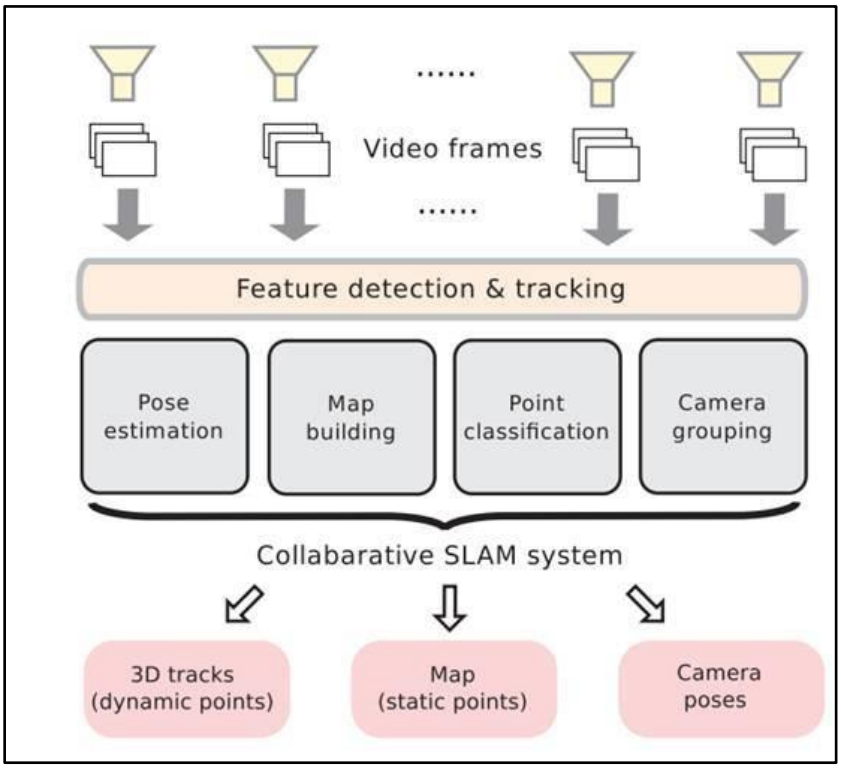


Figure 1.3. Each camera can run separately and complete SLAM together [11].

Among these SLAM solutions, ORB-SLAM has the best flexibility. It supports multiple cameras and can be used both indoors and outdoors. It combines many advantages in the previous VSLAM solutions and makes some improvements based on their disadvantages. Therefore, it is a relatively complete visual SLAM solution and chosen in this project.

1.3 Thesis Overview

Chapter 2 introduces the basic principle of camera imaging and discusses camera calibration. If some features need to be extracted from the image, camera should be calibrated to achieve higher accuracy and low distortion of the real world in the captured images.

Chapter 3 explains the ORB-SLAM procedure. According to the function differences, the algorithm is constituted of three modules: tracking, local mapping and loop closing. Tracking and local mapping are introduced in detail while loop closing is briefly mentioned at the end of the chapter.

Chapter 4 includes the hardware and software settings and three experiments to reveal the performance of ORB-SLAM on different cameras for different indoor environments. Although some limitations are found during the experiments, the ORB-SLAM still shows the potential in future autonomous navigation.

Chapter 5 provides conclusions and discussed future work. The future research to improve ORB-SLAM is also mentioned in this chapter.

2. CAMERA IMAGING AND CALIBRATION

Before introducing the ORB-SLAM algorithm, the camera imaging principle and calibration need to be explained in advance. This chapter explains how to locate feature points from the image accurately in the world coordinate system and how to obtain the camera intrinsic and extrinsic.

2.1 Principle of Camera Imaging

World coordinate, also known as measuring coordinate system, is a three-dimensional coordinate system, which can be used as a benchmark to describe the spatial position of the camera and the object to be measured. The position of the world coordinate system can be determined freely according to the actual situation.

Camera coordinate is also a three-dimensional coordinate system. The origin is located at the center of the lens. The X_c and Y_c axes are parallel to the two sides of the phase plane, and the Z_c axis is the lens optical axis, which is perpendicular to the image plane. Objects in the world coordinate system need a rigid body transformation to the camera coordinate system and then it could relate to the image coordinate system. The unit in the defined coordinate system usually is millimeter.

The camera coordinate system can be obtained from the world coordinate system by rigid body transformation. A rigid body transformation includes rotation and translation shown in Fig 2.1.

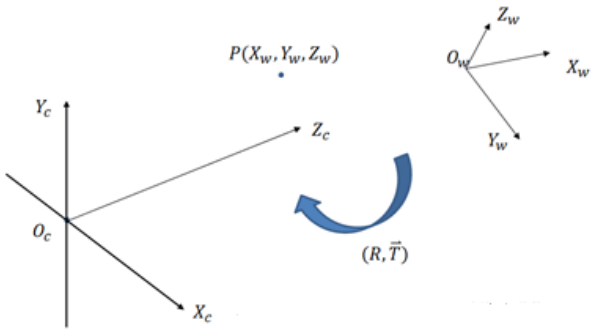


Figure 2.1. Transformation from World Coordinate to Camera Coordinate.

The transform matrixes are usually written as below:

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} = \begin{bmatrix} r_{00} & r_{01} & r_{02} \\ r_{10} & r_{11} & r_{12} \\ r_{20} & r_{21} & r_{22} \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \end{bmatrix} + \begin{bmatrix} T_X \\ T_Y \\ T_Z \end{bmatrix} \tag{2.1}$$

The R matrix is 3x3 the rotation matrix and *T* matrix is the translation vector. The corresponding homogeneous expression is:

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix} = \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} \tag{2.2}$$

R and *t* are independent of the camera, so these two parameters are called the extrinsic parameter of the camera, which can be understood as the distance between the two coordinate origins.

The pixel coordinate system is a two-dimensional coordinate system, which reflects the arrangement of pixels in the CCD/CMOS chip of the camera. The origin is located in the upper left corner of the image shown in Fig 2.2. The unit of coordinate axis in a pixel coordinate system is pixel. Pixel coordinate system is not conducive to coordinate transformation, so it is necessary to establish image coordinate system. The unit of image coordinate system is usually millimeter.

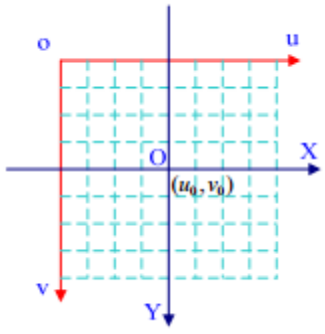


Figure 2.2. Pixel coordinate system.

The intersection point of the camera optical axis and the image plane is defined as the principal point of the image as the origin *O*, and the x-axis is parallel to the u-axis, and the y-axis is parallel to the v-axis, assuming that (u_0, v_0) represents the coordinates of *O* in the u-v coordinate system, dx and dy represent the physical dimensions of each pixel on the horizontal axis *x* and the vertical axis *y*, respectively, and the coordinates of each pixel in the image in the *u*-*v* coordinate system

and in the *X*-*Y* coordinate system. The following relationships exist between the pixel coordinates and image coordinates:

$$u = \frac{x}{dx} + u_0$$
$$v = \frac{y}{dy} + v_0 \tag{2.3}$$

If the unit in the physical coordinate system is millimeter, then the unit of dx is: mm/pixel and the unit of x/dx is the pixel. Hence, the unit is the same as that of *u*. For ease of use, the equations (2.3) can be expressed in homogeneous coordinates and matrix form as:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} 1/dx & 0 & u_0 \\ 0 & 1/dy & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{2.4}$$

Perspective projection connects the camera coordinate system to the image coordinate system as shown in Fig 2.3.

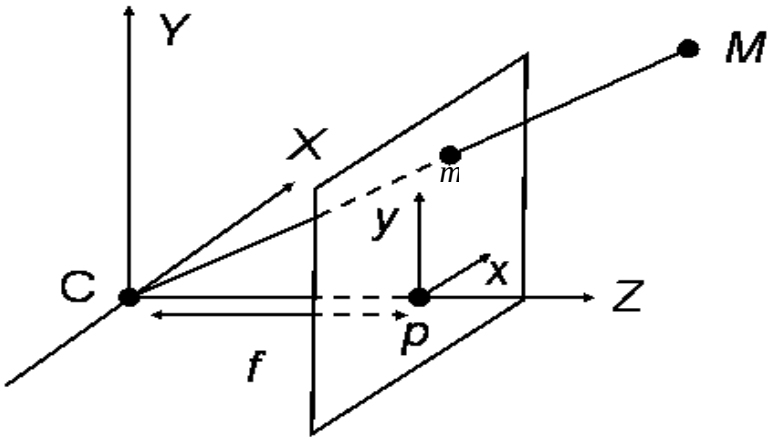


Figure 2.3. Perspective projection among camera coordinate and image coordinate.

The plane shown in Fig 2.3 is called the image plane of the camera, the point *C* is called the camera center (or optical center), *f* is the focal length of the camera. The intersection of the principal axis and the image plane *p* is the principal point of the camera. The image coordinate system is *p*-*xy* and the camera coordinate system is *C*-*XYZ*. The homogeneous coordinate *M* in the camera coordinate system is:

$$M = \left(X_c, Y_c, Z_c, 1\right)^T \tag{2.5}$$

The homogeneous coordinate of image point *m* in the image coordinate system is:

$$m = (x, y, 1)^T \tag{2.6}$$

According to the triangle similarity principle, the following is easy to obtain:

$$\begin{cases} x = \dfrac{f X_c}{z_c} \\ y = \dfrac{f Y_c}{z_c} \end{cases} \tag{2.7}$$

Hence, the transformation could be expressed as:

$$z_c \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix} \tag{2.8}$$

If the three transformations (world coordinate to camera coordinate; camera coordinate to image coordinate; image coordinate to pixel coordinate) combine together, the following equations can be obtained:

$$\begin{aligned} z_c \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} &= z_c \begin{bmatrix} \dfrac{1}{d_x} & 0 & u_0 \\ 0 & \dfrac{1}{d_y} & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} \dfrac{1}{d_x} & 0 & u_0 \\ 0 & \dfrac{1}{d_y} & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix} \\ &= \begin{bmatrix} \dfrac{1}{d_x} & 0 & u_0 \\ 0 & \dfrac{1}{d_y} & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} r_{00} & r_{01} & r_{02} & T_X \\ r_{10} & r_{11} & r_{12} & T_Y \\ r_{20} & r_{21} & r_{22} & T_Z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} \end{aligned} \tag{2.9}$$

Therefore, the world coordinate to image coordinate transformation matrix *P* is as follows:

$$P = \begin{bmatrix} \frac{1}{dx} & 0 & u_0 \\ 0 & \frac{1}{d_y} & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} r_{00} & r_{01} & r_{02} & T_X \\ r_{10} & r_{11} & r_{12} & T_Y \\ r_{20} & r_{21} & r_{22} & T_Z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.10}$$

P can be regard as a projection camera:

$$P = \begin{bmatrix} P_{00} & P_{01} & P_{02} & P_{03} \\ P_{10} & P_{11} & P_{12} & P_{13} \\ P_{20} & P_{21} & P_{22} & P_{23} \end{bmatrix} = \begin{bmatrix} fx & 0 & cx & 0 \\ 0 & fy & cy & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{2.11}$$

where *fx*, *fy*, *cx* and *cy* are camera intrinsic. Three transformations consist the imaging model of the camera. The complete coordinate transformations are shown in Fig 2.4.

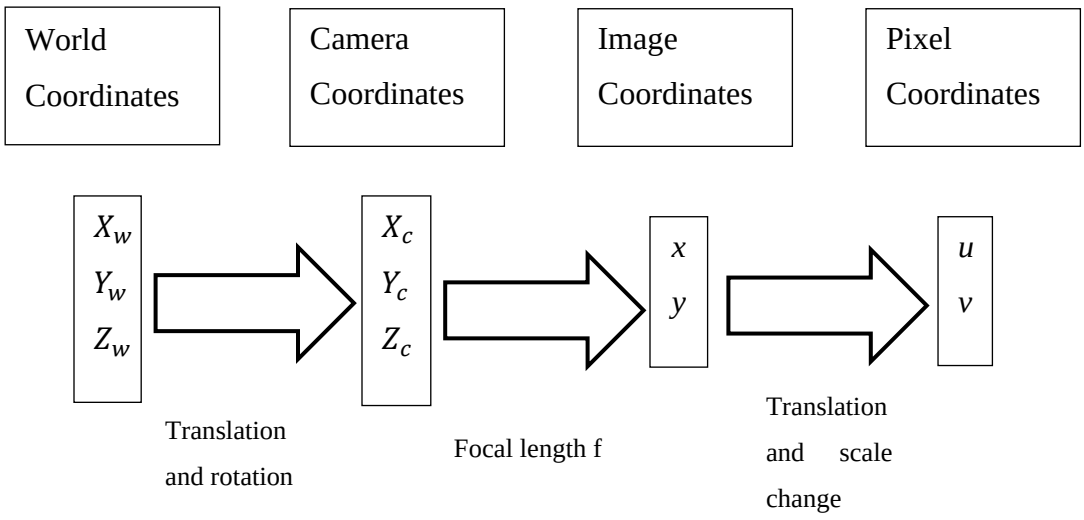
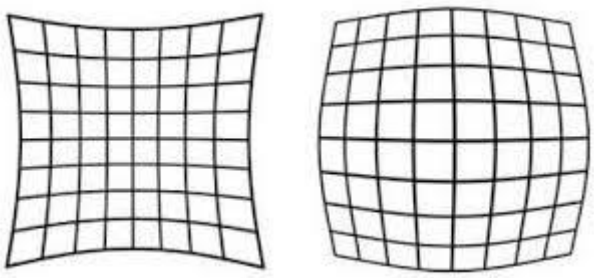


Figure 2.4. Transformations from world coordinates to pixel coordinates.

Due to the camera's hardware limitation, the image will be distorted. The distortion of the camera is actually the general term for the perspective distortion in optical lenses. The most direct effect is that the coordinates of the object on the imaging plane and the theoretical values calculated from above are not equal. Therefore, it is necessary to eliminate distortion.

Distortion can generally be categorized into two: radial distortion and tangential distortion. There are some more types of distortions, but these two mentioned above are the most significant.

Radial distortion is the general name of Barrel distortion and Pincushion distortion shown in Fig 2.5. The actual camera lens always produces significant distortion at the edge of the image. This phenomenon is more serious for common conventional lenses. Barrel distortion is widely found in cheap webcams, but not in high-level cameras because these lens systems have been done a lot of work to eliminate radial distortion.



(a) (b)

Figure 2.5. (a): Barrel distortion; (b): Pincushion distortion.

For radial distortion, the distortion of the optical center is zero. And the distortion becomes more and more serious when the point closer to the edge of the image.

2.2 Camera Calibration

In the process of image measurement and the application of machine vision, the geometric model of camera imaging must be established in order to determine the relationship between the three-dimensional geometric position of a point from the environment and its corresponding points in the image. These geometric model parameters are camera parameters. In most cases, these parameters must be obtained through experiments and calculations. The process of solving these parameters is called camera calibration. Whether in image measurement or computer vision applications, the calibration of camera parameters is a must. The accuracy of the calibration results and the stability of the algorithm directly affect the accuracy of the imaging. Therefore, camera calibration is one of the keys of this project.

To solve this problem, Professor Zhang, Zhengyou proposed a camera calibration method for single-plane checkerboard in 1998 [12]. The method in this paper is between the traditional

calibration method and the self-calibration method, but overcomes the shortcomings of the traditional calibration. It only needs to use a printed checkerboard. Meanwhile, compared with the self-calibration, the accuracy is improved and the operation is quite convenient. Therefore, Zhang's calibration method is widely used in computer vision. Zhang's calibration only considers radial distortion which is the most influential and does not consider tangential distortion. It is used to calibrate the cameras which are used in the experiments. The detailed algorithm is explained in appendix A and the approach to get the calibration data will be provided in Chapter 4.

3. RESEARCH METHODOLOGY

After introducing the principle of camera imaging and camera calibration, ORB-SLAM will be explained in detail in this chapter. This chapter introduces each module according to the ORB-SLAM operating sequence: Tracking, Local Mapping and Loop Closing.

In 2015 Mur-Artal, Raul, Jose Maria Martinez Montiel, and Juan D. Tardos brought Oriented FAST and rotated BRIEF (ORB) -SLAM into public sight [7]. A total of two versions have been released. First version was designed for monocular camera. Stereo and RGB-D cameras such as Kinect are adapted by the solution in 2017 [13]. The result of ORB-SLAM using RGB-D camera in small-scale or indoor environment is shown in Fig 3.1. ORB-SLAM is a complete SLAM system that includes visual odometry (VO), tracking, and loop closing detection. It is a multiple cameras SLAM system based entirely on sparse feature points. The highlight of the approach is the use of ORB as a core feature detector and descriptor in the VO. It has some innovative features. First, feature points extracted and tracked use the ORB algorithm. The extraction process of ORB features is very fast and is suitable for real-time systems. Second, loop closing detection uses the bag of words. Third, it has many interfaces, supporting for monocular, stereo and RGB-D cameras. ROS is also supported by the system. Finally, real-time calculations can be performed on a computer with high computational power.

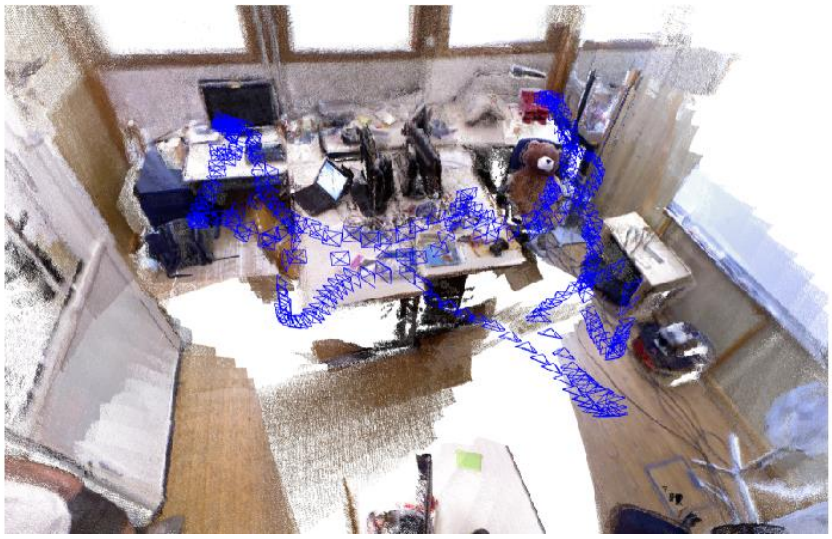


Figure 3.1. Indoor map rebuilt by RGB-D camera. The blue frames are the keyframes which record the trajectory of the camera [7].

3.1 Tracking

3.1.1 FAST Algorithm

Feature point detection is widely used in objects matching, objects tracking, 3D reconstruction and etc. Colors, corner points, contours, textures and other features are generally used. Edward Rosten and Tom Drummond proposed a FAST (Features from Accelerated Segment Test) feature point in "Machine learning for high-speed corner detection" in 2006 [10]. And in 2010, they published "Features from Accelerated Segment Test"[14].

The principle of the corner point is derived from the perceptual sense of the corner point from people, that is, the image has a significant change in the gray level in all directions. Corner points are often detected at the intersection of edges, occluded edges, and highly textured parts. These points are generally stable and repetitive. So, it is not important to determine whether they are corners or not. Because the goal is to find some stable, reproducible points as feature points. Rosten defined the FAST corner point as: If a pixel has a large difference between enough pixels in its surrounding neighborhood, the pixel may be a corner point.

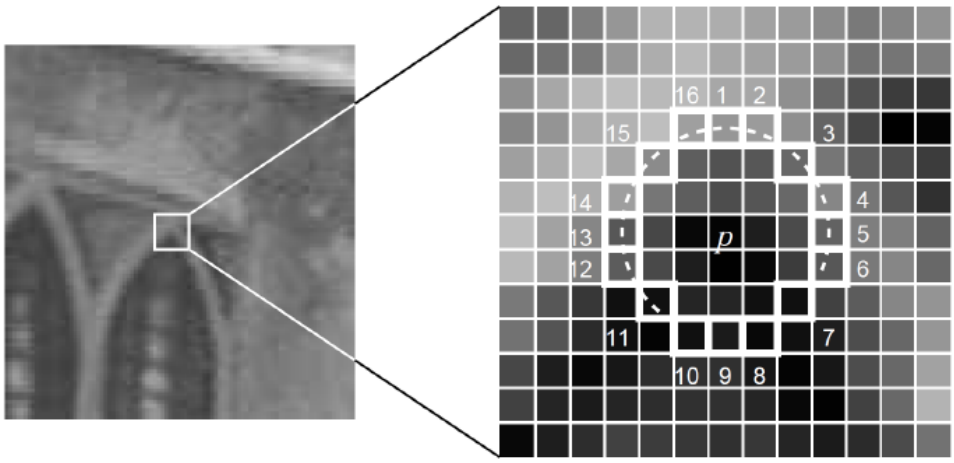


Figure 3.2. 16 neighboring pixels are considered in the FAST algorithm [14].

The steps for the FAST algorithm are described as follows and shown in Fig 3.2:

1. As shown in the Fig 3.1, a circle with a radius of 3 centered on the pixel p has 16 pixels (p1, p2, ..., p16);

2. A threshold needs to be defined such as 30% of the pixel *p*'s gray level. The gray level difference between p1 and center *p* can be calculated. The same calculation can be done between p9 and the center *p*. If their absolute values are less than the threshold, *p* point cannot be a feature point and skip directly; otherwise, as a candidate point, it needs further evaluation;
3. If *p* is a candidate point, the pixel difference between p1, p9, p5, p13 and center *p* need to be calculated. If at least 3 of their absolute values exceed the threshold, the point remains as a candidate point; otherwise, it is skipped;
4. If *p* is a candidate point, the pixel difference between p1 to p16 and center *p* need to be calculated. If at least 9 of their absolute values exceed the threshold, the point remains as a candidate point; otherwise, it is skipped;
5. Calculate the FAST score value of the feature point which is the sum of the absolute values of the 16 points to the center difference, that is, the s value, and filter a neighborhood (such as 3x3 or 5x5) centered on the feature point *p*. If there are multiple feature points, determine the s value of each feature point. The point with the largest s value among them could be saved, and the rest can be skipped. If there is only one feature point (corner point) in the neighborhood, it needs to be saved.

The flow chart of FAST algorithm is shown in Fig 3.3. The FAST algorithm is simple and clear to implement. The definition of corner points is also concise. It reduces the calculation amount of finding feature points. As a result, the running speed of the algorithm is greatly improved compared with the traditional feature point detectors [15][16]. However, this detection method cannot be extended to the case of continuous bright pixels or dark pixels where $n<12$ (n is the number of pixels which their difference exceeds the threshold). And this method has implicit assumption about the spatial distribution of feature point: The feature points are randomly spread in the image. But from large number of tests showed that the feature points are mostly adjacently distributed which means they gather in the part of the images.

FAST algorithm is used to extract features from the images. If the features need to pair with each other in two images, the descriptor needs to be added to the feature point. BRIEF algorithm is designed to generate a specific binary id code for the feature point which is used for pairing as discussed in the next subsection.

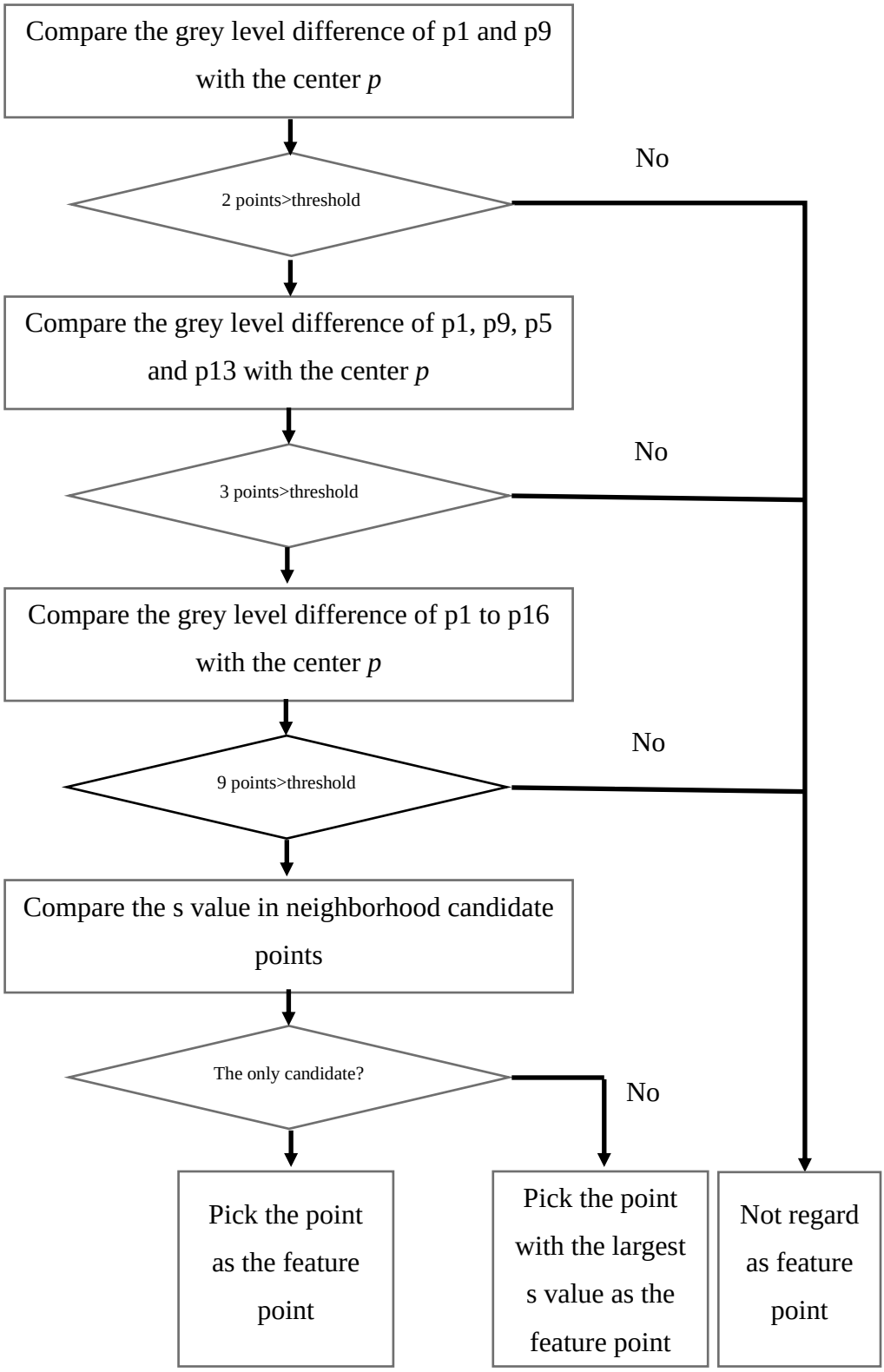


Figure 3.3. The procedure of FAST algorithm.

3.1.2 BRIEF Algorithm

FAST algorithm solves the problem of extracting feature points, while BRIEF algorithm gives a solution for feature point description. BRIEF was proposed in an article in 2010 entitled "BRIEF: Binary Robust Independent Elementary Features" by Michael Calonder, Vincent Lepetit, Christoph Strecha, and Pascal Fua [17]. BRIEF describes the detected feature points in binary code. It eliminates the traditional method of describing feature points by using region gray histograms, which greatly speeds up the generation of feature descriptors. The steps of BRIEF algorithm are as follows:

1. To reduce noise interference, Gaussian filtering [18] is performed on the image Gaussian filter window is 9x9;
2. Based on the feature points, the neighborhood matrix window of SxS (S is the number of pixels) is taken. A pair of pixels are randomly selected in the window. And the grey level of the two pixels is compared, and the following binary assignment is performed;

$$\tau(p; x, y) = \begin{cases} 1 & \text{if } p(x) < p(y) \\ 0 & \text{otherwise} \end{cases} \tag{3.1}$$

   where *p(x)* and *p(y)* is the grey level of the random pixels.
3. Repeat the binary assignment of step 2 to form a binary code. This code is a description of the feature points, that is, feature descriptors. The authors tested 5 sampling method to randomly pick N pair for *p(x)* and *p(y)*, such as random sampling at discrete positions in polar coordinates and sampling x and y with Gauss distribution (0, $S^2/25$). The last sampling method has the best paring performance according to the test from the authors [7].

After the encoding process, a 256-bit binary code is obtained for each feature point in the image. Then, it is possible to match the two images with similar or overlapping parts. Hamming distance [19] is used to determine feature point matching. Two rules are applied in this part:

1. If the Hamming distance of the feature code corresponding to two pixels is greater than 128, the two points must not be paired.
2. If the feature code corresponding to two pixels has the smallest Hamming distance, it can be paired.

In the image where the rotation is not great, the matching quality of the feature points descriptor generated by the BRIEF is very high. At the same time, the time consumption of the BRIEF is

very short. When calculating the descriptors of 512 feature points, the BRIEF only needs 2.19ms [7] which much faster than traditional algorithm.

3.1.3 ORB Algorithm

FAST and BRIEF algorithm are the base to extracting features, encoding the features and pairing features in ORB-SLAM. ORB-SLAM utilizes FAST and BRIEF with modifications discussed later.

Feature point extraction

ORB algorithm utilizes FAST algorithm to extract feature points from the images. And some modifications are made to improve the FAST algorithm. The modified algorithm is called oFAST (Oriented FAST). The feature points extracted by original FAST algorithm will not contain the parameter for the direction. So, the authors propose a method to find the directions which provide great help solving the rotation problem in feature point matching part.

Building a scale pyramid

FAST feature points have no scale invariance. It means that if the scale of the same image change, the feature points extracted by FAST are different from the two images. A scale pyramid shown in Fig 3.4 can solve this problem.

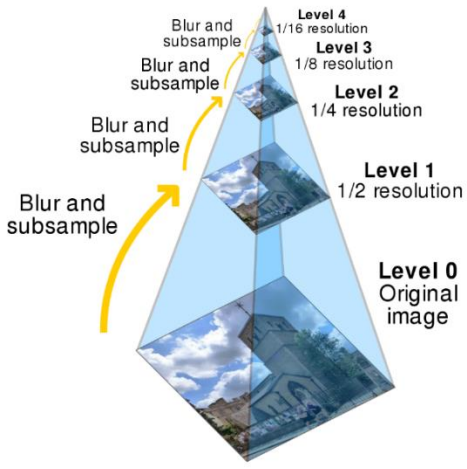


Figure 3.4. Image pyramid with different layers.

The original image lays at the bottom of the pyramid. As the pyramid moves up to the top, both size and resolution will reduce. In the ORB algorithm, the scale factor of the pyramid is 1.2 which means the ratio of the pixel area from closed layers is 1.44. The number of pyramid layers is set to 8 layers. In other words, including the original image, a total of 8 different sizes of images were generated. The grey level of scaled image I' can be expressed as:

$$I' = \frac{I}{(\text{ScaleFactor})^k} \tag{3.2}$$

where *k* represents the number of image level and *I* is the original image grey level. The sum of the feature points extracted from 8 different scales of image are regard as the oFAST feature points.

Determining the feature point orientation

The ORB algorithm proposes the use of moments to determine the direction of the FAST feature points. That is to say, the centroid in the range r of the feature point radius can be calculated by the moment [20]. The feature point coordinates to the centroid forms a vector as the direction of the feature point. The *p+q* order moment is defined as follows:

$$m_{pq} = \sum_{x,y} x^p y^q I(x, y) \tag{3.3}$$

where *I(x, y)* is the image grey level expression and *p,q* are 1,0 and 0,1. Because the first order moment m_{10} and m_{01} can be used to describe the centroid of the image, the centroid of the image *C* can be expressed as:

$$C = \left(\frac{m_{10}}{m_{00}}, \frac{m_{01}}{m_{00}} \right) \tag{3.4}$$

where zero order moment m_{00} is the sum of the gray levels of each pixel in the range r of the feature points. Then the angle θ between the feature point and the centroid is defined as the direction of the FAST feature point:

$$\theta = \operatorname{atan} 2\left(m_{01}, m_{10} \right) \tag{3.5}$$

Feature point description

However, the BRIEF algorithm has a very low pairing success rate in two images with big rotation. The authors found that if a rotation greater than 30°, the matching rate of the BRIEFF quickly dropped to around zero [7]. To solve this problem, steered BRIEF is introduced.

Steered BRIEF

It is assumed that the original BRIEF algorithm selects n pairs of points D in the neighborhood pixels of the feature point SxS (generally S = 31).

$$D = \begin{pmatrix} x_1, x_2, \ldots, x_{2n} \\ y_1, y_2, \ldots, y_{2n} \end{pmatrix} \tag{3.6}$$

Rotate the angle θ to get a new point pair:

$$D_\theta = R_\theta D \tag{3.7}$$

where R_θ is the rotation matrix and θ is the orientation of the feature point calculated by oFAST using the equation (3.5).

At this point, the original BRIEF calculation can be performed on the new point set D_θ. Since the oFast algorithm extracts features at different scales, the images are also need to convert to the corresponding scale when using the BRIEF feature descriptor using equation (3.2).

Modification to steered BRIEF-rBRIEF

The feature descriptor obtained by the steered BRIEF method has rotation invariance, but it is inferior to the original BRIEF algorithm in the distinguishability of the descriptor, or the correlation. Distinguishability has a great impact on the quality of feature matching. A descriptor is a description of the nature of a feature point. The descriptors express the difference between feature points and other feature points. The descriptors calculated should try to express the uniqueness of the feature points. If the distinguishability of descriptors of different feature points is relatively poor, it is not easy to find the corresponding matching points and even leads to mismatch. In [7], different methods are used to calculate binary descriptors for 100k feature points, and analysis these descriptors, as shown in the following Fig 3.5. The x-axis represents the distance from the mean value of 0.5, and the y-axis is the number of feature points on the corresponding mean.

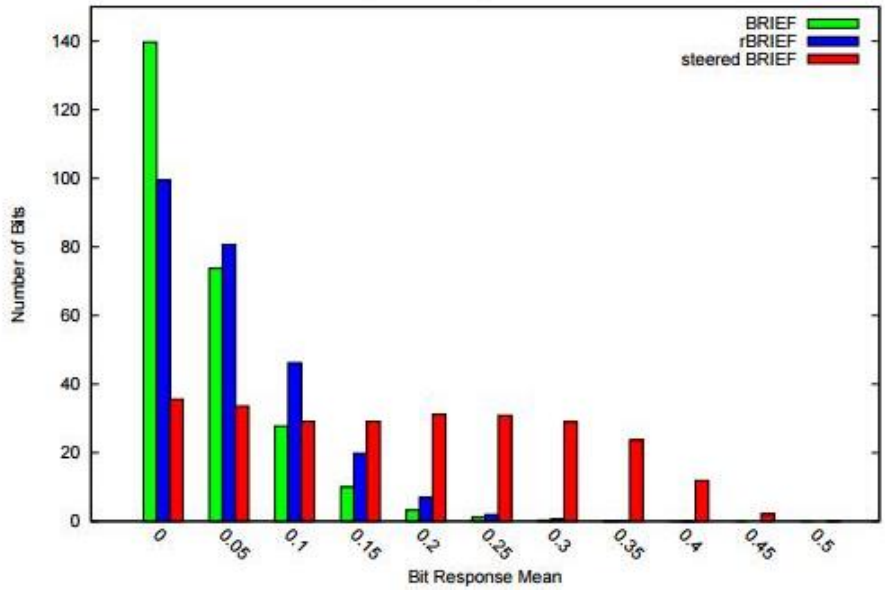


Figure 3.5. The distribution of bits response means [7].

As it is mentioned in BRIEF introduction, the descriptor here is a binary string. The value inside is 0 or 1. If the average value of the binary string is around 0.5, then the string will include the closed amount of 0 and 1, so the variance is larger. If the average of two binary strings is much larger than 0.5, it means that there are more 1 in both binary strings. The probability that 1 will appear simultaneously in the same position of the two strings will be high. Then the descriptors of these two feature points have great similarities. This increases the correlation between descriptors and reduces the discriminability of cases.

Comparing the two algorithms between BRIEF and steered BRIEF, the number of feature points of the BRIEF algorithm falling on 0 is large. Therefore, the mean value of the descriptors calculated by the BRIEF algorithm is about 0.5, and the variance of each descriptor is larger. So, it has better distinguishability. But steered BRIEF lost this feature.

The authors abandon the way mentioned in the original BRIEF algorithm to select point pairs [7]. Instead, they find another way to reselect point pairs collection. This new method is called rBRIEF. For each point in the data set, its 31x31 pixels neighborhood is brought into consideration. Gaussian smoothing is applied to the image which means the average gray level in a 5x5 neighborhood of a point is used to instead the actual grayscale value of the point. This feature improves its ability to resist noise. There are (31-5) x (31-5) =676 such sub-windows in the

neighborhood of 31x31. After eliminating the overlap pairs, the possibility way to pick the pair is M=205590. The authors [7] uses the following steps to select 256 pairs from the data set.:

1. Assume that the total number of feature points is 300k, a matrix Q with 300k rows and M columns is built. Each column represents the binary code according to the rBRIEF result;
2. An average is calculated for each column in Q matrix. According to the distance from the average value to 0.5, resort the column vector from small to large and generate a new matrix T;
3. Select the first column in T as the vector R;
4. Calculate the correlation between the next column in T (1x300k) and all the vectors in R(1x300k) separately. If the absolute correlation is smaller than the threshold, the column vector in T is moved to R;
5. Repeat step 4 until there are 256 vectors in matrix R. If there are less than 256 vectors in matrix R, the threshold in step 4 needs to increase.

The rBRIEF combined the rotation invariance and low correlations together. Therefore, it is chosen to select feature points pair in ORB-SLAM. If the paired feature points are obtained, the camera pose and map points location can be calculated based on these data which is introduced in next subsection.

3.1.4 3D Points Matching and Camera Pose

After the feature extraction and matching, the camera's motion based on the matched point pairs needs to be estimate. The approach taken depends on the camera used.

For a monocular camera, only the 2D pixel coordinates are known, so the problem is to estimate camera motion based on two sets of 2D points. This problem is solved with the epipolar geometry. For stereo, RGB-D cameras, the camera motion can be estimated based on two sets of 3D points. This problem is solved by Iterative Closest Point（ICP）.

Mono camera

In the ORB algorithm, the monocular initialization process uses a special initializer to complete the initialization by two frames of images. This process uses two frames of image and the epipolar geometric constraint method to solve the essential matrix E and the homography matrix H. Then, it is possible to estimate the pose of the camera. At last, using triangulation to calculate the spatial position of the feature points. Therefore, the epipolar constraint and triangulation play an important role in the ORB algorithm.

Epipolar constraint

Suppose a bunch of matching feature points is obtained, as shown in the Fig 3.6.

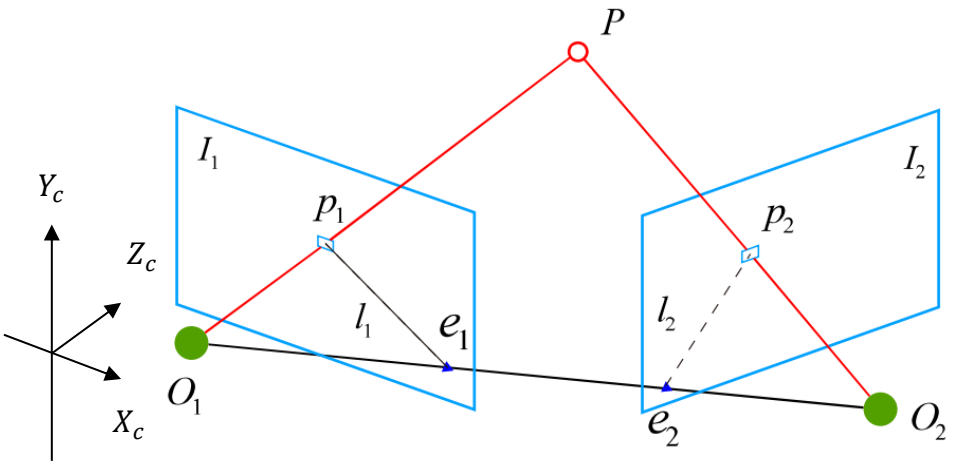


Figure 3.6. The feature points relationship between two frames.

In the camera coordinate system of the first frame (left) image, let the spatial position of P be:

$$P = [X, Y, Z]^T \tag{3.8}$$

According to the camera model, (camera intrinsic matrix) $\times$ (points coordinate under camera coordinate system) = (points coordinate under image coordinate system). Therefore, two pixels (feature points) $p_1$and p_2 coordinate can be obtained:

$$s_1 p_1 = KP, s_2 p_2 = K(RP + t) \tag{3.9}$$

where K is the camera intrinsic matrix obtained by camera calibration, R and t are the camera motions between two frames, and s_1 and s_2 represent the depth of the two pixels, respectively.

Using homogeneous coordinates on the depth which means s_1=s_2=1, the equation could be written as:

$$p_1 = KP, p_2 = K(RP + t) \tag{3.10}$$

In the 2D-2D problem, only pixel coordinates p_1 and p_2 are known, while P is unknown. At this point, the concept of a normalized coordinate can be introduced here. Let

$$\begin{aligned} x_1 &= K^{-1} p_1 \\ x_2 &= K^{-1} p_2 \end{aligned} \tag{3.11}$$

where x_1 and x_2 are the coordinates on the normalized plane of two pixels. Bring it into equation (3.11), get:

$$x_2 = Rx_1 + t \tag{3.12}$$

Both sides are multiplied by t at the same time, which is equivalent to the outer product of both sides and t:

$$t \otimes x_2 = t \otimes Rx_1 \tag{3.13}$$

Multiply x_2^T on both sides simultaneously:

$$x_2^T t \otimes x_2 = x_2^T t \otimes Rx_1 \tag{3.14}$$

In left side of the equation, $t \otimes x_2$is a vector that is perpendicular to both t and x_2. When it multiplies with x_2, the result is 0. Therefore, the equation (3.14) can be written as:

$$x_2^T t \otimes Rx_1 = 0 \tag{3.15}$$

Substituting (3.11) in (3.15):

$$p_2^T K^{-T} t \otimes RK^{-1} p_1 = 0 \tag{3.16}$$

Equation (3.15) and (3.16) are called epipolar constraints, and their geometric meanings are O_1, P and O_2 are coplanar. Both the translation and the rotation are included in the epipolar constraint. Record the middle part of (3.16) as two matrices: Fundamental matrix F and Essential matrix E:

$$\begin{aligned} E &= t \otimes R \\ F &= K^{-T} E K^{-1} \end{aligned} \tag{3.17}$$

Substituting (3.17) in (3.16) and (3.15), results in:

$$x_2^T E x_1 = p_2^T F p_1 = 0 \tag{3.18}$$

Therefore, the camera pose estimation problem becomes the following two steps:

1. Find *E* or *F* according to the pixel position of the paring points;
2. Find *R*, *t* from *E* or *F*.

E and *F* differ from camera intrinsic, and camera intrinsic are usually known in SLAM. Therefore, *E* is in simple expression and often used in practice.

Way to find essential matrix E

The essential matrix $E = t \otimes R$, which is a 3x3 matrix with 9 unknowns. From the perspective of the construction of *E*, it consists of several important features [21] as follows

1. The essential matrix is defined by the epipolar constraint. Since the epipolar constraint (3.15) is a constraint of equality 0, the epipolar constraint is still satisfied after multiplying by any non-zero constant. This is called *E* is equivalent in different scales.
2. According to $E = t \otimes R$, it can be proved that the singular value of the essential matrix *E* must be the expression $[\sigma, \sigma, 0]^T$ where σ is the singular value.
3. There are 6 degrees of freedom in essential matrix E, as the translation and rotation have 3 degrees of freedom separately. However, due to the scale equivalence, *E* actually has only 5 degrees of freedom.

The fact that *E* has five degrees of freedom indicates that *E* can be solved with at least five pairs of points. However, *E* is a nonlinear matrix, which causes trouble when solving linear equations. Therefore, it is also possible to consider its scale equivalence, and use eight pairs of points to estimate *E*—this is the classic Eight-Point-Algorithm [22].

A pair of matching points whose normalized coordinates are: $x_1 = (u_1, v_1, 1)$ and $x_2 = (u_2, v_2, 1)$. According to the epipolar constraint, the following can be written:

$$(u_1, v_1, 1)\begin{pmatrix} e_1 & e_2 & e_3 \\ e_4 & e_5 & e_6 \\ e_7 & e_8 & e_9 \end{pmatrix}\begin{pmatrix} u_2 \\ v_2 \\ 1 \end{pmatrix} = 0 \tag{3.19}$$

Then, rewrite the equation into the following form:

$$[u_1u_2, u_1v_2, u_1, v_1u_2, v_1v_2, v_1, u_2, v_2, 1] \cdot e = 0 \tag{3.20}$$

$$\begin{pmatrix} u_1^1u_2^1 & u_1^1v_2^1 & u_1^1 & v_1^1u_2^1 & v_1^1v_2^1 & v_1^1 & u_2^1 & v_2^1 & 1 \\ u_1^2u_2^2 & u_1^2v_2^2 & u_1^2 & v_1^2u_2^2 & v_1^2v_2^2 & v_1^2 & u_2^2 & v_2^2 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ u_1^8u_2^8 & u_1^8v_2^8 & u_1^8 & v_1^8u_2^8 & v_1^8v_2^8 & v_1^8 & u_2^8 & v_2^8 & 1 \end{pmatrix} \begin{pmatrix} e_1 \\ e_2 \\ e_3 \\ e_3 \\ e_4 \\ e_5 \\ e_7 \\ e_8 \\ e_9 \end{pmatrix} = 0 \tag{3.21}$$

If the matrix consisting of eight pairs of matching points satisfies the condition of rank 8, then E can be solved by equation (3.21). Once the essential matrix E is obtained, the pose of the camera R and the camera extrinsic t can be recovered by Singular Value Decomposition (SVD) [23].

Homography matrix H

The homography H is a special case of fundamental matrix. It describes the transformation between points on a same plane between two images. If the feature points $p_1(x_1, y_1)$ and $p_2(x_2, y_2)$ are matched on two frames, and these feature points fall on the same plane, then H will satisfy the equation:

$$\begin{pmatrix} x_2 \\ y_2 \\ 1 \end{pmatrix} = \begin{pmatrix} H_{11} & H_{12} & H_{13} \\ H_{21} & H_{22} & H_{23} \\ H_{31} & H_{32} & H_{33} \end{pmatrix} \begin{pmatrix} x_1 \\ y_1 \\ 1 \end{pmatrix} \Leftrightarrow p_2 = Hp_1 \tag{3.22}$$

$$\begin{cases} x_2 = H_{11}x_1 + H_{12}y_1 + H_{13} \\ y_2 = H_{21}x_1 + H_{22}y_1 + H_{23} \\ 1 = H_{31}x_1 + H_{32}y_1 + H_{33} \end{cases} \tag{3.23}$$

In the equation set (3.23), the left and right sides of the first and second equations are simultaneously multiplied by the third equations, the following can be obtained:

$$\begin{aligned} x_2\left(H_{31}x_1 + H_{32}y_1 + H_{33}\right) &= H_{11}x_1 + H_{12}y_1 + H_{13} \\ y_2\left(H_{31}x_1 + H_{32}y_1 + H_{33}\right) &= H_{21}x_1 + H_{22}y_1 + H_{23} \end{aligned} \tag{3.24}$$

And they can be written as:

$$\begin{aligned} x_2\left(H_{31}x_1 + H_{32}y_1 + H_{33}\right) - H_{11}x_1 + H_{12}y_1 + H_{13} &= 0 \\ y_2\left(H_{31}x_1 + H_{32}y_1 + H_{33}\right) - H_{21}x_1 + H_{22}y_1 + H_{23} &= 0 \end{aligned} \tag{3.25}$$

The homography matrix is a homogeneous matrix, so the last element can be normalized to 1. If $h=\left(H_{11},H_{12},H_{13},H_{21},H_{22},H_{23},H_{31},H_{32},1\right)^T$, the equations (3.25) can be rewritten as:

$$\begin{aligned} a_x h &= 0 \\ a_y h &= 0 \end{aligned} \tag{3.26}$$

where $a_x=(-x_1,-y_1,-1,0,0,0,x_2x_1,x_2y_1,x_2)^T$, $a_y=(0,0,0,-x_1,-y_1,-1,y_2x_1,y_2y_1,y_2)^T$. H has 8 unknowns, that is to say, at least 4 pairs of matched pairs (any 3 points of 8 cannot collinear), the homography matrix of the two images can be found. But usually, the matching point of used in the to solve the H are more than 4 pairs. Finally, the homography matrix can be estimated using the or Random Sample Consensus (RANSAC) method which is a built-in function of OpenCV and can be used directly [24].

The fundamental matrix F represents the epipolar constraint of the two views. It is independent of the structure of the 3D scene. It only depends on the intrinsic and extrinsic of the camera. It needs the rotation and translation of the camera. While the homography matrix has more requirements for the three-dimensional structure of the scene, and the points in the scene are required to be on the same plane; Or, the movement is pure rotation without any translation. The homography matrix can find the exact position from another image. But the fundamental matrix can only indicate the mapping between points and the epipolar line on another image.

Triangulation

After knowing the relationship of the two images and the corresponding features points pair, the feature point 3D position can be restored.

Triangulation is used to the estimation of the depth value of a map point. After estimating the camera motion using the epipolar geometry constraint, it is necessary to estimate the spatial position of the feature points using the motion of the camera. In monocular SLAM, since the depth of the map point cannot be obtained from a single image, it can be calculated by triangulation. Triangulation means that the angle of the point is determined by observing the angle of the same point in two places.

The 3D position of the spatial point can be solved using Direct Linear Transformation (DLT) [25].

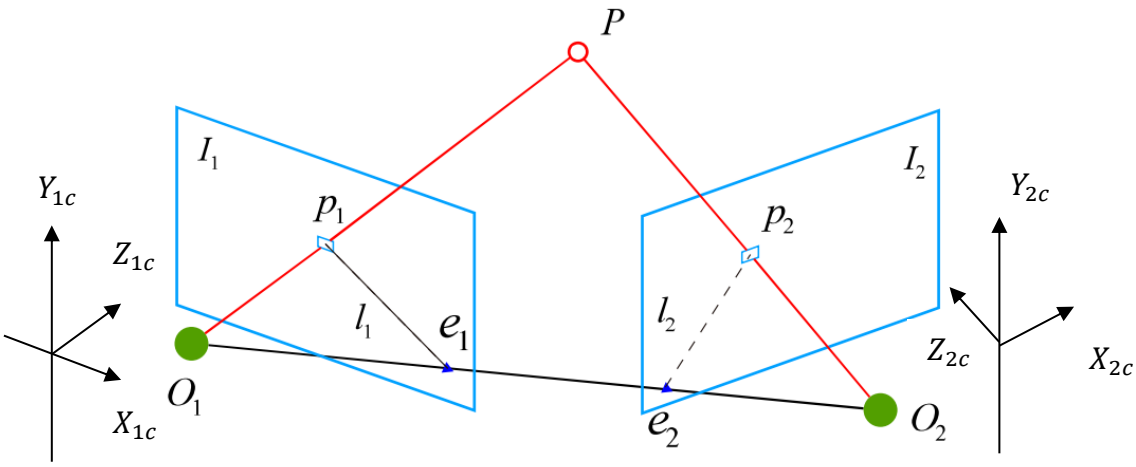


Figure 3.7. Calculate the pose transformation between two frames based on matching feature points.

As shown in the Fig 3.7 above, the feature points in the two images are p_1 and p_2, respectively. In theory, the straight lines O_1p_1 and O_2p_2 intersect in a scene at a point P, which is the position of the map point corresponding to the two feature points in the three-dimensional scene. However, due to the influence of noise, the two lines often cannot intersect. Assume that the coordinates of point P in the camera coordinate system are c_1 and c_2. According to the coordinate transformation equation (3.9), it is possible to get:

$$\begin{aligned} s_1 p_1 &= Kc_1 \\ s_2 p_2 &= Kc_2 \end{aligned} \tag{3.27}$$

where s_1 is the depth of P on the Z_{1c} axis, s_2 is the depth of P on the Z_{2c} axis, and K is the intrinsic matrix of the camera as defined earlier in equation (3.9). Let $p_1 = Kx_1$, $p_2 = Kx_2$, $x_1$and x_2 are normalized coordinates of point P in the two camera coordinate systems. According to the motion model in three-dimensional space, the following could be obtained:

$$c_2 = Rc_1 + t \tag{3.28}$$

$$k^{-1}s_1 p_1 = Rk^{-1}s_2 p_2 + t \tag{3.29}$$

$$s_1 x_1 = s_2 Rx_2 + t \tag{3.30}$$

The R and t are calculated from essential matrix E, what needs to be solved is the depth value s_1 and s_2 of two feature points. First, multiply the skew-symmetric matrix $\hat{x}_1$ of x_1 at the same time:

$$s_1 x_1^{\wedge} x_1 = 0 = s_2 x_1^{\wedge} R x_2 + x_1^{\wedge} t \tag{3.31}$$

On the right side of the above equation is an equation for the unknown variable x_2. Solving this equation leads to s_2. And s_1 could be solved easily. However, due to the the error, the estimated *R* and *t* do not necessarily make the equation (3.31) equal to 0. So, the more common approach is to find the least squares solution.

The premise of triangulation is the translation between the two images. To improve the accuracy of triangulation, one is to improve the extraction precision of feature points, that is, to improve the image resolution; the other is to increase the amount of translation.

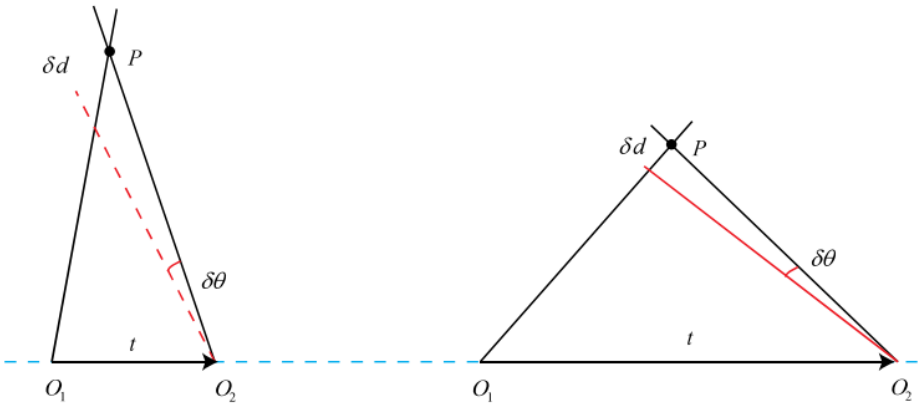


Figure 3.8. The effects of different distance of translation on feature point position accuracy.

As shown in the Fig 3.8 above, when the translation is small, the uncertainty of the depth estimation δd increases. Therefore, in order to increase the accuracy of the depth, it is necessary to increase the translation, but increasing the translation causes a significant change in the appearance of the image, which makes it the difficult to extract and match features. So, triangulation cannot take both the accuracy and the difficulty of matching feature points into account. That is the contradiction of triangulation.

Stereo camera

Stereo camera has a different imaging principle with the monocular camera. In ORB-SLAM, both stereo camera and RGB-D camera could obtain 2 sets of features from two sensors

at the same time. Therefore, the solutions of visual SLAM are the same. Next, stereo camera is taken as an example.

Stereo imaging principle

In stereo camera system, the disparity is the key to obtaining depth information. Assume that the left and right cameras are on the same plane paralleling with the optical axes and the camera parameters such as focal length *f* are the same.

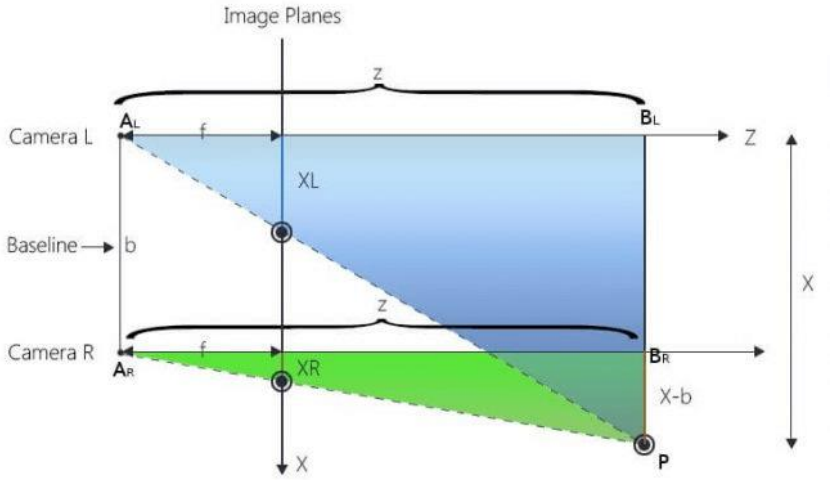


Figure 3.9. The basic imaging principle of stereo camera.

According to the triangle similarity, it can be easily obtained from the Fig 3.8:

$$\frac{z}{f} = \frac{X}{XL}$$
$$\frac{z}{f} = \frac{X - b}{XR} \tag{3.32}$$

the coordinate P (x, z) can be calculated as:

$$z = \frac{f \times b}{XL - XR} = \frac{f \times b}{d}$$
$$x = \frac{XL \times z}{f} = b + \frac{XR \times z}{f} \tag{3.33}$$

where d is the disparity which equals to $XL - XR$. Focal length f and baseline b are obtained by camera calibration. And XL and XR can be calculated by feature point matching results which mentioned in rBRIEF. At the same time, the above 2D plane can be extended to 3D space:

$$y = \frac{YL \times z}{f} = \frac{YR \times z}{f} \tag{3.34}$$

where Y axis is perpendicular to the page in Fig 3.9, YL and YR are the matched points distances to the XOZ plane respectively.

The depth could be directly obtained for the stereo camera. So, there is no need to do triangulation. Based on these feature points locations, the camera trajectory can be restored which will be introduced in the following subsection.

Iterative Closest Point (ICP)

The ICP algorithm was proposed by P.J. Besl and Neil D. McKay in 1992 [26]. Usually, if the RGB-D camera, stereo camera or other tools are used to obtain the 3D point cloud of the object, the point cloud obtained by the same object will have a big difference due to different acquisition devices and different filming directions. The difference mainly relates rotation or translation between two set of point cloud. Therefore, it is necessary to register the point cloud which meets the requirements of stereo SLAM.

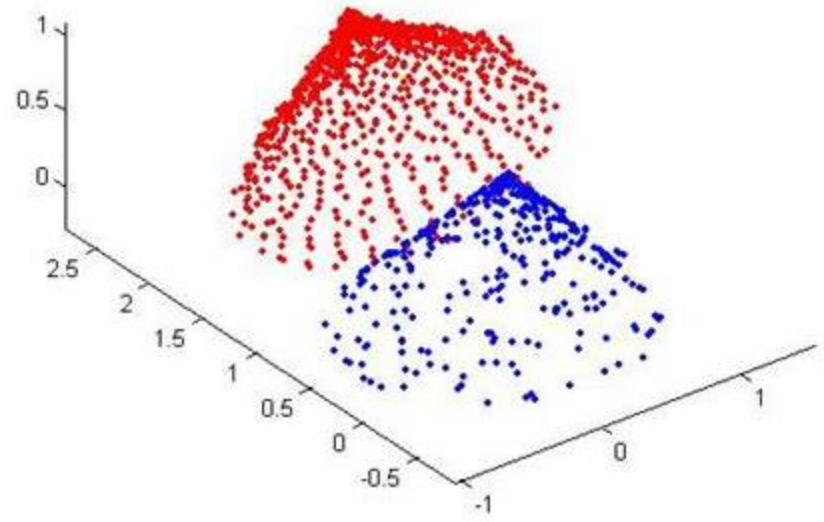


Figure 3.10. Correspondence between two sets of point clouds.

As shown in the Fig 3.10, PR (red point cloud) and RB (blue point cloud) are two sets of points. The algorithm is to calculate how to rotate and translate PB to make PB and PR overlap as much as possible. Two sets of paired point cloud images could be regard as two frames before and after, and the camera trajectory could be calculated between two frames. The authors [27] use the quaternion method to solve the rotation matrix, while K. S. Arun, T. S. Huang and S. D. Blostein utilize SVD to get the final rotation matrix [28]. The SVD was applied in ORB Algorithm and used in this project. According to the rBRIEF algorithm, two sets of matched point clouds can be obtained $P = \{P_1, \cdots, P_n\}$ and $P' = \{p'_1, \ldots, p'_n\}$. There must be a R and a t to relate two corresponding points:

$$p_i = Rp_i^{'} + t \tag{3.35}$$

An error between two points can be defined as:

$$e_i = p_i - (Rp_i^{'} + t) \tag{3.36}$$

The problem can be turn to find the minimum sum of squared error:

$$\min_{R,t} J = \frac{1}{2}\sum_{i=1}^{n} \| p_i - (Rp_i^{'} + t) \|^2 \tag{3.37}$$

Define the center of masses of two set of point cloud are p and p':

$$p = \frac{1}{n}\sum_{i=1}^{n}(p_i) \tag{3.38}$$

$$p^{'} = \frac{1}{n}\sum_{i=1}^{n}(p_i^{'}) \tag{3.39}$$

Then, the new coordinates q_i and q'_i can be obtained by minus the center of masses coordinates:

$$q_i = p_i - p \tag{3.40}$$

$$q_i^{'} = p_i^{'} - p_i^{'} \tag{3.41}$$

The error equation (3.37) can be written as:

$$\min_{R,t} J = \frac{1}{2}\sum_{i=1}^{n} \| p_i - p - R(p_i^{'} - p^{'}) \|^2 + \| p - Rp^{'} - t \|^2 \tag{3.42}$$

The first term is related to *R*, and the second term is related to *R*, *t* and center of mass. Once *R* is obtained, let the second term equals to zero, *t* can be solved. Therefore, the problem turns to minimize the equation:

$$\frac{1}{2}\sum_{i=1}^{n}\| q_i - Rq_i^{'} \|^2 \tag{3.43}$$

After the transformation, the equations need to be optimized is:

$$R\sum_{i=1}^{n} q_i^{'} q_i^T \tag{3.44}$$

$\sum_{i=1}^{n} q_i' q_i^T$ can be singular value decomposed:

$$\sum_{i=1}^{n} q_i^{'} q_i^T = U\Sigma V^T \tag{3.45}$$

Equation (3.45) is a 3x3 matrix. When its rank is 3, $R = UV^T$[23]. Then *t* can be calculated based on the second term of equation (3.41).

3.1.5 Initial Tracking

The ORB-SLAM system can receive images from various sensors, whether stereo, monocular or RGB-D, and turn the original image into a frame containing all the information needed for the ORB-SLAM, including camera intrinsic matrix, distortion parameters, timestamp, scale pyramid information, feature descriptor, depth information, bag of words, camera pose, rotation matrix, translation vector, map points before and after correction, reference keyframe and etc. All the processing is based the images on the level 0 of the pyramid. Once all the data are obtained, the original image is discarded, and the subsequent processing has no relationship with the original image.

ORB-SLAM system could divide into tracking, local mapping and loop detection three threads and the SLAM VO is completed in tracking thread. In order to balance the computational complexity and tracking robustness, the tracking part mainly uses 3D-2D “Perspective-n-Point Camera Pose Estimation” (PnP) which uses the pairs of 3D points in the world coordinate frame and 2D points in the image coordinate frame to estimate the camera pose [29]. The tracking thread includes three models: the motion model “Track with Motion Model”, the keyframe “Track Reference Keyframe” and the “Relocalization”. The three tracking models are designed to obtain a rough initial value of the camera pose, and then Bundle Adjustment is performed for further pose optimization.

Track with motion model

The motion model solves the estimated pose based on the constraint relationship between the two frames. Assuming that the camera is moving at a constant speed, which means R and t is the same with the previous motion. The paired feature points are calculated. If there are enough matched points, this model tracks successful. This motion model is suitable for situations where the speed and direction of motion are relatively consistent and there is no big rotation, such as cars, robots, people and etc. If there are not enough paired feature points, this model will fail and the "Track Reference Keyframe" model will be applied as described next.

Track reference keyframe

If the first motion model fails, the last keyframe is used to restore the camera motion, as the distance between the current frame and the previous keyframe is not very far. The bag of words is used to speed up matching and bundle adjustment (BA) is used to correct the pose. The procedure is shown below:

1. Calculate the bag of words (BoW) of the current frame [30][31].
2. Compare the current BoW with the previous keyframe BoW. If the matched words more than 20, the BA is used to correct the pose.

Relocalization

If the matching between the current frame and the nearest neighbor keyframe also fails, it means that the current frame has been lost and the real position cannot be determined. At this point, the only way to find the location is to match all the keyframe using BoW as the procedure is shown below:

1. The BoW is found according to the current keyframe.
2. Detect the candidate keyframes based on the similar BoW.
3. Filter out the keyframes with the numbers of BoW whose range are in 0.8-1 times maximum BoW matched.
4. Calculate the similarity score for the BoW matching quality.
5. Calculate the cumulative similarity score for keyframes which have the same observation area

6. Find out the keyframe with the highest BoW score plus same observation score.
7. Utilize PnP to estimate pose and relocalization.

3.1.6 Track Local Map

Once the initial camera pose and initial feature match are obtained through the previous mentioned three models, the map points can be used in the current frame to search for more matches. But building a complete map costs lots of computation. Therefore, the local map is used to reduce the number of map points that need maintenance. Updating local maps include updating local keyframes and local map points. The Fig 3.11 shows the process:

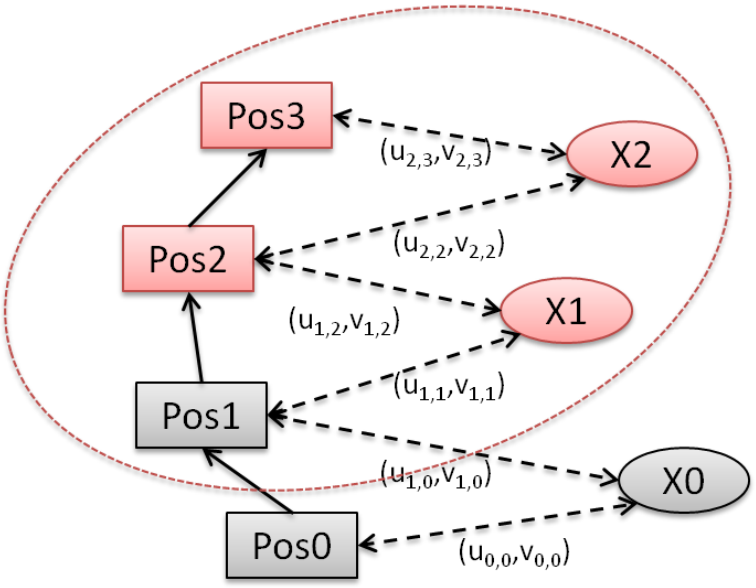


Figure 3.11. The connections among map points and poses.

Pos3 is the current keyframe and Pos2 is the previous keyframe. Map points X2 can be observed by Pos3 and X1 can be observed by Pos2. These data are related to current keyframe Pos3. So, Pos3, X2, Pos2, X2 and the (u,v) transformations among them directly participate in local optimization. Meanwhile, X1 can be observed by Pos1. But it is not the direct relations ship to Pos3. Therefore, Pos1 is not be optimized and the transformation between Pos1 and X1 $(U_{1,1}, V_{1,1})$ is fixed in local optimization. The red ellipse circle indicates the parts which participate in local optimization and they can be replaced by the optimized value. As the process continues,

the new keyframes and corresponding map points come in, the optimized camera pose and map points based on the previous keyframes will turn into black. These data will not be considered in the next optimization and become Pos1, X0 and Pos0 in the Fig3.11.

New keyframe creation

Keyframes are mentioned in previous subsections, but they not introduced in detail. This subsection will explain how keyframes are generated. If the following conditions are met, the keyframes are inserted:

1. There are at least 20 frames from the last relocalization;
2. The local map thread is idle, or 20 frames have been added since the last time the keyframe added;
3. The current frame tracks at least map 50 points which ensures the accuracy of tracking positioning;
4. The number of map points in the current frame is less than 90% map points from last key frame which ensures significant visual changes between keyframes.

If the requirements for creating a keyframe are met, the keyframe is created and sent to the LocalMapping thread.

3.2 Local Mapping

The Local Mapping module is used to place the keyframes sent from Tracking in the mlNewKeyFrame list and process new keyframes, map points; generate new map points, implement local bundle adjustment and etc. The main job is to maintain the local map.

As mentioned in Track Local Map subsection, the Tracking thread adds image frames that meet certain conditions to keyframes, but in fact the keyframes in the ORB-SLAM are in large quantities, which ensures the accuracy of localization. At the same time, the LocalMapping thread will eliminate keyframes, ensuring that the number of keyframes does not increase indefinitely. This subsection introduces how the local map get updated and maintained.

3.2.1 Map Point Culling

The map point culling module checks the new added local map points. If the map point is to be saved, the map point must be rigorously tested to ensure it could be properly tracked and triangulated. Once the map point meets one of the following conditions, it will be eliminated:

1. The number of normal frames which successfully tracked the map points less than the number of normal frames that should be observed for the map point. If so, it indicates that such a map point is within the field of view but is rarely detected by a normal frame;
2. Map points can not to be identified by all adjacent keyframes. There are some differences in the different ORB-SLAM solutions. Stereo and RGB-D require two adjacent keyframes, while monocular SLAM requires three consecutive frames detect the map point.

If the map point can be observed from three or more keyframes, it means the map point are in high quality.

3.2.2 New Map Points Insertion

New map points insertion module recovers some new map points based on the current keyframe, excluding the local map points that matched the current keyframe. The specific steps are as follows:

1. Find 10 keyframes (20 in the monocular mode) that have same observation area with the current frame;
2. Check if the movement between two frames is too short. As the map points are restored according the triangulation, the feature in triangulation is applied. If the ratio between camera movement distance and map point depth is small, that means the movement is too short. The map points are not reliable and they will not be insert to the map;
3. Calculating the fundamental matrix F between two frames;
4. Search for matches that satisfy the epipolar constraint;
5. Complete the triangulation of the matching results to create a new map point;

3.2.3 Local Bundle Adjustment

Bundle Adjustment (BA) refers to the extraction of the optimal 3D model and the intrinsic and extrinsic of the camera from visual reconstruction [32]. After the camera pose and the feature point position are optimally adjusted, the process of collecting bundles of light rays reflected from each feature point to the camera's optical center is referred to as BA. In visual SLAM, BA usually plays a core role. It not only has high precision, but also has a good real-time performance.

3.2.4 Redundant Keyframe Culling

This module is relatively simple. If 90% of the map points detected by a keyframe are simultaneously detected in no less than three other keyframes with the same or more precise scale, the keyframe is considered redundant and need to be culled.

The insertion and culling of keyframes and map points in ORB-SLAM is based on the strict criteria, so the calculation amount is well limited under the premise of improving the accuracy of localization and mapping.

3.3 Loop Closing

The Loop Closing thread is a very important part of the SLAM system. Due to the cumulative error (drifting) of the VO process, the main task of the Loop Closing thread is to detect the closed loop using BoW, that is, to determine whether it has been reached before. After detecting the closed loop, Sim3 transformation is calculated to close the loop [33]. At last, the global BA is applied to optimize the keyframe poses and map points cumulative error to an acceptable range.

4. EXPERIMENTS

The ORB-SLAM algorithm has been explained in Chapter 3. Three experiments will be introduced to evaluate the performance of the algorithm and platform. First experiment is performed in the office rooms with the minimum features to find how these features affect the monocular SLAM initialization. Second experiment is designed to compare indoor performance among three different cameras. The last experiment is performed in both static and dynamic environments to test the algorithm robustness. The hardware and software setup is introduced next.

4.1 Experimental Hardware and Software Setup

4.1.1 Robot Operating System (ROS)

ROS is an open source operating system designed for robots. It was born from STAIR (STanford Artificial Intelligence Robot) and Personal Robotics (PR) project in 2007 [34]. ROS runs on Linux systems at the beginning. It did not support windows system until the latest ROS Melodic Morenia in 2018. There are total 12 releases and ROS Indigo which released in 2014 is used for ORB-SLAM algorithm.

ROS has three communication method: Topics, Services and Actions. Topics is the most commonly used. It is a one-way communication method. Node links to the terminal sensors like camera shown in Fig 4.1. The process of communication between nodes can described as:

1. Nodes are registered by node manager;
2. Nodes publish and subscribe topics;
3. Topics are transmitted in the node manager.

All the transitions are one-way and each node runs separately. For real-time, periodic messages, topics communication is the best choice. Therefore, it is used in this project to transfer images between master (robot) and server (laptop). In this project, data processing runs on robot while the ORB-SLAM runs on the laptop.

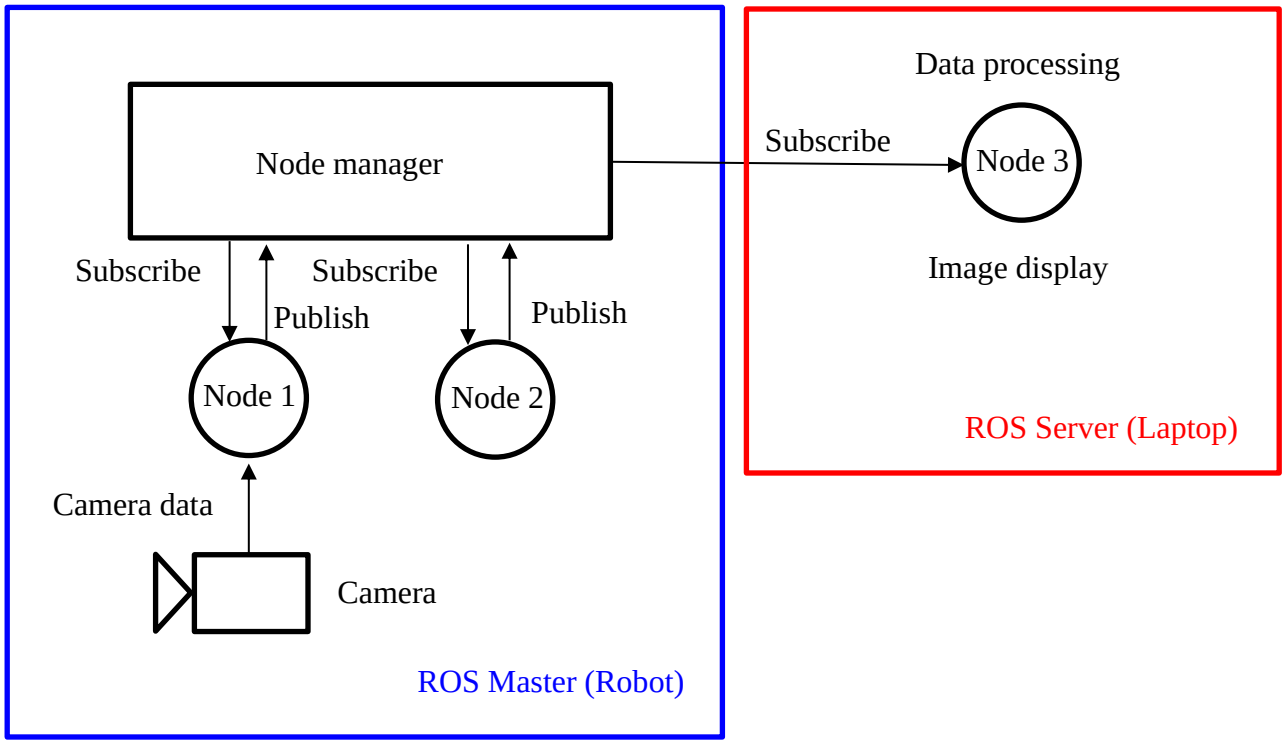


Figure 4.1. The framework of topic communication between the robot and the laptop.

4.1.2 Jackal Mobile Robot

The mobile robot Jackal used in this project with three different kind of cameras is shown in Fig 4.2. Jackal is a 4-wheel drive mobile robot which give it abilities to run through rough and uneven surface. Each wheel cannot change directions. However, it turns in differential mode which means it could turn in situ.

The hardware is specially designed for the robot. The motherboard carries a 2.4GHz Celeron J1800 duo core processer and 2GB memory card. Ethernet and WIFI are both available for the communication. The detailed specification of Jackal is attached in Appendix B. Ubuntu 14.4 modified by the Clearpath Robotics, Inc is originally installed as the operation system. The ROS Indigo comes with the Ubuntu system and provides an access to manipulate the robot. The robot can receive commands from both PS4 controller by Bluetooth and server computer by WIFI. The robot does not carry a monitor as user-interface and the wireless communication does not work at the beginning. So, an extra monitor with VGA port and a keyboard are great helpful. If the robot successfully connects to the router, the robot can be accessed wirelessly from another computer (server) via secure shell (SSH).

Figure 4.2. The Jackal mobile robot mount with both Bumblebee camera and RGB-D camera.

4.1.3 Bumblebee Camera

The Bumblebee2 stereo camera is mounted in the front of the robot deck. It provides 648 × 488 resolutions at 48 FPS color images from two Sony ICX204 sensors. Low resolution means less calculation and less data needs to be transfer and that is of great benefit for the real-time performance. The camera connects to the adapter on the motherboard, as it utilizes the standard IEEE1394b port which is not commonly seen on the consumer level motherboard.

4.1.4 RGB-D Camera

Microsoft published an epoch-making camera called Kinect V1 in 2010. It combined monocular camera and infrared sensor together. As a result, the camera had the ability to get depth information directly. Comparing with the Bumblebee camera, Kinect has a big disadvantage that apart from a data cable, it needs an extra power cable. Unfortunately, the Jackal robot cannot provide such port. So, the connection of this camera is quite different from stereo. The camera directly connects to server laptop, rather than Jackal as shown in Fig 4.3. And that connection way limits the scanning range of the camera. The RGB-D camera, likes stereo camera, can provide two set of images: one from monocular camera, another from infrared sensor. According to the ORB-SLAM algorithm, two set of images are not utilized directly. It extracts features from the registered

images (the colored image with depth information processed from monocular camera and infrared sensor). Therefore, it is necessary to perform the registration to combine and match the image from the camera and infrared sensor together before using the camera.

Figure 4.3. Kinect camera connects to the laptop via a USB-cable.

4.1.5 Calibration

Three platforms are usually used to calibrate the camera on computer, which are MATLAB, OpenCV and ROS. In our case, ROS is used for convenience. The checkboard is prepared according to Zhang's calibration method which is shown in Fig 4.4. It has 7x8 squares and each square has 75mm side length.

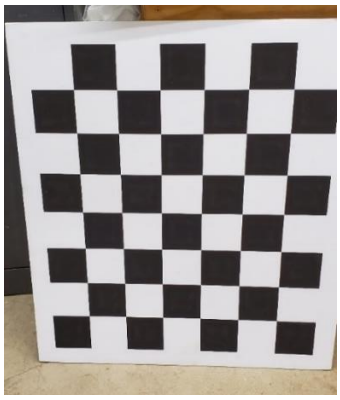


Figure 4.4. 7x8 Checkboard used for calibration.

Monocular and stereo calibration are similar. First, Bumblebee camera needs to launch in monocular or stereo mode. Then, the checkboard is shown in front of the camera. To calibrate the camera, the checkboard needs to be moved forward and backward, up and down, left and right, and skewed inward and outward. The calibration data includes camera matrix which describes the camera intrinsic, distortion coefficients, rectification matrix and projection matrix which are mentioned in Chapter 2. The lag of the images occurs when the stereo camera is calibrated due to wireless camera data communication. As introduced in Chapter 2, camera calibration is based on images rather than video stream. As long as the computer can obtain the sampled images, the lag can't affect the accuracy of the calibration result.

RGB-D camera needs to calibrate both depth camera and color camera. To get the intrinsic data from the depth camera, the infrared radiation (IR) emitter needs to be obscured. Because it emits speckle and an effect on the detection of checkerboard corners. The color camera calibration is the same as monocular camera. The .yaml file relating to ORB_SLAM needs to be modified according to the corresponding calibration data. All the calibration data is shown in Table 4.1. Camera intrinsic are fx, fy, cx and cy. Radial distortion parameters are k1 and k2. Tangential distortion parameters are p1 and p2.

Table 4.1. Camera calibration parameters used in the project

Calibration parameters	Monocular Camera	RGB-D Camera	Stereo Camera	
			Left	Right
fx	526.017008	504.398153	519.476137	524.701825
fy	527.002117	509.612491	517.878998	522.801534
cx	315.953630	332.645276	338.677098	323.188121
cy	242.022883	254.755404	253.996175	253.996175
k1	-0.358357	0.127508	-0.349529	-0.353332
k2	0.144488	-0.214968	0.130417	0.139960
p1	0.000866	-0.002119	-0.001112	-0.000758
p2	-0.000393	0.002685	-0.000015	-0.000439

4.2 Experiment 1

The ORB-SLAM runs on the premise of successful initialization. While the initialization has specific requirements for the number of feature points that can be extracted from the environment. This experiment reveals the limitation when initialization is performed. All three cameras are consistent in the number of feature points required. For the convenience of operation, the monocular camera is used in the experiment.

The monocular SLAM is quite different from the other two visual SLAM solutions. The depth of the map points is estimated not directly from the measurement calculation. So, it needs camera movements to find the feature points location on the map and it is easily lost tracking when scanning the environment. This limitation causes many problems when the monocular SLAM is used. This experiment aims to explain the shortcoming of the monocular SLAM. The minimum matched feature points need for initialization is set as 100. That means the camera should find at least 100 pairs of features from the adjacent two frames. If the number of the matched points is less than 100, the initialization will fail and the VSLAM cannot continue.

Two office rooms are chosen as the environment. One room is filled with some objects like desk, desktop, chairs and backpacks and another is empty as the Fig 4.5 shows.

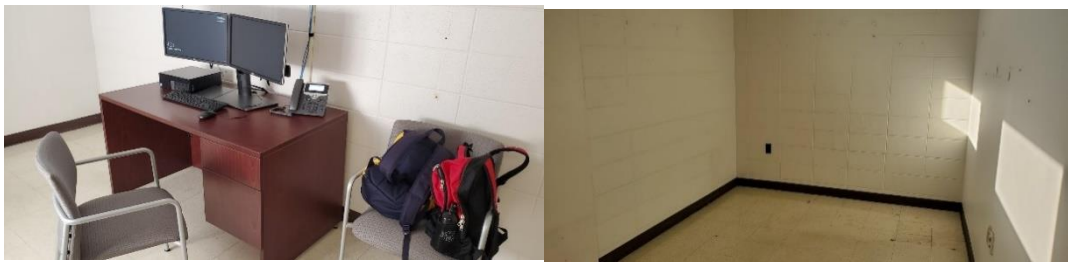
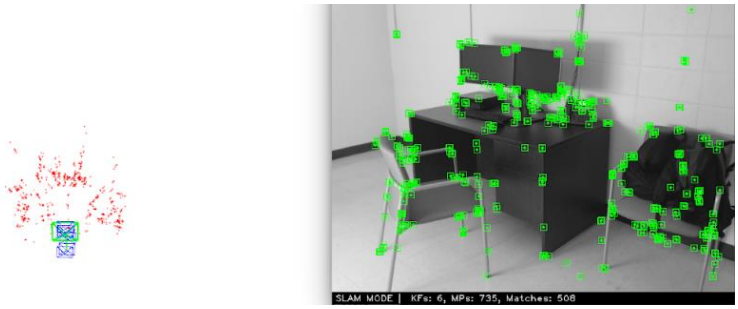


Figure 4.5. Two office rooms with different furniture.

Monocular SLAM initializes in two rooms. It is easily to extract feature points in the first room (left). It takes about 735 feature points from this image (Fig 4.6) and matches 508 pairs among them.



(a) (b)

Figure 4.6. Numbers of feature points found and matched. Red dots in (a) shows the corresponding green features from (b). Green frame in (a) is the current camera position in the map.

While the same test could not be done in the room without furniture. It is hard to extract enough features for initialization.

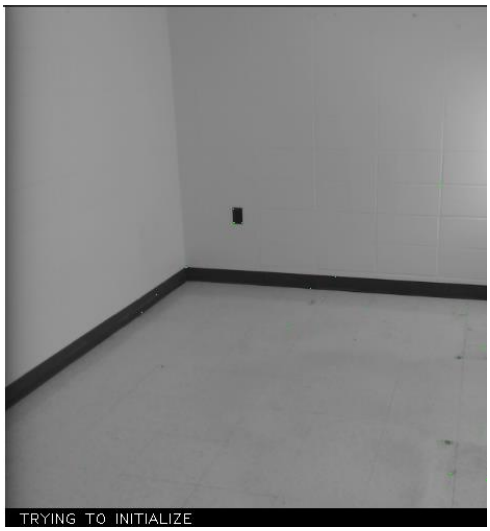


Figure 4.7. Fail to initialize in this room. The map points and camera pose cannot be obtained in the map.

The Fig 4.7 shows that there are few tiny dots extracted from the images, much less than the feature points extracted from room with furniture and the number of matched feature points is 84. These environments do not have much texture. From the Fig 4.6 and Fig 4.7, it is easy to find that if the image lacks features, the grey level of the surrounding pixels should be similar. When comparing the difference of pixels around, it is hard to filter out the feature points which meets the FAST algorithm requirement according to the flow chart in Chapter 3. So, it is hard to extract enough feature points.

4.3 Experiment 2

Most indoor environment is constituted of corridors and rooms. Therefore, this experiment is conduct in two environments to evaluate the performance of ORB-SLAM from different cameras. The corridor is selected as the first environment and the computer lab is used in the second environment.

4.3.1 Performance in Corridor

It is a typical indoor corridor on campus shown in Fig 4.8. Labs and offices lay on both sides and the notice boards are hanged on the wall.

(a) (b)

Figure 4.8. The typical corridor. (a) is taken at the north corner. (b) is taken at the south corner.

All three cameras are used to scan the environment, which means the mobile robot needs to run through the same trajectory 3 times. Because three cameras cannot run simultaneously on the robot. To guarantee the robot trajectories are the same for each scan, the mobile robot moves along the bricks in the middle of the corridor.

First, the RGB-D camera is used. The mobile robot moves forward and turn right at the corner and stops at the end of the corridor. Then, it reveres to the corner and turns its directions to the south. After scanning the corridor, it turns left to the senior design lab and reveres to the corner. Finally, the robot turns its directions to the north, and move to the original place in order to complete a loop.

RGB-D camera as discussed in Chapter 3 provides depth information. So, it has the best performance and accuracy. As mentioned before, the Kinect camera needs an extra power cable. Hopefully, some sockets lay on different area of the corridor. The mobile robot could run freely in this environment. The SLAM result is shown in Fig 4.9.

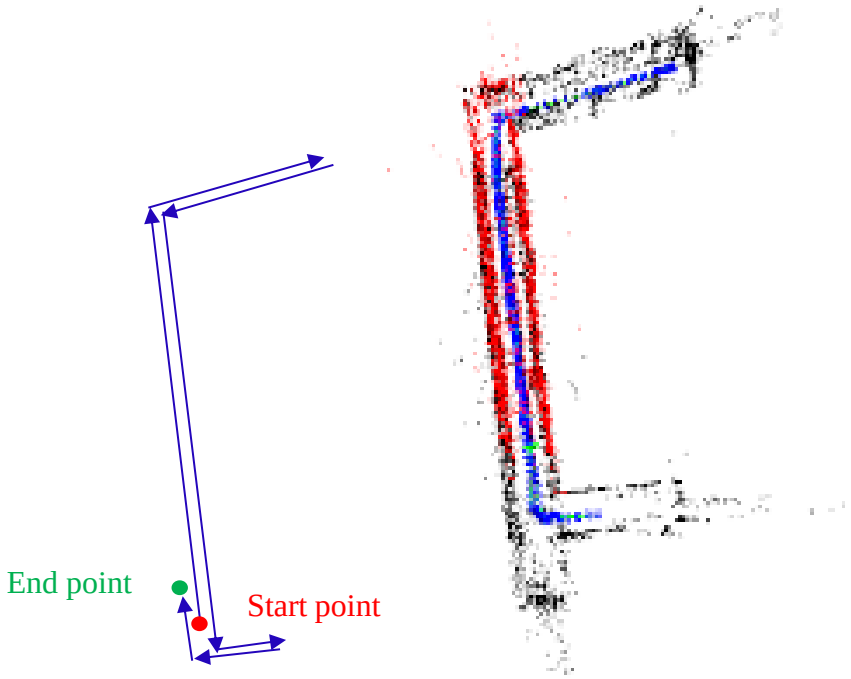


Figure 4.9. Camera trajectory and map point from Kinect camera. The red points are the estimate map points which can be detected at the camera's current location while black points are optimized map points. Blue squares represent the locations and pose of the keyframes and record the camera trajectory.

The RGB-D SLAM extracts 288 keyframes and 5494 map points. The shape of the corridors has been recorded. There are not many spares map points outside the corridor area. The RGB-D SLAM result is taken as the ground truth. That means another two datasets from stereo SLAM and monocular SLAM are compared to the RGB-D SLAM result. The Root Mean Squared Error (RMSE) value is used to evaluate the camera trajectory [35]. The equation for calculating RMSE is presented in Appendix C.

Then, the 2D map from RGB-D camera is built in Fig 4.10. This map is filtered by a pass-through filter which is commonly used in point cloud map to filter out the sparse points. The redline reveals the walls in this environment. It could be noticed that, there are some map points beyond the red line at the top right. The reason of this phenomenon is that there is a door with glass at the end of this corridor. The camera takes features behind the doors via glass which leads to some points laying outside the corridor. The camera also extracts features from the ceiling and the ground, which leads to many map points laying in the middle of the corridors in the 2D map.

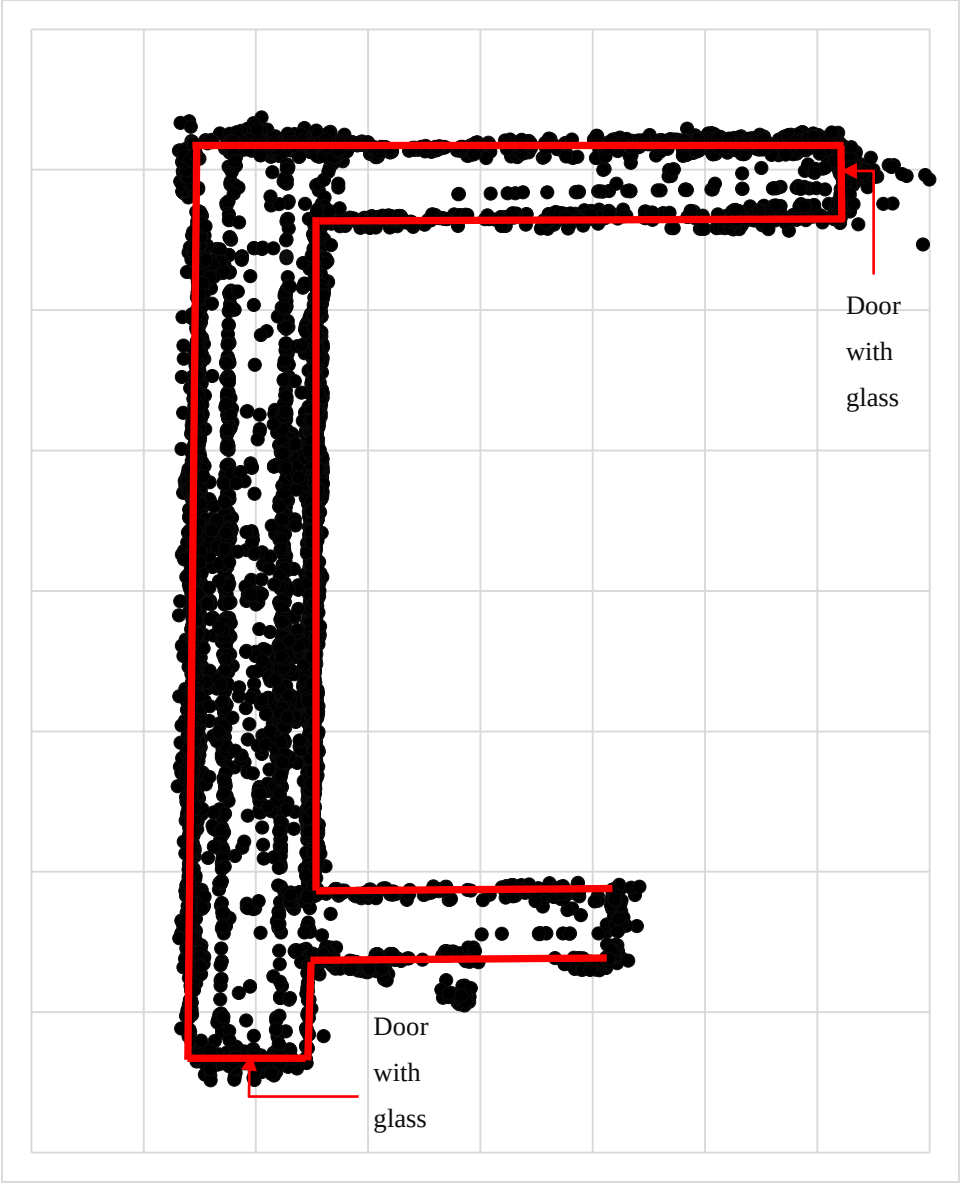


Figure 4.10. 2D corridor map points after filtering from RGB-D camera.

After RGB-D SLAM, the Bumblebee camera is set in monocular mode which means the right camera is used. Unlike RGB-D SLAM, the monocular SLAM requires movement of the camera to initialize the system. The starting location for monocular SLAM is little behind the original location. The robot could turn directions on situ. But from the features of the monocular visual SLAM mode in Chapter 3, the movement between two frames are regarded as translation, no pure rotation involved. If the robot turns on situ, the camera will turn the rotation into translation and fail to record this rotation. As a result, the trajectory and map are misled even the bundle

adjustment and loop closing cannot correct this error. The trajectory at the corner involves translation rather than rotation. The robot follows the same trajectory. The SLAM result is shown in Fig 4.11:

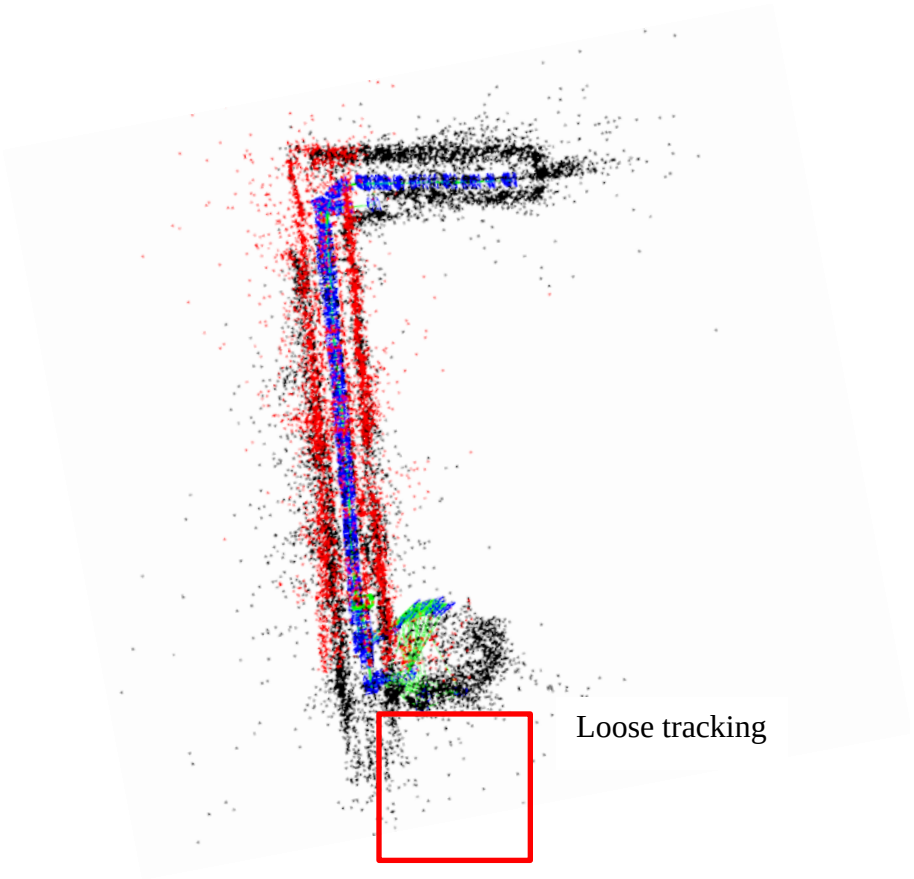


Figure 4.11. Camera trajectory and map point from monocular camera. The red points are the estimate map points which can be detected at the camera's current location while black points are optimized map points. Blue squares represent the locations and pose of the keyframes and record the camera trajectory. The monocular SLAM loses tracking at bottom of the map.

The monocular SLAM extracts 390 keyframes and 19845 map points. The map shows the shape of the corridor. The camera trajectory mostly coincides and the loop is closed correctly. The camera loses tracking when the robot turns directions at the bottom, hence the map points and trajectory are not accurate there.

From Chapter 3, it is obvious that the relocalization is based on BoW. While BoW is based on features extracted by ORB algorithm. When the robot turns at the right corner, the images camera captured are plain walls which lack features and the captured features are not robust at different orientation. That is to say, the features cannot be observed by adjacent frames. Therefore,

the BoW of these keyframes are not big. When it needs to compare the different BoW among frames, the lack of features will lead to the inaccurate matching. Once the match is found, these features will also affect the recovery of the pose when PnP is utilized which introduced in Chapter 3.

At last, Bumblebee camera is used in the stereo mode. In this situation, the data communication is doubled compared with the monocular SLAM. As mentioned in stereo camera calibration, the same problem, lag, occurs again. The average lag is more than 2 second. The lag has impact on the precision of the system. From the algorithm, it is known that the keyframe is picked in sequence. If the sequence is messed up, it is impossible to build the map correctly. The lag prevents the computer from checking each frame of the image and the computer has to skip many frames when the new frames come in. What makes the problem worse is that the keyframes are selected from these normal frames. As a result, the system hardly tracks the camera trajectory and maps the environment precisely. The mobile robot has to move at a very low speed to help the computer keep up the pace with the image refreshing rate. As if the images picked as keyframes differ a lot, it is easily lost tracking. Because of the lag, it is not easy to find out where it lost tracking. The robot has to reverse slowly trying to re-localize its position. If the camera successfully re-track the map, there is great possibility to cause the error.

This scan generates 468 keyframes and 29107 map points. The map is shown in Fig 4.12. Before doing this experiment, a better result from stereo camera is expected as stereo camera can directly provide depth information. But from the Fig 4.12, it can be found that the trajectory is not accurate and the map is misplaced. The camera loses tracking several times especially at corners. Every time the camera re-tracks the map, the error on directions cannot be corrected. And the error accumulates to the end which leads to an unacceptable loop closing result. It is obviously that, the stereo ORB-SLAM on this setup fails in this environment. To overcome the lag problem, the cable communication setup is implemented in the next experiment.

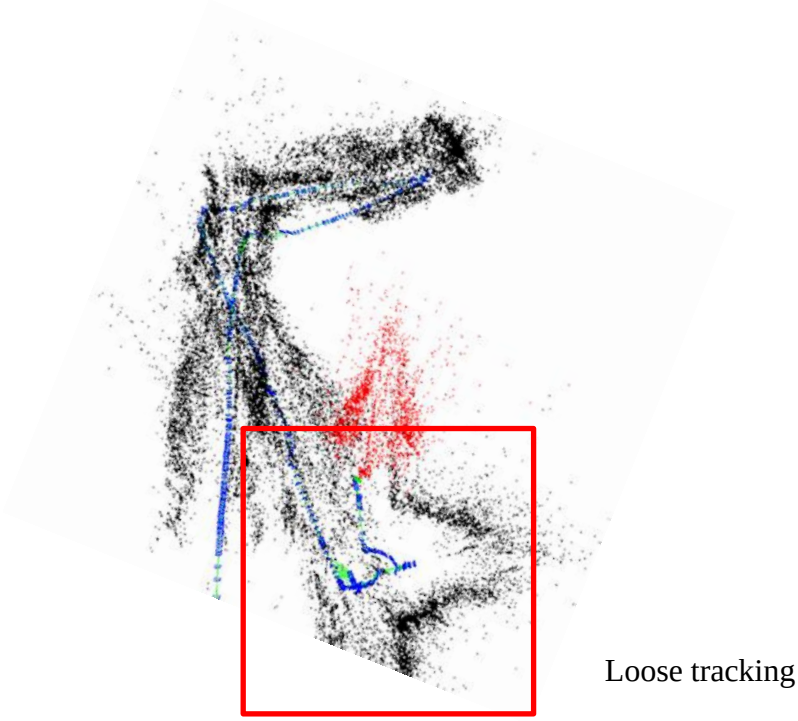


Figure 4.12. Camera trajectory and map point from stereo camera. The red points are the estimate map points which can be detected at the camera's current location while black points are optimized map points. Blue squares represent the locations and pose of the keyframes and record the camera trajectory.

Monocular SLAM performance

First, the connection needs to be built between the keyframes from 2 datasets. Each dataset is in different scale and the number of the keyframes is different. The factor difference is multiplied to the monocular dataset to correct the scales. 244 pairs of the keyframes are found and after the calculation. The RMSE value is 0.69 meter. The trajectory error in this small-scale environment is shown in Table 4.2.

Table 4.2. Trajectory error from monocular SLAM in the corridor

	Monocular
RMSE	0.693m
Mean	0.565m
Standard Deviation	0.402m
Min	0.122m
Max	1.823m
No. Pairs	244

Stereo SLAM performance

Only 244 pairs are found between stereo and RGB-D datasets. The RMSE of the trajectory is 1.95 meter which is more than half of the corridor width. The maximum error even comes to 3.06 meter. The analysis results are consistent with the previous rough observations from the map shown in Table 4.3. Stereo SLAM on the robot has the bad performance in this environment.

Table 4.3. Trajectory error from stereo SLAM in the corridor

	Stereo
RMSE	1.952m
Mean	1.770m
Standard Deviation	0.822m
Min	0.285m
Max	3.063m
No. Pairs	244

4.3.2 Performance in Computer Lab

The computer lab has three aisles and the rest of the space is filled with computers, oscilloscopes, desks and other experiment tools which is shown in Fig 4.13.

Figure 4.13. A typical Engineering laboratory.

Comparing with the corridor environment, the lab is a separate room and smaller. It has some similar features such as three aisles. The experiment starts with the RGB-D SLAM. The robot launches at the entrance of the lab, facing north. Then it moves forward to scan the first aisle and reverses at the end of the aisle. After finishing the scan of the first aisle, the robot turns to east and moves to the second aisle. The robot runs on similar trajectories when scanning the second and third aisles. Finally, the robot turns to west at the east-south corner of the lab and move directly

to the original point. It runs along the lines on the ground which helps on keeping the same trajectory when the rest experiments are performed.

The trajectory from RGB-D SLAM is shown in Fig 4.14.

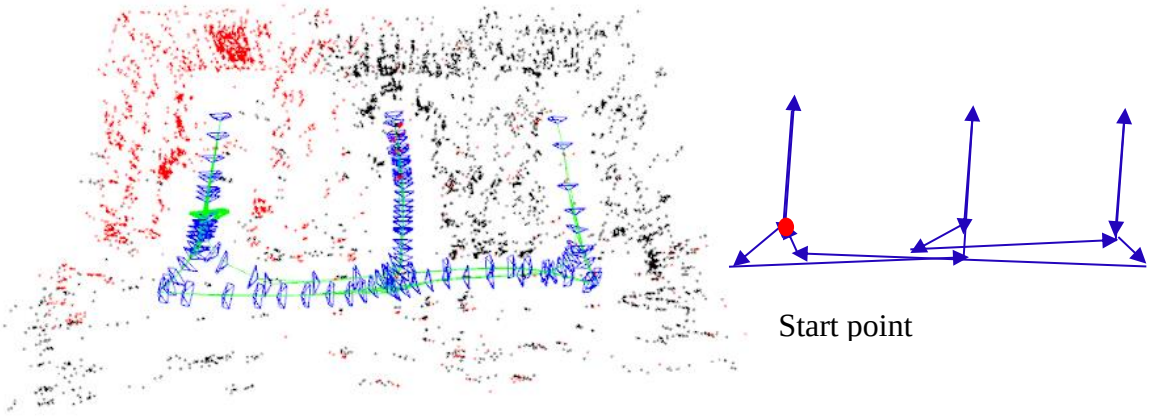


Figure 4.14. Camera trajectory and map point from RGB-D camera. The red points are the estimate map points which can be detected at the camera's current location while black points are optimized map points. Blue squares represent the locations and pose of the keyframes and green line strings all the keyframes together which records the camera trajectory. Start point and end point are in the same location.

RGB-D SLAM generates 110 keyframes and 6577 map points. The trajectory likes the letter "E". The 3D shape of the lab could be recognized from the map. The trajectory of RGB-D camera is taken as the ground truth.

Then, monocular SLAM is done in the same way. The result is shown in Fig 4.15. Monocular SLAM totally generates 197 keyframes and 10220 map points. The trajectory when the robot reverses overlaps the trajectory when the robot move forward. The objects constructed by map points are hard to recognized, but the map points roughly describe the shape of the room. The problem comes from the height of the camera. The monocular camera is fixed on the robot deck. The camera hardly sees the desktops and oscilloscopes on the desk. What can be observed at low level mainly are legs from the chairs and desks. What's worse, because of the 15 degrees of pitch angle from the camera, the light from the ceiling could be seen almost everywhere. That high intensity lighting keeps the aperture becoming smaller. As a result, the features from the dark place hardly can be found.

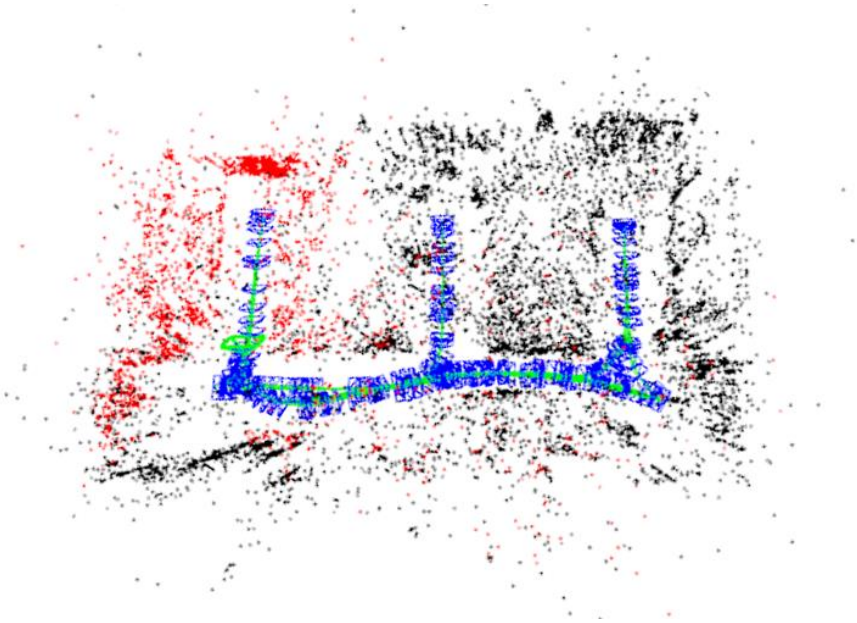


Figure 4.15. Camera trajectory and map point from monocular camera. The red points are the estimate map points which can be detected at the camera's current location while black points are optimized map points. Blue squares represent the locations and pose of the keyframes and green line strings all the keyframes together which records the camera trajectory.

The stereo SLAM is performed at last. The map is shown in Fig 4.16.

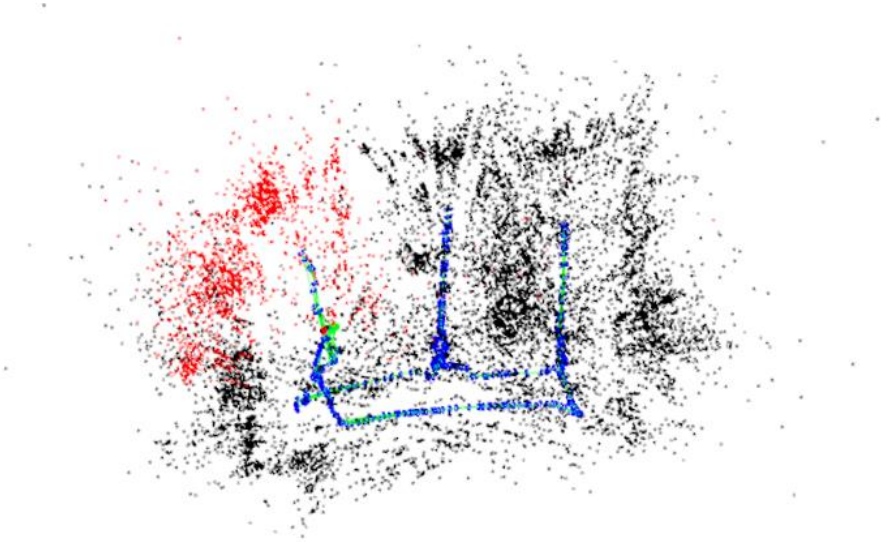


Figure 4.16. Features extracted and camera trajectory, map point from the stereo camera. The red points are the estimate map points which can be detected at the camera's current location while black points are optimized map points. Blue squares represent the locations and pose of the keyframes and green line strings all the keyframes together which records the camera trajectory.

Stereo SLAM generates 358 keyframes and 11356 map points from the Lab. The trajectory should like letter “E”, but the results shown in Fig 4.15 does not show like that. Also, the map points are not accurate. The stereo SLAM closes the loop at the end point and correct the trajectory and map points.

Monocular SLAM performance

In this comparison, 99 pairs of keyframes are used to evaluate. Scale of the map needs to be correct as it is done in corridor experiment. The RMSE of the trajectory is 0.60 meters and other indicators shown in Table 4.4.

Table 4.4. Error of trajectory from monocular SLAM in the lab

	Monocular
RMSE	0.599m
Mean	0.532m
Standard Deviation	0.274m
Min	0.117m
Max	1.257m
No. Pairs	99

Stereo SLAM performance

99 pairs of keyframes are found for evaluation. The RMSE of the trajectory is 0.67 meters which is much smaller than the RMSE from the corridor trajectory. The map may show that the stereo and monocular result are not close, but other indicators shown in Table 4.5 reveals that stereo and monocular SLAM have the similar performance in the lab.

Table 4.5. Error of trajectory from stereo SLAM in the lab

	Stereo
RMSE	0.668m
Mean	0.588m
Standard Deviation	0.316m
Min	0.133m
Max	1.229m
No. Pairs	99

Lag problem occurs each time when the stereo SLAM is implemented. The next experiment overcomes the lag by using the cable communication between the robot and the laptop. The Experiment 2 is performed three times and the results above are the best among them.

4.3.3 ORB-SLAM Performance with Wired Communication

Considering sending the image data wirelessly to laptop for processing leads to the lag, especially when stereo SLAM is performed, the new setup shown in Fig 4.17 is used. A router is used as a bridge between the robot and the laptop. Three devices are connected by cable to improve the communication environment.

Figure 4.17. Cable communication is built between the laptop and the robot.

Under this setup, there is no obvious lag when the stereo SLAM is performed. The laptop is placed on the deck of the robot when experiments are conduct again in the same way. The map of the lab from monocular SLAM is shown in Fig 4.18.

The monocular SLAM generates 175 keyframes and 9036 map points which are less than the previous monocular SLAM map. The map shows the trajectory and the outline of the environment.

Then, the stereo SLAM is performed in the same way. The map is shown in Fig 4.19. The stereo SLAM extracts 463 keyframes and 13320 map points. To evaluate the improvement of this setup, the comparison combines two experiments together. The detailed results are shown in Table 4.6.

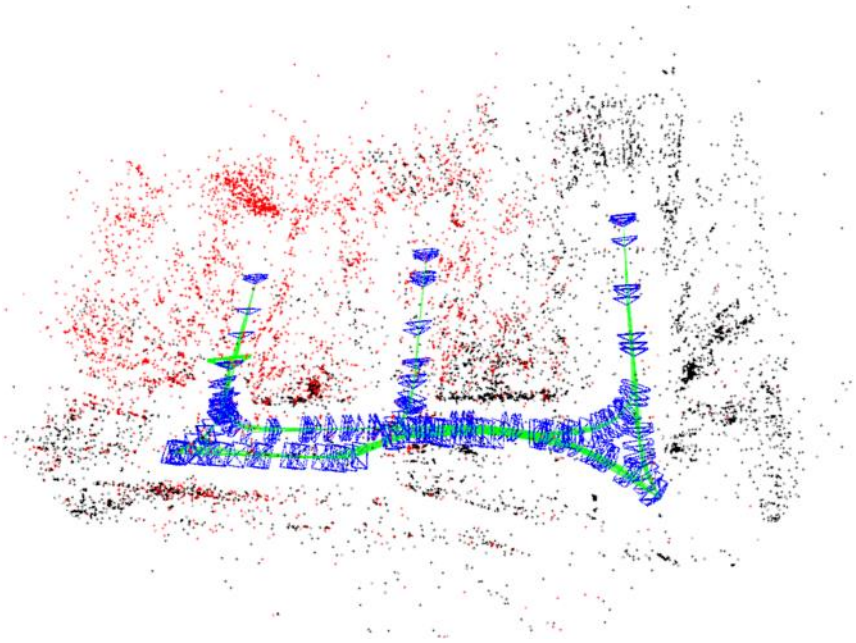


Figure 4.18. Map points and trajectories from monocular SLAM (cable). The red points are the estimate map points which can be detected at the camera's current location while black points are optimized map points. Blue squares represent the locations and pose of the keyframes and green line strings all the keyframes together which records the camera trajectory.

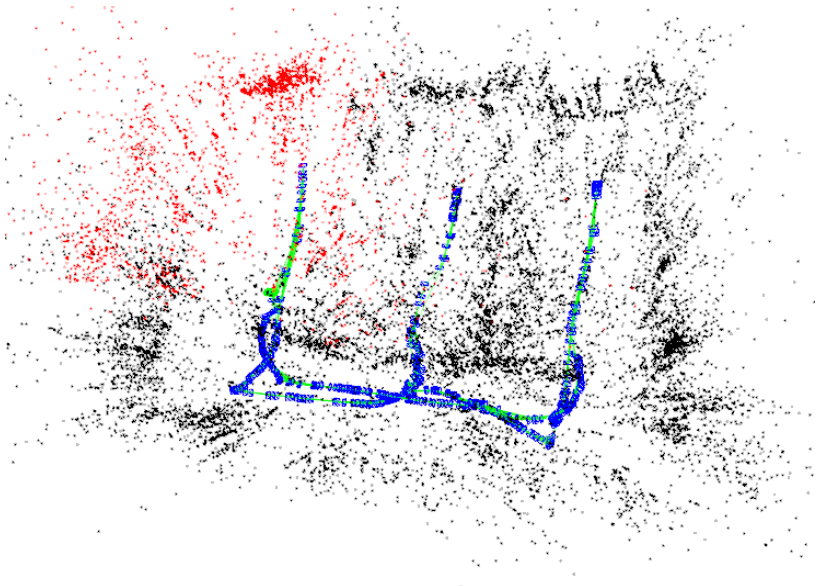


Figure 4.19. Map points and trajectories from stereo SLAM (cable). The red points are the estimate map points which can be detected at the camera's current location while black points are optimized map points. Blue squares represent the locations and pose of the keyframes and green line strings all the keyframes together which records the camera trajectory.

Table 4.6. The improvement from the cable setup

	Monocular SLAM			Stereo SLAM		
	Wireless Setup(m)	Cable Setup(m)	Improvement	Wireless Setup(m)	Cable Setup(m)	Improvement
RMSE	0.599	0.574	4.1%	0.668	0.656	1.7%
Mean	0.532	0.507	4.7%	0.588	0.576	2.0%
Standard Deviation	0.274	0.269	1.9%	0.316	0.313	0.8%
Min	0.117	0.077	33.4%	0.133	0.094	29.0%
Max	1.257	1.207	3.9%	1.229	1.263	-2.8%
No. Pairs	99					

From the table, the improvement is little especially when stereo SLAM is performed. In other words, lag does not affect the accuracy of the map very much if there is no lost tracking during the whole procedure. Therefore, the conclusion should be lag have little direct impact to the data processing if the program runs smoothly. But it does make lost tracking easier which indirectly affect the accuracy. After eliminating the lag problem, it is possible that the capability of the CPU on robot limits the stereo SLAM performance. Compared with monocular SLAM, the data need to process are doubled. The CPU on robot may not powerful to handle this heavy computation. If the CPU of the robot are upgraded, it will be expected to obtain an improvement from stereo SLAM.

Experiment 2 shows the ORB-SLAM has the potential to navigate indoor. It can be applied in the future mobile robot food delivery in restaurant.

4.4 Experiment 3

In this experiment, the robustness of the algorithm in dynamic environment would be tested. The first-floor corridor at CLO building is chosen as the environment which is shown in Fig 4.20. This experiment is divided into 2 parts. In the first part, RGB-D SLAM is performed during weekend, as there are no pedestrians walking around. The second part is performed at weekdays in order to involve more dynamic features.

(a) (b)

(c) (d)

Figure 4.20. The four corners of the CLO corridors.

The robot starts at the south corner of the corridor(a) which is shown in Fig 4.19 and faces towards the west. It slowly moves to the first junction(b) and turns right. Then, it runs in the middle of the corridor until the next turning(c) and turns right again. After a short straight movement, the robot arrives at the café(d) and turns right. It runs through the hall of the CLO and returns to the origin corner(a). At last, the robot turns right 90 degrees to help the camera close the loop. The trajectory should be a rectangle. The map of the first part is shown in Fig 4.21. The RGB-D SLAM extracts 1018 keyframes and 18998 map points. The blue frames reveal the trajectory of the camera. The map points distribute along the corridors. Then, the 2D map of this environment is built in Fig 4.22. The walls which have been measured are indicated in the map. Then, the comparison between the actual measurements of the corridor width and calculation based on the map points are made.

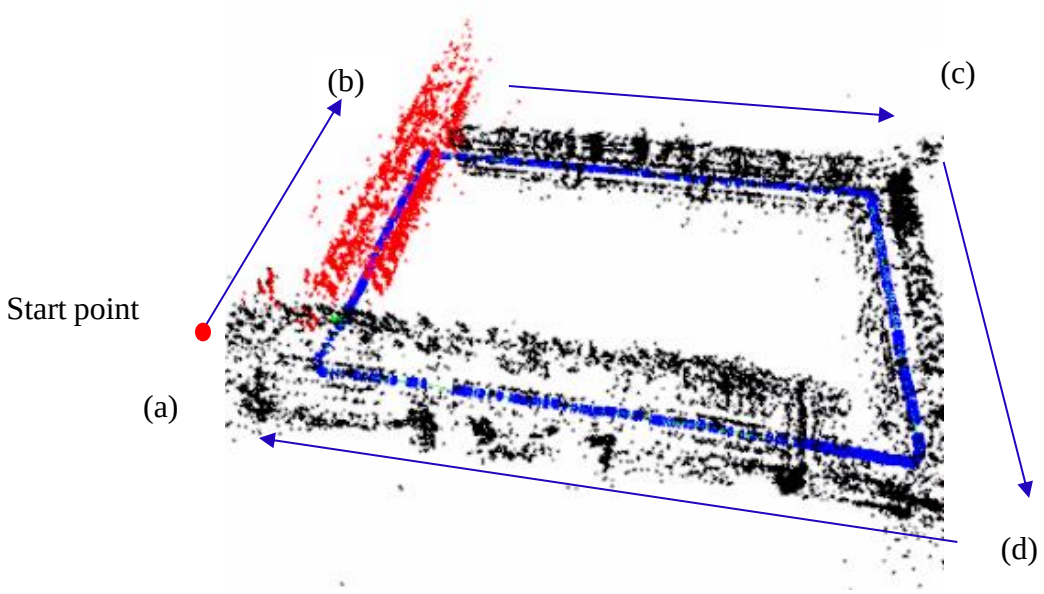


Figure 4.21. Map points and trajectory taken from the static environment. The red points are the estimate map points which can be detected at the camera's current location while black points are optimized map points. Blue squares represent the locations and pose of the keyframes which records the camera trajectory. The view of four corners (a),(b),(c) and (d) in Fig 4.20.

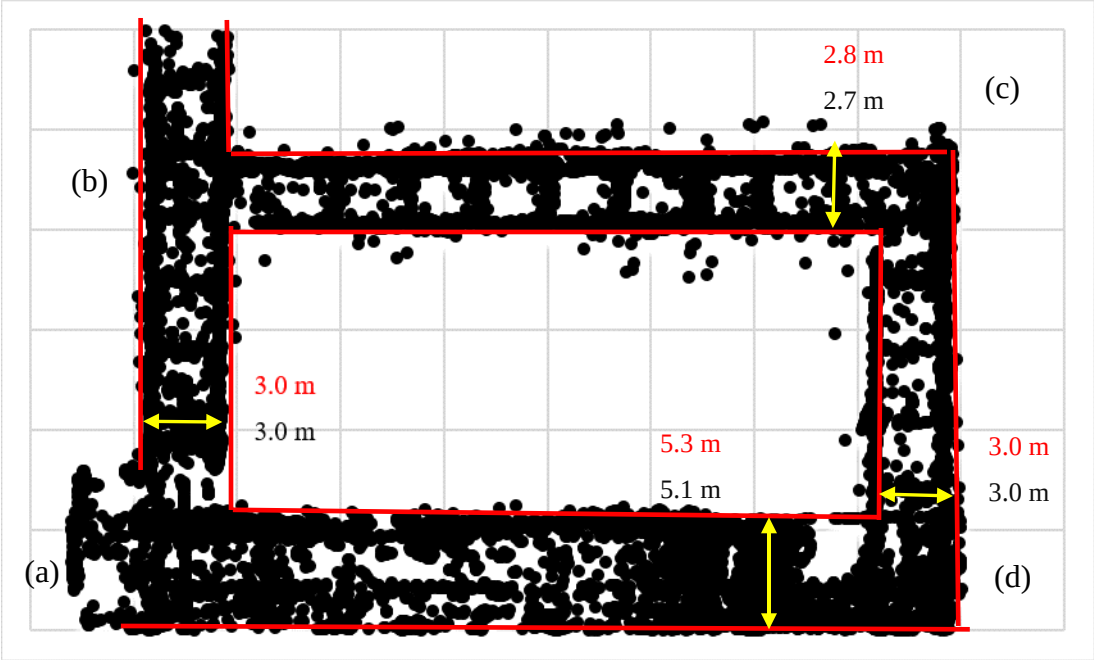


Figure 4.22. 2D CLO corridor static map points after filtering from RGB-D camera. The red line reveals the walls in this environment and the yellow line is the width of the corridors. All the red indicators are calculated length based on the map points while the black indicators are the measurement from real world. The error of map to the real world is around 3%-4%.

In part two, the experiment is performed with people passing by. The robot runs along the trajectory in the last experiment. During the experiment, the camera does not take any features until the walking people are three meters away from the camera. When they come close and stay for a moment, the camera extracts features from them. These features are recorded as map points in the map. The map of the dynamic environment is shown in Fig 4.23.

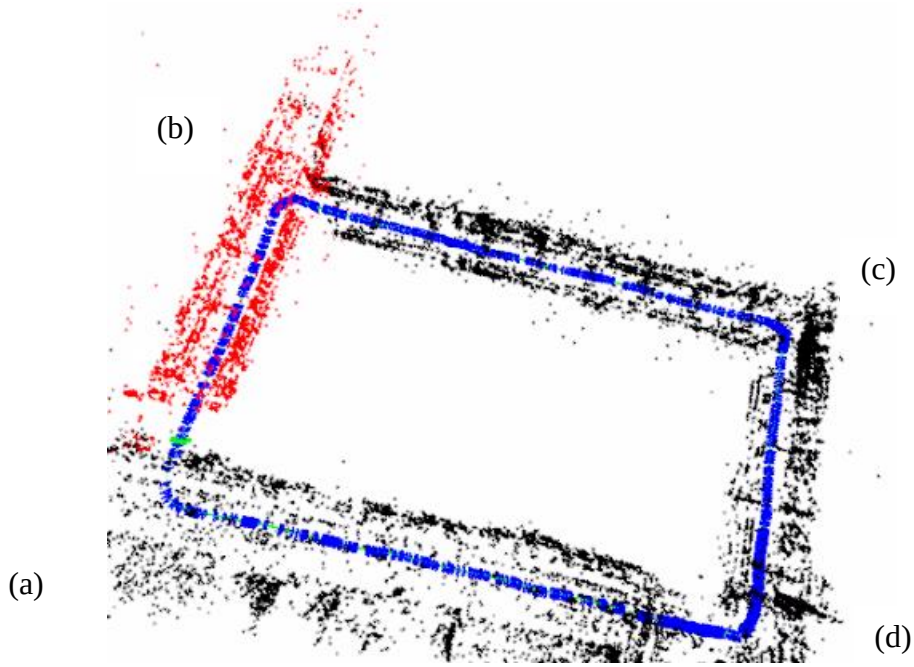


Figure 4.23. Map points and camera trajectory from dynamic environment.

This time, the RGB-D SLAM extracts 1132 keyframes and 19338 map points. The map and trajectory are similar with the results shown in Fig 4.21.

To analysis the performance of the algorithm in dynamic environments, the datasets from two experiments are compared. The static environment map is regard as the ground truth. The trajectory RMSE is calculated to evaluate the algorithm robustness in dynamic environment. In the two datasets, 1018 pairs of keyframes are involved in the calculation. The RMSE of the trajectory is 0.72m and the minimum error is 0.11m. The detailed result is shown in Table 4.7.

Table 4.7. The dynamic environment mapping performance

	RGB-D
RMSE	0.720m
Mean	0.603m
Standard Deviation	0.393m
Min	0.114m
Max	2.179m
No. Pairs	1018

Considering the scale of the scanned environment, the trajectory RMSE in dynamic environment is close to the RMSE of corridor experiment (experiment 2). That means, even in the dynamic environment, the ORB-SLAM is robust enough to provide the locations and the map. Its performance is consistent in static and dynamic environments. This experiment shows the ORB-SLAM has the potential to navigate in dynamic environments. It could be used for outdoor autopilot.

5. CONCLUSIONS AND FUTURE WORK

Comparing all the stereo and monocular SLAM results, monocular has the better performance for indoor environment under our platform with limited computational power. The ORB-SLAM system runs based on feature points. It needs to calculate the ORB feature for each image, so it is computing and time consuming. At the same time, ORB-SLAM's three-thread structure also brings a heavy burden on the CPU, which leads to poor performance on embedded devices with lower computing power such as the robot used in the project. The Jackal mobile robot carries a 2.4GHz Celeron J1800 which has great computation capability gap compare to desktop-level processor. As discussed in Chapter 4, the image processing runs on the mobile robot processor and the ORB-SLAM algorithm runs on the laptop. Comparing with monocular SLAM, the burden of the CPU is doubled when the stereo SLAM is performed. So, upgrading the processor is expected to receive a better VSLAM experience and result for the stereo camera.

Based on the experiments, the ORB-SLAM performs well using RGB-D camera in different indoor environments. It could provide robust and accurate map for autonomous navigation and autopilot. However, once the ORB-SLAM loses tracking due to various reasons, the camera has to return and retrieve the tracking by relocalization. The combination of Inertial Measurement Unit (IMU) and ORB-SLAM can restore the camera's pose when the camera fails to track the trajectory. The ORB-SLAM is also limited by the sparse map points. It is hard to directly use the map for navigation. If deep learning fuses with ORB-SLAM and the generated semantic map [36] can identify objects, it will provide new a method for navigation.

APPENDIX A. ZHANG'S CALIBRATION

Appendix A introduces the Zhang's calibration method mathematically. According the introduction of principle of camera imaging in Chapter 2, the transformation between pixel coordinate frame and world coordinate frame can be written as:

$$z_c\begin{bmatrix}u\\v\\1\end{bmatrix}=z_c\begin{bmatrix}\frac{1}{d_x}&0&u_0\\0&\frac{1}{d_y}&v_0\\0&0&1\end{bmatrix}\begin{bmatrix}x\\y\\1\end{bmatrix}=\begin{bmatrix}\frac{1}{d_x}&0&u_0\\0&\frac{1}{d_y}&v_0\\0&0&1\end{bmatrix}\begin{bmatrix}f&0&0&0\\0&f&0&0\\0&0&1&0\end{bmatrix}\begin{bmatrix}x_c\\y_c\\z_c\\1\end{bmatrix} \tag{A.1}$$

$$=\begin{bmatrix}\frac{1}{d_x}&0&u_0\\0&\frac{1}{d_y}&v_0\\0&0&1\end{bmatrix}\begin{bmatrix}f&0&0&0\\0&f&0&0\\0&0&1&0\end{bmatrix}\begin{bmatrix}r_{00}&r_{01}&r_{02}&T_X\\r_{10}&r_{11}&r_{12}&T_Y\\r_{20}&r_{21}&r_{22}&T_2\\0&0&0&1\end{bmatrix}\begin{bmatrix}x_w\\y_w\\z_w\\1\end{bmatrix}$$

In order to be consistent with Zhang's paper, the formula symbols are changed as below:

$$z_c\begin{bmatrix}u\\v\\1\end{bmatrix}=\begin{bmatrix}\frac{1}{d_x}&0&u_0\\0&\frac{1}{d_y}&v_0\\0&0&1\end{bmatrix}\begin{bmatrix}f&0&0&0\\0&f&0&0\\0&0&1&0\end{bmatrix}\begin{bmatrix}r_{00}&r_{01}&r_{02}&T_X\\r_{10}&r_{11}&r_{12}&T_Y\\r_{20}&r_{21}&r_{22}&T_2\\0&0&0&1\end{bmatrix}\begin{bmatrix}x_w\\y_w\\z_w\\1\end{bmatrix}=A\cdot\begin{bmatrix}R_{3\times3}&t_{3\times1}\end{bmatrix}\cdot\begin{bmatrix}X\\Y\\Z\\1\end{bmatrix} \tag{A.2}$$

$$m=[u,v,1]^T \tag{A.3}$$

$$M=[X,Y,Z,1]^T \tag{A.4}$$

$$sm=A[R\quad t]M \tag{A.5}$$

$$R_{3\times3}=\begin{bmatrix}r_1&r_2&r_3\end{bmatrix} \tag{A.6}$$

$$A=\begin{bmatrix}\alpha & \gamma & u_0\\ 0 & \beta & v_0\\ 0 & 0 & 1\end{bmatrix} \tag{A.7}$$

Calculating external parameters

Let $sm = HM$ with $H = A\begin{bmatrix} r_1 & r_2 & t\end{bmatrix}$, then

$$z_c\begin{bmatrix}u\\ v\\ 1\end{bmatrix} = H\cdot\begin{bmatrix}X\\ Y\\ 1\end{bmatrix} \tag{A.8}$$

$$H=\begin{bmatrix}h_1 & h_2 & h_3\end{bmatrix} \tag{A.9}$$

Let $[h_1 h_2 h_3] = \lambda A[r_1 r_2 t]$.

According to the properties of orthogonal matrices, it is easy to get:

$$\begin{aligned} &h_1^T A^{-T} A^{-1} h_2 = 0\\ &h_1^T A^{-T} A^{-1} h_1 = h_2^T A^{-T} A^{-1} h_2 \end{aligned} \tag{A.10}$$

Then, the following can be calculated:

$$r_1 = \lambda A^{-1} h_1 \quad r_2 = \lambda A^{-1} h_2 \quad r_3 = r_1 \times r_2 \quad t = \lambda A^{-1} h_3 \tag{A.11}$$

$$\lambda = \frac{1}{\left\| A^{-1} h_1 \right\|} = \frac{1}{\left\| A^{-1} h_2 \right\|} \tag{A.12}$$

H has 8 unknowns, at least 8 equations are required. Each pair of points can provide two equations, so at least four pairs of points are needed to solve the H.

Calculating internal parameters

As r1 and r2 are orthogonal, and $\|r_1\| = \|r_2\| = 1$,the following constraints can be obtained:

$$\begin{cases} h_1^T A^{-T} A^{-1} h_2 = 0\\ h_1^T A^{-T} A^{-1} h_1 = h_2^T A^{-T} A^{-1} h_2 = 1 \end{cases} \tag{A.13}$$

Define:

$$B = A^{-T}A^{-1} = \begin{bmatrix} B_{11} & B_{12} & B_{13} \\ B_{12} & B_{22} & B_{23} \\ B_{13} & B_{23} & B_{33} \end{bmatrix} \tag{A.14}$$

$$= \begin{bmatrix} \frac{1}{\alpha^2} & -\frac{\gamma}{\alpha^2\beta} & \frac{v_0\gamma - u_0\beta}{\alpha^2\beta} \\ -\frac{\gamma}{\alpha^2\beta} & \frac{\gamma^2}{\alpha^2\beta^2} + \frac{1}{\beta^2} & -\frac{\gamma(v_0\gamma - u_0\beta)}{\alpha^2\beta^2} - \frac{v_0^2}{\beta^2} \\ \frac{v_0\gamma - u_0\beta}{\alpha^2\beta} & -\frac{\gamma(v_0\gamma - u_0\beta)}{\alpha^2\beta^2} - \frac{v_0^2}{\beta^2} & \frac{(v_0\gamma - u_0\beta)^2}{\alpha^2\beta^2} + \frac{v_0^2}{\beta^2} + 1 \end{bmatrix}$$

It can be seen that matrix B is a symmetric matrix with only six valid elements, so make a 6-dimensional vector *b*:

$$b = [B_{11}, B_{12}, B_{22}, B_{13}, B_{23}, B_{33}]^T$$

B is composed of A matrix. Therefore, it meets the following equations:

$$h_i^T B h_j = v_{ij}^T b \tag{A.15}$$

$$\mathbf{v}_{ij} = [h_{i1}h_{j1},\ h_{i1}h_{j2} + h_{i2}h_{j1},\ h_{i2}h_{j2},\ h_{i3}h_{j1} + h_{i1}h_{j3},\ h_{i3}h_{j2} + h_{i2}h_{j3},\ h_{i3}h_{j3}]^T \tag{A.16}$$

Then, the following formula can be derived:

$$\begin{bmatrix} v_{12}^T \\ (v_{11} - v_{22})^T \end{bmatrix} b = 0 \tag{A.17}$$

B has 6 unknowns, and the three images can get the unique solution of b. Then, the camera intrinsic reference matrix A can be obtained by using cholesky decomposition to matrix B.

Radial distortion estimation

The Zhang's calibration method only focuses on the most influential radial distortion. Then the mathematical expressions are:

$$\begin{aligned} x &= x + x\left[k_1\left(x^2 + y^2\right) + k_2\left(x^2 + y^2\right)^2\right] \\ y &= y + y\left[k_1\left(x^2 + y^2\right) + k_2\left(x^2 + y^2\right)^2\right] \end{aligned} \tag{A.18}$$

Where (x, y) is the ideal undistorted image coordinate. $\left(x, y\right)$ is the image coordinate after the actual distortion. k_1 and k_2 are the distortion parameters of the first two orders. Since the center of the radial distortion and the center of the camera are in the same position, the following equations can be obtained:

$$\begin{aligned} \mu &= \mu_0 + \alpha \hat{x} + \gamma \hat{y} \\ \hat{v} &= v_0 + \beta \hat{y} \end{aligned} \tag{A.19}$$

Assume that $\gamma = 0$, the following equations can be obtained:

$$\begin{aligned} u &= u + \left(u - u_0\right)\left[k_1\left(x^2 + y^2\right) + k_2\left(x^2 + y^2\right)^2\right] \\ \hat{v} &= v + \left(v - v_0\right)\left[k_1\left(x^2 + y^2\right) + k_2\left(x^2 + y^2\right)^2\right] \end{aligned} \tag{A.20}$$

Where (u, v) is the ideal undistorted pixel coordinate, and $\left(u, \hat{v}\right)$ is the pixel coordinate after the actual distortion. $\left(u_0, v_0\right)$ represents the principal point.

Then transfer the above equations into matrix:

$$\begin{bmatrix} \left(u - u_0\right)\left(x^2 + y^2\right) & \left(u - u_0\right)\left(x^2 + y^2\right)^2 \\ \left(v - v_0\right)\left(x^2 + y^2\right) & \left(v - v_0\right)\left(x^2 + y^2\right)^2 \end{bmatrix} \begin{bmatrix} k_1 \\ k_2 \end{bmatrix} = \begin{bmatrix} u - u \\ \hat{v} - v \end{bmatrix} \tag{A.21}$$

It can be simplified as: $Dk = d$

So, distortion parameter k can be calculated as:

$$k = \left[k_1 k_2\right]^T = \left(D^T D\right)^{-1} D^T d \tag{A.22}$$

APPENDIX B. JACKAL AND BUMBLEBEE TECHNICAL SPECIFICATIONS

Jackal specification

Dimensions (L x W x H)	508 x 430 x 250 mm
Internal storage dimensions	250 x 100 x 85 mm
Weight	17 kg
Payload	10 kg (all terrain), 20 kg (maximum)
Maximum Speed	2.0 m/s
Clearance	65 mm
Operating time	Heavy usage: 2 hours, Basic usage: 8 hours
Drive power	500w
Battery	270 Watt-hours Lithium ion
Encoders	78,000 pulses/m Quadature
Communication	Ethernet, USB 3.0, RS232, IEEE 1384
Control modes	Open-loop, Wheel velocity, Kinematic commands
Feedback	Battery + Motor, Current wheel velocity, Integrated GPS, gyroscope and accelerometer
Computer	Celeron J1800 dual core, 2.4GHz 2GB RAM, WIFI adapter
IP rating	62

Bumblebee2-03S2

Image sensor type	ICX424 (648 x 488 max pixels) 7.4µm square pixels
Baseline	12 cm
Focal lengths	2.5mm with 97° HFOV
Aperture	f/2.0
A/D converter	12-bit analog-to-digital converter
White balance	Automatic / Manual (Color model)
Frame rate	48 FPS
Interfaces	6-pin IEEE-1394a for camera control and video data transmission
Voltage requirements	8-30V via IEEE-1394 interface or GPIO connector
Power consumption	2.5W at 12V
Shutter	Automatic/Manual, 0.01ms to 66.63ms at 15 FPS
Trigger modes	DCAM v1.31 Trigger Modes 0, 1, 3, and 14
Signal to noise ratio	60dB
Dimensions	157 x 36 x 47.4 mm
Mass	342 g

Kinect V1

Depth sensor type	Structured light
RGB camera resolution	640 x 480 pixels
Infrared camera resolution	320 x 240 pixels
IR spectrum	~827 – 850 mm
Field of view of RGB image	62° x 48.6°
Field of view of depth image	0.8 m - 4 m
Operative measuring range	20 joints
Frame rate	2

APPENDIX C. TRAJECTORY RMSE CALCULATION

The trajectory is saved in the format of T_c^w which means each row is written as [t, tx, ty, tz, qx, qy, qz, qw]. t is the timestamp and tx, ty, tz represent the translation between two timestamps. qx qy, qz and qw are the rotation between two timestamps in quaternion. The translation and rotation are based on the world coordinate system. If the two datasets T_a and T_b are obtained and timestamps correspond to each other, the error for any pose e_i can be expressed as:

$$e_i = \left\| \log\left(T_{a_i}^{-1} T_{b_i}\right)^{\nu} \right\|_2 \tag{C.1}$$

Then the RMSE of the two trajectories can be defined as:

$$RMSE(a, b) = \sqrt{\frac{1}{n} \sum_{i=1}^{n} e_i^2} \tag{C.2}$$

REFERENCES

[1] Smith, R., M. Self, and P. Cheeseman. “Estimating Uncertain Spatial Relationships in Robotics.” *1987 IEEE International Conference on Robotics and Automation*.

[2] Davison, Andrew J., Ian D. Reid, Nicholas D. Molton, and Olivier Stasse. “MonoSLAM: Real-Time Single Camera SLAM.” *IEEE Transactions on Pattern Analysis and Machine Intelligence* 2007, No. 6.

[3] G. Klein and D. Murray, “Parallel tracking and mapping for small ARworkspaces.” *IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR)*, Nara, Japan, November 2007, pp. 225–234.

[4] Newcombe, Richard A., Steven J. Lovegrove, and Andrew J. Davison. “DTAM: Dense Tracking and Mapping in Real-Time.” *International Conference on Computer Vision*, 2011.

[5] Forster, C., Pizzoli, M. and Scaramuzza, D. (2014). SVO: Fast semi-direct monocular visual odometry. *IEEE International Conference on Robotics and Automation (ICRA)*, 2014

[6] J. Engel, T. Sch¨ops, and D. Cremers, “LSD-SLAM: Large-scale directmonocular SLAM.” *European Conference on Computer Vision (ECCV)*, Zurich, Switzerland, September 2014, pp. 834–849.

[7] Raúl Mur-Artal, J. M. M. Montiel and Juan D. Tardós, “ORB-SLAM: A Versatile and Accurate Monocular SLAM System.” *IEEE Transactions on Robotics*, October 2015, Vol. 31, No. 5, pp. 1147-1163.

[8] Ruymgaart, Peter A. and Tsu T. Soong. “The Kalman-Bucy Filter.” *Mathematics of Kalman-Bucy Filtering Springer Series in Information Sciences*, 1988, pp. 100–153.

[9] K. P. Murphy, “Bayesian Map Learning in Dynamic Environments,” *Advances in Neural Information Processing Systems*, Denver, 1999, pp. 1015-1021.

[10] Edward Rosten, Tom Drummond, “Machine Learning for High-Speed Corner Detection.” *European Conference on Computer Vision (ECCV) 2006*, pp. 430-443

[11] Danping Zou and Ping Tan, “Coslam: Collaborative visual slam in dynamic environments.” *IEEE transactions on pattern analysis and machine intelligence*, 2013

[12] Zhengyou Zhang, “A Flexible New Technique for Camera Calibration.” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, December 2000, Vol 22: pp. 1330-1334

[13] Raúl Mur-Artal and Juan D. Tardós. ORB-SLAM2, “an Open-Source SLAM System for Monocular, Stereo and RGB-D Cameras.” *IEEE Transactions on Robotics*, Vol. 33, No. 5, pp. 1255-1262, 2017.

[14] Edward Rosten, Reid Porter and Tom Drummond, "FASTER and better: A machine learning approach to corner detection". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2010, pp 105–119.

[15] Lowe, David G. “Object recognition from local scale-invariant features”. Proceedings of the *International Conference on Computer Vision*, 1999, pp 1150-1157.

[16] Herbert Bay, Andreas Ess, Tinne Tuytelaars, Luc Van Gool "SURF: Speeded Up Robust Features", *Computer Vision and Image Understandin*g (CVIU), Vol. 110, No. 3, pp. 346–359, 2008.

[17] Michael Calonder, Vincent Lepetit, Christoph Strecha and Pascal Fua, “BRIEF: Binary Robust Independent Elementary Features.” *European Conference on Computer Vision (ECCV) 2010*, pp 778-792.

[18] “Gaussian Blur.” *Wikipedia*, Wikimedia Foundation, 25 Nov 2019, https://en.wikipedia.org/wiki/Gaussian_filter.

[19] Hamming, R. W. “Error Detecting and Error Correcting Codes.” *Bell System Technical Journal*, Vol. 29, No. 2, 1950, pp. 147–160.

[20] M. K. Hu, "Visual Pattern Recognition by Moment Invariants", *IRE Trans. Info. Theory*, Vol. IT-8, 1962, pp.179–187.

[21] Hartley, Richard, and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. 2004, doi:10.1017/cbo9780511811685.

[22] “Eight-Point Algorithm.” *Wikipedia*, Wikimedia Foundation, 21 Nov 2019, https://en.wikipedia.org/wiki/Eight-point_algorithm.

[23] “Singular Value Decomposition.” *Wikipedia*, Wikimedia Foundation, 9 Oct 2019, https://en.wikipedia.org/wiki/Singular_value_decomposition.

[24] “Camera Calibration and 3D Reconstruction”. *OpenCV*, 6 Nov 2019, https://docs.opencv.org/2.4/modules/calib3d/doc/camera_calibration_and_3d_reconstruction.html?highlight=ransac.

[25] ″Direct Linear Transformation.” *Wikipedia*, Wikimedia Foundation, 2 Nov. 2019, https://en.wikipedia.org/wiki/Direct_linear_transformation.

[26] P.J. Besl and Neil D. McKay. “A method for registration of 3-D shapes.” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, February 1992, Vol 14: pp. 239 – 256

[27] K. S. Arun, T. S. Huang and S. D. Blostein, “Least-Squares Fitting of Two 3-D Point Sets.” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, September 1987, Vol PAMI-9, Issue: 5: pp. 698 – 700.

[28] “ORB (Oriented FAST and Rotated BRIEF).” *OpenCV*, 9 Nov, 2019, https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_feature2d/py_orb/py_orb.html.

[29] Lepetit, Vincent, et al, “EPnP: An Accurate O(n) Solution to the PnP Problem.” *International Journal of Computer Vision*, Vol. 81, No. 2, 2008, pp. 155–166.

[30] Galvez-Lopez, Dorian, and Juan D. Tardos. “Real-Time Loop Detection with Bags of Binary Words.” *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2011.

[31] Jutta Willamowski, Damian Arregui, Gabriella Csurka, Christopher R. Dance, Lixin Fan, "Categorizing nine visual classes using local appearance descriptors." *Icpr Workshop on Learning for Adaptable Visual Systems (2004)*.

[32] Bill Triggs, Philip F. McLauchlan, Richard I. Hartley, and Andrew W. Fitzgibbon, “Bundle Adjustment — A Modern Synthesis” *Vision Algorithms: Theory and Practice Lecture Notes in Computer Science*, 2000, pp. 298–372.

[33] Horn, Berthold K. P, “Closed-Form Solution of Absolute Orientation Using Unit Quaternions.” *Journal of the Optical Society of America A*, Jan. 1987, Vol. 4, No. 4, p. 629.,

[34] Quigley, Morgan; Berger, Eric; Ng, Andrew Y. “STAIR: Hardware and Software Architecture “, *AAAI 2007 Robotics Workshop*.

[35] ″Root-Mean-Square Deviation.” *Wikipedia*, Wikimedia Foundation, 12 Nov. 2019, https://en.wikipedia.org/wiki/Root-mean-square_deviation.

[36] Nüchter, Andreas, and Joachim Hertzberg. “Towards Semantic Maps for Mobile Robots.” *Robotics and Autonomous Systems 56 (2008)*, No. 11.