# Autonomous Aerial Robot Using Dual-fisheye System

**by**

**Wenliang GAO**

A Thesis Submitted to
The Hong Kong University of Science and Technology
in Partial Fulfillment of the Requirements for
the Degree of Master of Philosophy
in the Department of Electronic and Computer Engineering

August 2018, Hong Kong

# Authorization

Wenliang GAO

2 August 2018

# Autonomous Aerial Robot Using Dual-fisheye System

**by**

**Wenliang GAO**

This is to certify that I have examined the above MPhil thesis

and have found that it is complete and satisfactory in all respects,

and that any and all revisions required by

the thesis examination committee have been made.

PROF. SHAOJIE SHEN, THESIS SUPERVISOR

PROF. BERTRAM SHI, HEAD OF DEPARTMENT

Department of Electronic and Computer Engineering
2 August 2018

# ACKNOWLEDGMENTS

I really appreciate spending two years in HKUST. It would be part of my meaning life.

I would never have completed this work without the help from many people. First of all, I thank my supervisor, Professor Shaojie Shen, for his years of mentoring, advice, and encouragement. I have learned how to develop, evaluate, express, and defend my ideas from him. They are important for my past and later study, research and working in developing.

I thank my colleagues in UAV group. Tianbo Liu, Yi Lin, and Yonggen Ling, we talk about the technology development and share a lot of ideas. Fei Gao, Kaixuan Wang, Kejie Qiu, Peiliang Li, Tong Qin, and William Wu, we work together and help to build up my knowleage system. Kaixuan Wang, Peiliang Li, we arrive at UST at the same time and have finished several deadlines. In daily life, we have been good friends. Without them, my graduate study and research at HKUST would not be so colorful.

I thank the members of my thesis committee, Professor Michael Yu WANG and Professor Fu ZHANG, for their insightful comments on improving this work.

And I thank the DJI Technology Co. Ltd., which fund my research and study during my graduate study.

Last but not least, I thank my parents, for their support, encouragement, and understanding.

# TABLE OF CONTENTS

# LIST OF FIGURES

# LIST OF TABLES

# Autonomous Aerial Robot Using Dual-fisheye System

**by**

**Wenliang GAO**

Department of Electronic and Computer Engineering

The Hong Kong University of Science and Technology

## ABSTRACT

Safety is undoubtedly the most important requirement in moving robots applications, especially for micro aerial vehicles (MAVs). And both academia and industry have been working hard to equip drones with GPS-free self-localization, obstacle sensing, and autonomous navigation capabilities without prior information about the environments.

The fisheye cameras with ultra-wide filed-of-view (FOV) can provide more spherical coverage of the surrounding environment. A dual-fisheye omnidirectional visual-inertial navigation system (VINS) combine two fisheye cameras and an inertial measurement unit (IMU), cover the whole surroundings of the MAV. It is the minimum sensor suite lending the omnidirectional perception with lightweight and small footprint.

In this paper, we show that it is possible to achieve reliable online autonomous navigation with omnidirectional perception using dual-fisheye VINS. Our system is a quadrotor equipped with two ultra-wide FOV fisheye cameras, which are rigidly mounted on two sides of a rod and facing opposite directions, a low-cost IMU, and heterogeneous onboard computing resources. The two fisheye cameras provide stereo observations with full 360-degree FOV in the horizontal direction and 60-degree FOV in the vertical direction, also provide full spherical monocular coverage of the surrounding situational awareness. Combine a highly accurate optimization-based dual-fisheye visual-inertial state estimator with online initialization and self-extrinsic calibration, three-dimensional map of the environments can be built. And an online trajectory planner that operates guarantees the safe navigation through cluttered environments in any direction. We provide experimental results to validate individual system modules as well as the overall performance in both indoor and outdoor environments.

# CHAPTER 1

# INTRODUCTION

The micro aerial vehicles (MAVs) are widely used in last decade. As ideal platforms for numerous applications in indoor and outdoor environments, with their agile mobility, small size and the cost is low. Some of them are capable of autonomous navigation. There is no doubt that one of the most important requirements for autonomous navigation is safety.

The most important requirements are range sensing, obstacle detection, and avoidance for autonomous drones. Active range sensors such as LiDARs and radars are widely used for mobile robots navigation around obstacles. However, active range sensors are unsuitable for small-scale aerial robots, which have very tight cost and weight constraints, unlike some larger platforms such as land robots. In these cases, vision-based approaches are one of the viable options due to their excellent size, weight, and power (SWaP) characteristics, especially monocular vision-based approaches. Monocular vision-inertial systems using only one camera and an inertial measurement unit (IMU) is the smallest setup satisfy the SWaP.

The dynamics of aerial robots such as quadrotor support the motion in multiple directions. An aerial robot such as a quadrotor is able to However, the movements of autonomous aerial robots are limited because of the lack of perception of multiple directions. The stereo vision systems or the monocular vision systems installed on existing aerial robot system is often insufficient to guarantee the required perception ability because of the narrow field of view. To this end, much emphasis has to be placed on the development of omnidirectional stereo vision systems for aerial robots using different camera configurations.

To install more cameras on the aerial robots and construct multiple stereo systems is the direct approach of the omnidirectional vision system, but this not only increases cost but also results in tight constraints for structural design, also lead to huge computation requirement. There exists extensive work focusing on dealing with multiple stereos, some of them achieved with advanced FPGA [78, 25]. Using catadioptric cameras is economic due to their panoramic view. Some omnidirectional vision systems are constructed with

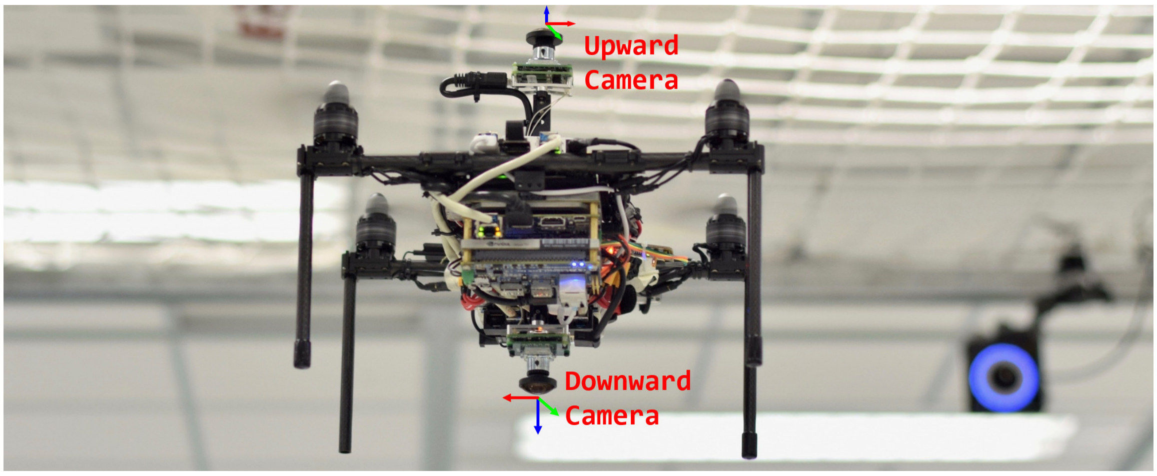


Figure 1.1: A closeup of our aerial robot platform, the two cameras are rigidly mounted and face opposite directions.

two convex reflective mirrors [24], with one camera and one convex reflective mirror [43], or with plane mirrors [77]. However, the catadioptric cameras are not suitable for aerial robots due to the size and the installation requirements.

For aerial robots, the omnidirectional vision systems need to be assembled with the fewest number of components but achieve the best coverage.

In this work, we show the perception method which is able to provide omnidirectional stereo vision formed by two ultra-wide (245-degree) field-of-view (FOV) fisheye cameras installed facing opposite directions on a rigid rod. The configuration is developed and installed on an aerial robot, shown in Fig. 1.1. And the sensing scope is shown in Fig. 1.2, the hole 360-360-degree region are covered with the system. The blue area is the overlapping of dual-fisheye stereo, the yellow area is the monocular vision coverage, the red area is the blind spot and the gray area is the ground.

In this paper, we present a complete system solution to address all aforementioned challenges. The system-level contribution is the proof of the feasibility of using the minimum dual-fisheye omnidirectional VINS and fully onboard processing to achieve safe navigation through unknown cluttered environments. The decisions on each system module, from hardware to software, are results from careful engineering considerations and trade-offs. We choose to use a dual-fisheye camera system with IMU-triggered hardware synchronization to provide wide-angle observations with precise time stamping. We use a set of powerful heterogeneous onboard computing platforms for parallelizing the computationally demanding per-pixel image operations.

At the level of software modules, our contributions are embedded into the choice of

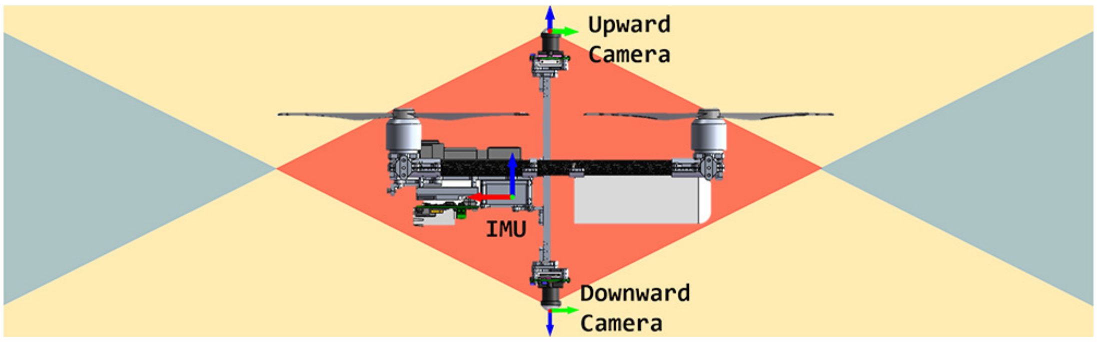


Figure 1.2: The 2D illustration of the sensing area.

suitable real-time algorithms that are either developed specifically for this work, or are improved from our previous works. The backbone of our algorithm package is a high-accuracy, tightly-coupled, optimization-based dual-fisheye omnidirectional visual-inertial state estimator with fisheye camera support, online initialization, and camera-IMU extrinsic calibration. A plane-sweeping-based multi-view depth estimation module, in combination with semi-global (SGM) depth smoothing, provides real-time dense depth images for obstacle detection. The depth images are fused into a global 3D map using a probabilistic truncated signed distance function (TSDF) fusion. Finally, a gradient-based trajectory planning module, which directly operates on the reconstructed 3D map, provides collision-free trajectories through cluttered 3D environments. The perception-action loop is closed by executing the desired trajectory using a standard multi-rotor controller. To the best of our knowledge, we are the first to achieve such navigation capability using this minimum sensor suite. This work points out an direction towards autonomy for very small (less than 10 centimeters in diameter) aerial vehicles.

The rest of this paper divides into eight parts. System-level related works are briefly discussed in Sect. 2. Sect. 3 gives an overview of our system in both hardware and software architectures. Sect. 4 and Sect. 5 discusses how we use the fisheye camera. State estimation and dense 3D mapping modules are presented in Sect. 6 and Sect. 7, respectively. Online experimental results (Sect. 8) verify the performance of individual modules, as well as the integrated system. We show autonomous navigation in both indoor and outdoor environments. Sect. 11 concludes this work and points out possible directions for future improvements, where we aim to extend our framework towards systems with higher speed, lower power consumption, smaller size, and lower cost.

# CHAPTER 2

# RELATED WORK

There are numerous scholarly works focus on autonomous movements in GNSS-denied environments based on vision system. Some of them premeditate the omnidirectional perception sensing. In this section, we review some omnidirectional sensing systems and some relevant system-level autonomous navigation works using visual sensing modalities.

Vision-based methods autonomous navigation have been developed in last decade. In our previous work [44], we have investigated the probability of the minimum sensing setup for autonomous perception, localization, navigation, and obstetrical avoidance. Which is a monocular visual-inertial system only combine with one camera and one IMU. Using a monocular visual-inertial navigation system and a GPU-accelerated monocular mapping approach, the environments can be constructed and the three-dimensional map satisfy autonomous motion planning. Similar works are introduced in [13, 18]. In [67], the authors implement the MAV with three onboard computing units: an Intel Core2Duo for visual odometry, mapping, and planning, an FPGA for real-time stereo-matching, and Gumstix for real-time embedded tasks. With a loosely-coupled visual-inertial extended Kalman filter (EKF) state estimation and the Octomap [30], the MAV is able to fly by an obstacle-free path. And in [20], the authors also show a vision-based mapping and planning framework with one front-ward stereo and a down-ward optical flow camera.

In [17], the authors shows an MAV based on the SVO [19] and REMODE [60]. The MAV can fly autonomously and provide real-time dense 3D reconstruction. Due to the narrow sensor, only a downward camera for both state estimation and mapping, the obstacle detection and avoidance is weak in cluttered environments. Most works deploy mapping module on a ground station or off-board [17, 10, 16]. Another direction of research comprise the use of learning-based approaches for end-to-end generation of navigation operations, such as [23, 53]. It is a promising inspiration to reach the level of maturity for safe flight in unknown environments.

However, most of exist works are based on stereo cameras and monocular cameras with narrow FOV so that the sensing scope is limited.

It is a direct idea to equip more sensors to cover a wider perception scope. In [25], the authors installed eight cameras on the aerial robots and construct multiple stereo systems to form an omnidirectional vision system directly. And in [78], the authors developed a product with ten cameras, an IMU and five ultrasonic range finders which construct an omnidirectional visual-inertial system. Both these methods process the data with advanced FPGA due to the huge computation requirement. Such configuration is used in Skydio R1 [1], an commercial autonomous flight MAV equipped with 12 sensing camera construct 6 pairs of stereos facing to six directions, with GPU-accelerated approaches.

An omnidirectional stereo system configured with two specially designed mirrors and one camera was shown in [38] and an improved version is presented in [34] with a focus on size reduction and calibration. The system was installed on a quadrotor micro aerial vehicle in [34] but without a perfect demonstration of flying test. This system is not suitable for deployment on small aerial robots due to its special configuration and the wast of onboard space of an aerial robot. Also, it is hard to get 3D information due to the complexity in calibrating the extrinsic parameters of the specially designed mirrors. Processing regions of the different spatial resolution from upward- and downward-facing mirrors in one image is also a challenging problem, and the narrow vertical FOV limits its usage for safe robotic navigation.

In our work, however, we show how only two cameras is sufficient to achieve the whole sphere space perception and omnidirectional sensing and miniaturize the platform. The system achieve autonomous localization, navigation, mapping and obstetrical avoidance and all of process are onboard.

[1]https://www.skydio.com/

# CHAPTER 3

# SYSTEM OVERVIEW

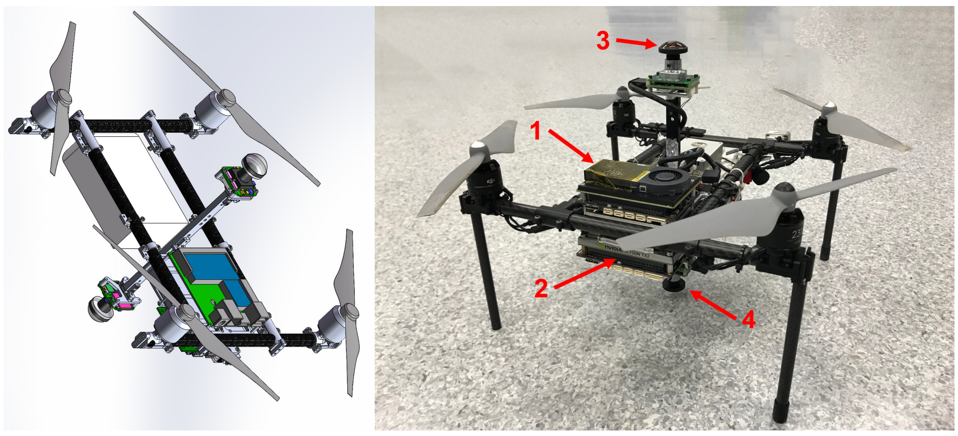


Figure 3.1: A synthetic and closeup of our aerial robot platform.

We now present the hardware design and the software architectures of our quadrotor experimental testbed. We address requirements in sensor selection, sensor synchronization, distribution of computing power, monitoring, and debugging tools, experimental interfaces, and modular system designs. We will explain the rationale behind each of our system design decisions along with the descriptions of system modules.

## 3.1 Hardware Architecture

We aim to design a modular, compact, durable, and easy-to-construct aerial robot testbed. Our airborne system consists two main modules, the Perception Core and the Quadrotor Frame. The perception core integrates onboard computers and sensors, and the quadrotor frame consists of the mechanical construction, motors, propellers, and the battery.

### 3.1.1 Perception Core

The perception core is a detachable module that houses all computing and sensing modalities. It enables independent testing of the state estimation and mapping modules and can be used with other mechanical frames.

We use two computers to form a powerful heterogeneous computing platform. One is a mini computer that is powered by a quad-core Intel i7-8500U processor running up to 4.00 GHz [1]. It is equipped with 8 GB memory and 128 GB SSD, and consumes around 15 watts of power. This is the primary computer for system scheduling and for performing tasks that cannot be parallelized (Fig. 3.2). Another computer onboard is a NVIDIA TX2, which is assembled by a core[2] on a carrier board. The 256 NVIDIA CUDA GPU cores on the TX2 make it particularly suitable for parallel computing of depth images and TSDF fusion. The TX2 is equipped with a six-core ARM Cortex-A57 processor and 8 GB memory and consumes approximately 15 watts of power. The two onboard computers are connected using Ethernet cable. Communication between two computers is done by utilizing the ROS infrastructure[3].

Our onboard computing resources cover most of the widely used development platforms - x86, ARM, and GPU, making it highly flexible for the development of advanced algorithms. It significantly reduces the development cycle since researchers do not need to spend excessive amount time on code-level optimization.

The onboard visual sensors include two PointGrey Chameleon3 Mono USB3 cameras[4] with 240-degree ultra-wide FOV fisheye lens, both with a 1280×1024 resolution. The two cameras are rigidly mounted and face opposite directions. A DJI A3 flight controller[5] is used both as the IMU and attitude stabilization controller. The main elements are shown in Fig. 3.1, they are: (1) Mini i7 computer; (2) NVIDIA TX2; (3) Upward camera; (4) Downward camera. We disable the GPS and magnetometer functionalities in A3 due to their lack of reliability in cluttered indoor environments.

Accurate timestamps are required for high-performance visual-inertial fusion. In our system, this is achieved via hardware synchronization, where individual image capture is triggered by synchronization signals from the DJI A3 flight controller. More specifically, we utilize the triggering model in the Hardware Real-Time Controller of the camera. The DJI Onboard SDK[6] provides an interface for generating a bundle of the pulse signal and the corresponding IMU measurements. Note that that IMU runs at a higher frequency

[1]https://ark.intel.com/products/122589/Intel-Core-i7-8550U-Processor-8M-Cache-up-to-4_00-GHz

[2]https://developer.nvidia.com/embedded/develop/hardware

[3]http://wiki.ros.org/

[4]https://www.ptgrey.com/chameleon3-13-mp-mono-usb3-vision-board-level-on-semi-python-1300

[5]http://www.dji.com/a3

[6]https://developer.dji.com/onboard-sdk/

than the camera, and an IMU measurement will be tagged if it corresponds to an image. Only one additional cable is required to form the hardware synchronization setup.

### 3.1.2 Quadrotor Frame

The quadrotor frame consists of the mechanical construction, motors, propellers, and the power supply module. The frame is built with carbon fiber tubes and aluminum components, making it light but durable. The whole system is shown in Fig. 3.1. The total weight, including the perception core, is 1.8 kg. Tip-to-tip distance is 33 cm. We use the DJI E310[7] motors and the DJI Takyon Z425-M[8] ESCs as the lifting system. We use the intelligent battery of DJI Phantom 4 Pro[9] to provide approximately 15 minutes of flight time. A DJI Lightbridge 2[10] is used for the implementation of the remote desktop functionality on the iPad.

## 3.2 Software Architecture

The software architecture of our system is shown in Fig. 3.2. Utilizing ROS as the communication middleware, we distribute computation loads among the two onboard computers under the principle that dense pixel-wise operations should be performed on the NVIDIA TX1. The two onboard computers are connected using Ethernet.

### 3.2.1 Algorithm Pipeline

On the Mini i7 computer, 400 Hz IMU measurements, as well as the 20 Hz synchronized fisheye images, are fused in the visual-inertial state estimator (Sect 6) to obtain pose, velocity, and attitude for dense mapping, trajectory planning, and feedback control. The estimator output at 400 Hz, which is sufficient for feedback control of agile aerial robots. We use a region-of-interest (ROI) extraction module (Sect. 4.4) to crop the two fisheye images into eight pinhole images to cover the 360-degree horizontal FOV. Pinhole images are then bundled with the estimated pose and sent to the TX2 for depth estimation.

[7]http://www.dji.com/cn/e310

[8]http://www.dji.com/takyon-z425-m-and-z415-m

[9]http://store.dji.com/product/phantom-4-pro-intelligent-battery-high-capacity

[10]http://www.dji.com/cn/lightbridge-2

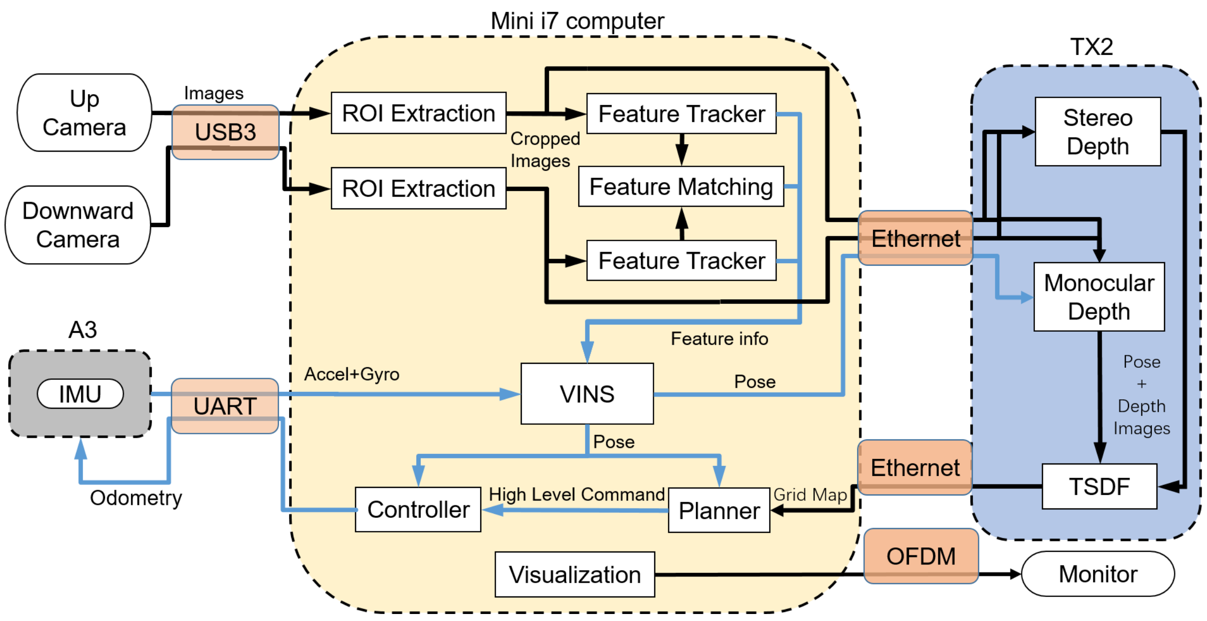


Figure 3.2: System diagram

Utilizing the GPUs onboard the NVIDIA TX2, depth images are computed at 10 Hz using dense semi-global smoothing. Depth images are turned into a global map using GPU-accelerated truncated signed distance function (TSDF) fusion, also running at 10 Hz in (Sect. 7).

The global map is sent back to the Mini i7 computer to assist trajectory planning, in which collision-free time parameterized trajectories that guide the robot towards user-specified goals are generated. The trajectory is executed using the attitude and thrust control interface in the DJI A3 flight controller.

Unlike many existing systems that relies on a downward-facing camera or a frontward-facing camera for state estimation, our system uses two camera to satisfy the omnidirectional sensing needs. We show through online indoor and outdoor experiments (Sect. 8) that our minimal sensing setup is sufficient to achieve complete autonomous navigation.

### 3.2.2 Remote Monitoring and Debugging Interface

Besides the airborne software, we also utilize the DJI Lightbridge 2 to implement an iOS app for remote monitoring and debugging. The Lightbridge directly streams the HDMI output of the Mini i7 computer to an iPad Pro[11]. This enables us to view the desktop in real-time even when the robot is flying. In this way, we are able to directly use the

[11]http://www.apple.com/ipad-pro/

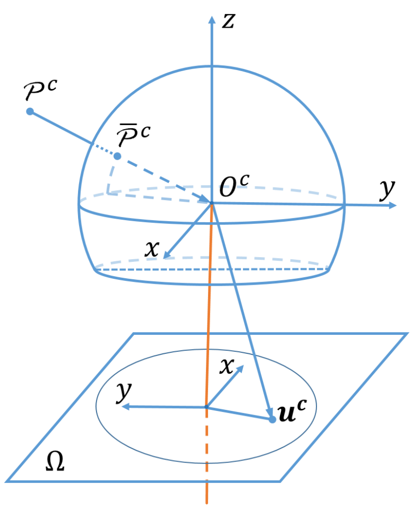
iOS app to run visualization tools such as rviz on the airborne computer. It even enables us to debug and recompile the airborne code while the quadrotor is flying. This setup is independent of any external infrastructure, which makes it very convenience for field tests.

# CHAPTER 4

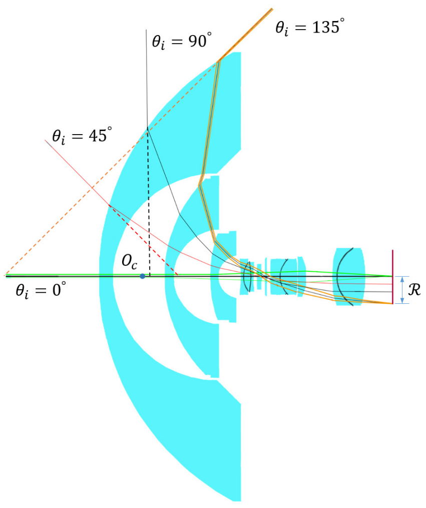
# CAMERA MODEL

For an autonomous aerial robot system, a wide sensing field of the environment observation is able to assist dynamic estimation and mapping. Therefore, we use only two fisheye cameras, which provides sight from all of directions: frontward, backward, upward, downward, leftward and rightward. With an ultra-wide field of view, the visual-inertial navigation system is able to keep tracking features even in high dynamic movement and the mapping system can perform a wide-field mapping. The general camera models are purposed for the ordinary perspective camera with limited FOV. However, a ultra-wide FOV model is required in this system. We proposed a camera model to fit the distortion as a polynomial considering the distortion curve from the manufacturer. Such model is accuracy enough and suitable for the ultra-wide FOV fisheye cameras.

In Sect. 4.2, we review some well-known camera models, then introduce the polynomial-based fisheye camera model which we used. In Sect. 9.1, we simply introduce the calibration. In Sect. 4.4, we show one way to use a fisheye camera: ROI extraction.



(a) Illustration of fisheye camera projection.



(b) Illustration of fisheye lens.

Figure 4.1: Illustration of fisheye camera projection and lens.

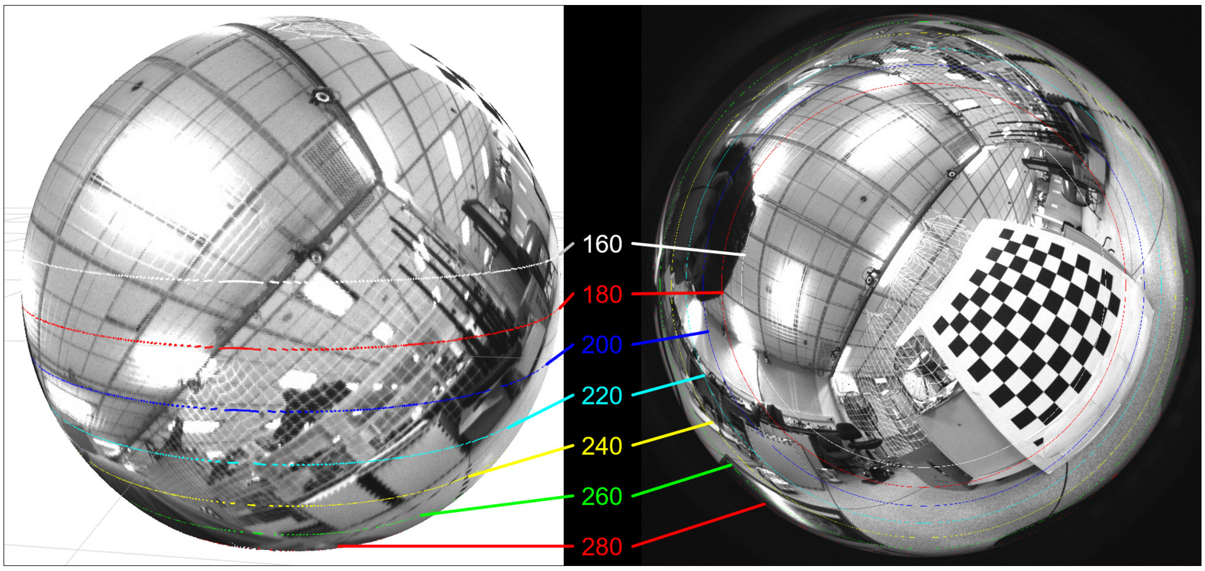


Figure 4.2: The demonstration of the ultra-wide field-of-view (280-degree) image back project to the unit sphere. The FOV range is figure out with circles.

## 4.1 Related Work

There exists extensive scholar work focusing on dealing with distortion, especially radial distortion of cameras. The radial distortion was traditionally modeled as an odd-order polynomial in [7]. But for a fisheye camera, the distortion is so significant that traditional models that are designed for pinhole cameras do not work well. Several models have been proposed to solve the problem, such as the fisheye transform [68], the polynomial fisheye transform [3], the FOV model [12], and the division model [6]. A comparison to these camera models is presented in [32]. [22] pioneers the modeling of catadioptric omnidirectional cameras with a convex mirror. It was further developed into [52], and [66] to process both catadioptric and fisheye cameras. However, all aforementioned camera models are lens-independent. Due to the use of highly distorted pixels for stereo correspondence, an accurate mapping from the raw image to a unit sphere is required. This requires sufficient representation capabilities of the camera model that are unachievable in a lens-independent setting.

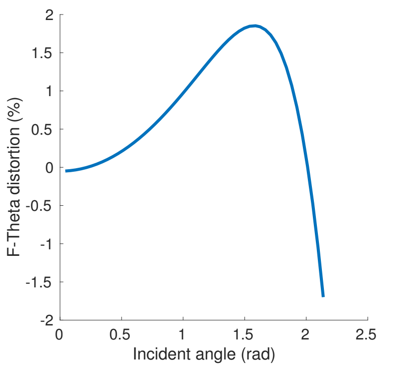
## 4.2 Polynomial-Based Fisheye Camera Model

The camera projection model is a function $\pi : \mathbb{R}^3 \rightarrow \Omega$, which models the relationship between the 3D points and the image domain. The back projection model $\pi^{-1} : \Omega \rightarrow \mathbb{R}^3$ is its inverse process. Fig. 4.1(a) presents the projection course. A 3D point observed by the fisheye camera in the camera frame $\mathcal{P}_c = [x_c, y_c, z_c]^T \in \mathbb{R}^3$ projects to the image plane as $\mathbf{u} = [u, v]^T \in \Omega$; $O_c$ is the origin of the camera frame and $O_cO_{cl}$ is the abstract of the lens. As Fig. 4.1(b) shows, in a particle lens, there are no 3D points $O_c$ and $O_{cl}$ that all the incident and refracted rays go through. The camera frame whose origin is at a virtual point $O_c$ can be built since the error can be ignored with the small lens size. We do not focus on what happens during a ray passing through the lens so we ignore the details of $O_cO_{cl}$. $S$ is a unit sphere with center coinciding with the origin of the camera frame, which also describes the FOV: an 180-degree FOV lens with a half sphere. An incident ray can be formulated as a unit vector $\mathcal{P}_s = [sin\theta cos\varphi, sin\theta sin\varphi, cos\theta]^T$, instant of $\overrightarrow{\mathcal{P}_s\mathcal{O}_c}$, since the reversibility of ray. Where $\varphi$ is the angle between the $x$ positive axis and the ray, and $\theta$ is the angle between ray vector and optical axis vector. $\theta_i$ is the angle of the incident ray $\mathcal{P}_s$, and $\theta_r$ is the angle of the refracted ray $\overrightarrow{O_{cl}\mathbf{u}}$.

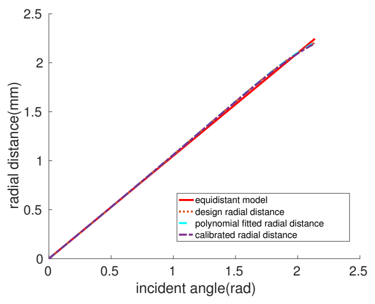
There are many works focus on dealing distortion, especially the radial distortion of cameras. The radial distortion was constantly modeled by an odd-order polynomial in [7], but for a fisheye camera, the distortion is so huge that the general model cannot describe it well. Several models have been put forward to solve the problem, such as the fisheye transform [68], the polynomial fisheye transform [3], the FOV model [12] and the division model [6]. Additionally, some works have compared these camera model methods [32].

A similar problem of catadioptric system due to the similar projection and huge distortion. A different model was proposed [22] to model the projection of an omnidirectional camera with a convex mirror. That model was further developed [52], [66] so that it is able to fit either an omnidirectional camera or fisheye camera. Though, these models are accurate, they are complex or unsuitable for the ultra-wide FOV fisheye cameras. For computer vision, a precise 3D-to-2D mapping table is more suitable as it is model-free and can include all of errors, although it is hard to build. In this paper, we propose a camera model considering the design data for radially symmetric lens with ultra-wide field of view.

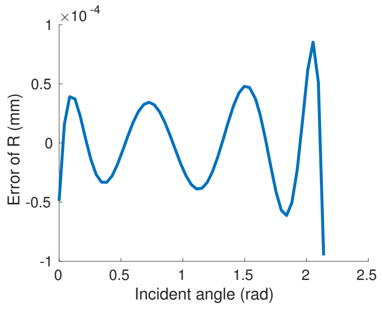
Optically, for any radially symmetric lens system, there are three essential assumptions: (1) any ray that goes through the optical axis will not change its direction; (2) with the increasing of the incident angle $\theta_i$, the refracted angle $\theta_r$ is monotone and continuously increasing; and (3) the refracted rays will be radially symmetric if the related incident rays are radially symmetric, which means for any ray with the same $\theta_i$, its $\theta_r$ is the same. With this three assumptions, the camera model needs to formulate the relationship between the



(a) "F-Theta" distortion curve of a lens with a 245-degree FOV.



(b) Projected radial distance of a lens with a 245-degree FOV.



(c) Radial error of a lens with 245-degree FOV.

Figure 4.3: The distortion of fisheye cameras.

incident angle $\theta_i$ and refracted rays for a radially symmetric lens.

Regard that $\mathcal{R}(\theta) = |\overrightarrow{O_i\mathbf{u}}|$ is the projected radial distance from the principal point on the image plane with the incident angle $\theta$, and $\mathcal{R}_{ref}(\theta)$ is a reference radial distance, which is referenced by a designed model, also formulated with $\theta$. Define $\mathcal{R}_{err}(\theta) = \mathcal{R}(\theta) - \mathcal{R}_{ref}(\theta)$ as the radial error. In an optical system, define distortion as

$$\mathcal{D}(\theta) = \frac{\mathcal{R}_{err}(\theta)}{\mathcal{R}_{ref}(\theta)} \times 100\%. \tag{4.1}$$

If the reference radial distance is defined as

$$\mathcal{R}_{ref}(\theta) = f\theta, \tag{4.2}$$

its distortion is called "F-Theta" distortion [75], as Fig. 4.3(a) shows, and this reference projected radial distance also formulates an equidistant model of the fisheye camera. For an optical system, the ideal "F-Theta" distortion can almost be accurately given by its designers thanks to optical design software products such as ZEMAX[1] or Code-V[2], which are able to perform an accurate ideal ray tracing simulation. The calibration process is to correct the real distortion with the manufactured error reference from the designed distortion. Fig. 4.3(a) shows the "F-Theta" distortion curve of one of our lenses with a 245-degree FOV. Our proposed model describes the distortion curve with a continuous smooth function $\mathcal{D}(\theta)$ of the incident angle $\theta$, such as a polynomial.

For the essential optical assumption, we should fit the relationship between $\theta_i$ and $\theta_r$, but this requires some extra calculation during projection. We fit the distortion curve

[1]http://www.zemax.com/

[2]https://optics.synopsys.com/codev/

and describe the relationship between $\theta_i$ and $\mathcal{R}(\theta)$. In this paper, we use a high-order polynomial to fit the "F-Theta" distortion curve $\mathcal{D}(\theta)$. With (4.1) and (4.2), the real projected radial distance can be formulated as

$$\mathcal{R}(\theta) = [1 + \mathcal{D}(\theta)]\mathcal{R}_{ref}(\theta) = [1 + \mathcal{D}(\theta)]f\theta = f\sum_{i=0}^{N} \eta_i\theta^i. \tag{4.3}$$

In the $N$ order polynomial $\sum_{i=0}^{N} \eta_i\theta^i$, there is $\eta_0 = 0$, $\eta_1 = 1$ from (4.3), so two of the coefficients are fixed. We use a 7 order polynomial to fit the fisheye camera. As Fig. 4.3(a) show lens, the comparison of the projected radial distance is shown in Fig. 4.3(b), and the projected radial distance error between the fitted and real one is small enough as Fig. 4.3(c) show.

As the projection shows in Fig. 4.1(a), there are three main steps for one 3D point in a world frame project to image: 3D point becoming a unit ray vector $\mathcal{P}_c \rightarrow \overrightarrow{\mathcal{P}_s O_c}$, ray refraction $\overrightarrow{\mathcal{P}_s O_c} \rightarrow \overrightarrow{O'_c \mathbf{u}}$ and ray projection $\overrightarrow{O'_c \mathbf{u}} \rightarrow \mathbf{u}$.

(1) A 3D point in the camera frame is transformed to a ray vector and the vector is normalized as

$$\overrightarrow{\mathcal{P}_s O_c} = \frac{\overrightarrow{\mathcal{P}_c O_c}}{||\overrightarrow{\mathcal{P}_c O_c}||}. \tag{4.4}$$

Transform this unit vector to spherical coordinates in the camera frame, and get the incident angle $\theta$ by

$$\begin{aligned} cos(\pi - \theta) &= \overrightarrow{\mathcal{P}_s O_c} \cdot \overrightarrow{O_i O_c} = z_s, \\ tan(-\varphi) &= \frac{y_s}{x_s}, \end{aligned} \tag{4.5}$$

where $\overrightarrow{\mathcal{P}_s O_c} = [x_s, y_s, z_s]^T$, $\overrightarrow{O_i O_c}$ is the optical axis vector. Especially if $x_s = 0$, $\varphi = 0$.

(2) Fit the real projected radial distance $\mathcal{R}_i(\theta)$ up-to-scale of focal length:

$$\mathcal{R}(\theta) = \sum_{i=0}^{N} \eta_i\theta^i. \tag{4.6}$$

(3) With the focal length in the pixel unit $f$, the real projected radial distance in the pixel is $\mathcal{R}(\theta) = f\mathcal{R}_i(\theta)$. Then the image coordinate can be obtained directly:

$$\mathbf{u} = \mathcal{R}(\theta)\begin{bmatrix} cos(\varphi) \\ sin(\varphi) \end{bmatrix} + \begin{bmatrix} c_x \\ c_y \end{bmatrix}, \tag{4.7}$$

where $[c_x, c_y]^T$ is the image point whose incident angle $\theta = 0$. Since the rotation between the lens and the CMOS chip, the focal length is not the same in different directions thus the real projected radial distance will be projected onto a 2D plane and described by an affine transformation. An individual focal length $f$ is replaced by a $2 \times 2$ affine matrix $A_f = \begin{bmatrix} A_{11} & A_{12} \\ 0 & A_{22} \end{bmatrix}$. Above of all, combine (4.4) to (4.7) to get

$$\mathbf{u} = \sum_i \eta_i \theta^i A_f \begin{bmatrix} cos(\varphi) \\ sin(\varphi) \end{bmatrix} + \begin{bmatrix} c_x \\ c_y \end{bmatrix}. \tag{4.8}$$

The function of the camera projection model $\pi(\cdot)$ is (4.5) and (4.8). This formulation is similar to the pinhole model but is suitable for almost all kinds of cameras whose projection is radially symmetrical. The radial distortion is well described in (4.6), and we refuse to model the tangential distortion because, by [35], the tangential distortion of most lenses is quite small due to the improvements of optical manufacture.

In the back projection model $\pi^{-1}(\cdot)$, the high-order polynomial $\sum_i \eta_i \theta^i$ may have several roots, which may be real or complex, and cannot have analytical solutions. the speed of solving the root is slow since the computation of the root requires numerical approaches. For real-time applications, we may use another high-order polynomial fitting or piecewise linearization approach to accelerate the calculation. The back-projected point $\mathcal{P}_s = [sin\theta cos\varphi, sin\theta sin\varphi, cos\theta]^T$ is the coordinates of the unit sphere since, with only one 2D image, the depth scale cannot be solved.

In particular, the polynomial coefficient $\eta_i$ can be initially guessed by fitting the "F-Theta" distortion curve with a polynomial in (4.3) before calibration. During camera calibration, the initial guess of the polynomial coefficients will not be far from the true value.

## 4.3 Photometric Model

In [21], we discussed that one of the main shortcomings of our omnidirectional stereo is that the stereo matching method is challenging due to the unbalanced illumination caused by vignetting of the optical system. Additionally, configured with an upper camera and a downer camera, the omnidirectional stereo earns a vertical epipolar line. Therefore, the illumination balance in the vertical direction in the ROI images is necessary.

The intensity of each pixel is influenced not only by the object but also by the imaging projection system. The imaging projection system of the camera can be divided into two part, the optical lens module, and the electronic sensor module. The image ray passes through the optical system and gets some attenuation, then arrives the sensor system. The sensor will trans the signal from optical irradiance to the electronic signal via the exposure time. The process is illuminated in Fig. 4.4. The influential most of the two system is called:

1. Vignetting: the optical attenuation by the optical system;

2. Radiometric response: the transform of the optical-electronic system;

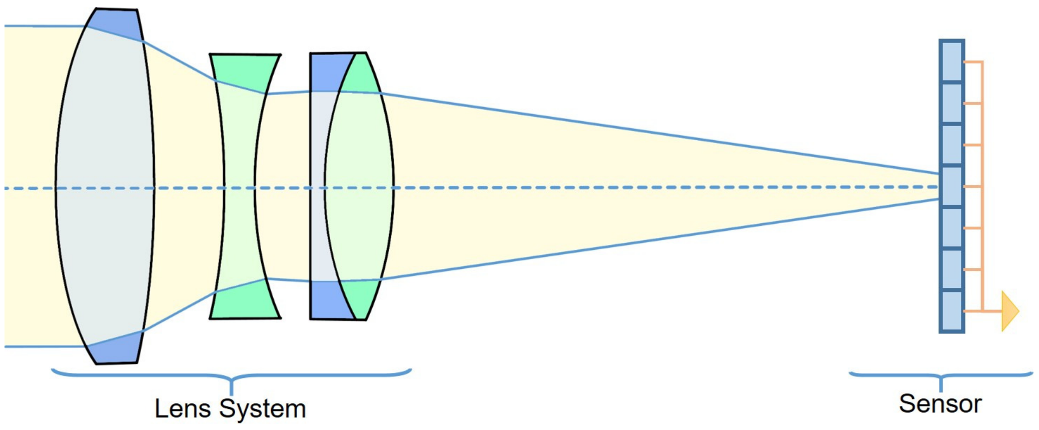


Figure 4.4: The demonstration of the imaging intensity, influenced by lens system and the sensor.

The property of the optical system can be formulated via the vignetting $V(x)$:

$$E_x = V(x)L_x, \tag{4.9}$$

where $L_x$ is the radiance of the incident scene ray $x$, and the $E_x$ is the irradiance of the emitted light.

The property of the optical-electronic sensor can be formulated via the radiometric response function $f(\cdot)$:

$$I_x = f(kE_x), \tag{4.10}$$

where the $I_x$ is the intensity of the pixel and $k$ is the exposure time.

Considering the property of the optical system and the optical-electronic sensor, the overall formation of each intensity is:

$$I_x = f(kV(x)L_x) \tag{4.11}$$

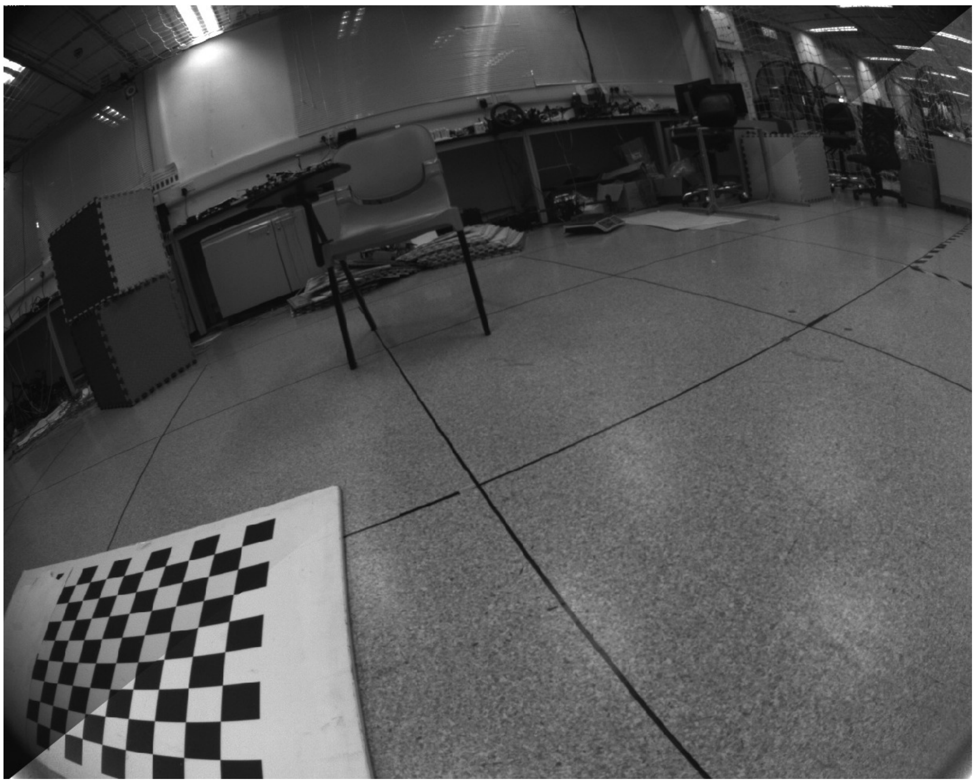


Figure 4.5: The demonstration of the vignetting. The top-left part is the raw image and the bottom-right part is removed vignetting.

A vignetting model which is radially symmetrical and formulated as a $k$-order polynomial:

$$V(r) = 1 + \sum_{i=1}^{k} \beta_i r^{2i} \tag{4.12}$$

where the $r$ is the distance between the pixel and the center of the camera, $\beta_i$ is the $i$ order coefficient of the polynomial and $V(r)$ is the spatially varying optical response which is able to describe the unbalanced illumination, the vignetting.

For an image with vignetting, the intensity of each pixel is

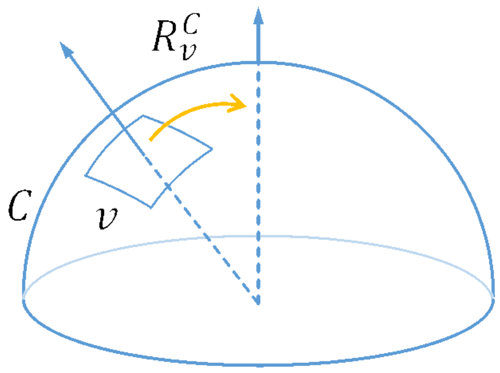
$$I(r) = I_0 V(r) = I_0 + I_0 \sum_{i=1}^{k} \beta_i r^{2i} \tag{4.13}$$

where the $I_0$ is the intensity of the camera principle point. For a typical camera lens, six order polynomial with three none-zero coefficients is able to fit the vignetting well, thus $k = 3$.

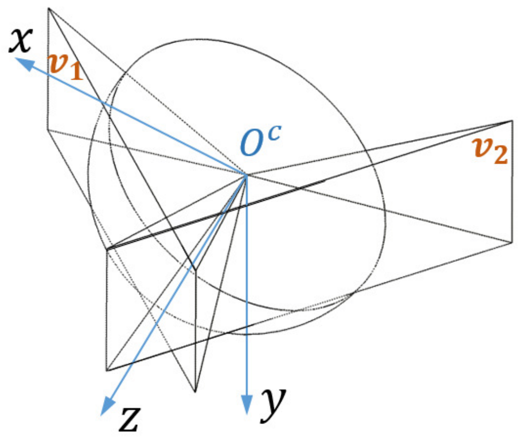
The vignetting compensation is able to improve the quality of the visual mapping system, as demonstration in [2].

## 4.4 ROI extraction

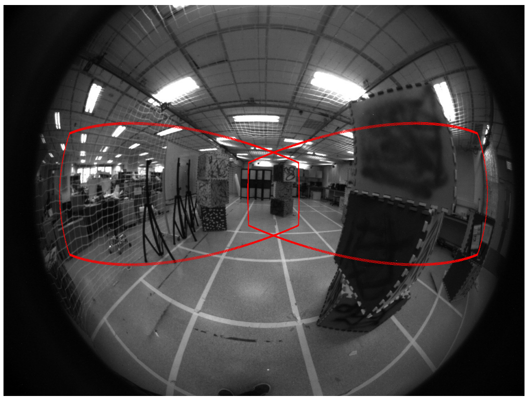
We have introduced the ROI extraction module for cropping regions of an original fisheye image and converting it into a distortion-free image that is applicable for the standard stereo matching algorithms.



(a) Illustration of ROI



(b) Crop two ROI images


(c) ROI regions on a fisheye image

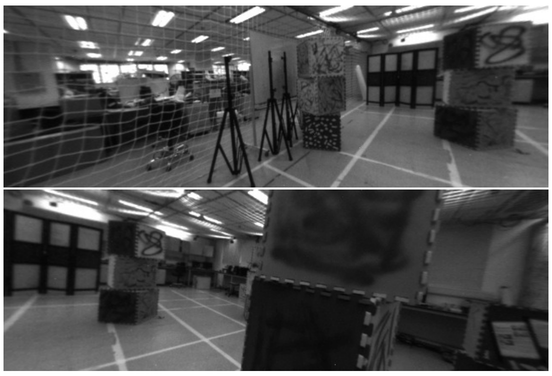

(d) Pinhole images from ROI

Figure 4.6: Illustrations of the ROI extraction process.

A two-step re-project is used: 1) back project each pixel of the desire virtual pinhole camera, whose intrinsic parameters are manually set, to a 3D space by $\pi_v^{-1}(\cdot)$; 2) to project these 3D points to the fisheye camera plane by $\pi_c(\cdot)$ (shown in Fig. 7.1(a)). The transformation between a pixel the in fisheye camera and a pixel in the virtual pinhole camera is:

$$\mathbf{u}^c = \pi_c(\mathbf{R}_v^c \pi_v^{-1}(\mathbf{u}^v)), \tag{4.14}$$

where $\pi_c(\cdot)$ is the projection function of the fisheye camera introduced in [21], with $\mathbf{u}^c$ being the pixel coordinates of the fisheye image. $\pi_v^{-1}(\cdot)$ is the back projection function of the virtual camera. $\mathbf{u}^v$ is the pixel coordinates of the extracted image, and $\mathbf{R}_v^c$ is the related orientation of the virtual camera in original camera frame.

The process can be shown in Fig. 4.6(a). The sphere figure out the spherical camera $C$ such as fisheye camera and the rectangle figure out the region of virtual camera $v$. Two ROIs that cover the 180-degree horizontal view are set in a fisheye image, as illustrate as $v_1$ and $v_2$ in Fig. 4.6(b), and the regions are marked out in Fig. 4.6(c). The two resulting distortion-free pinhole images from the ROI are shown in Fig. 4.6(d)

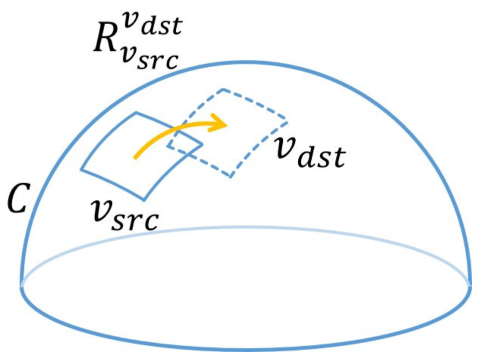


Figure 4.7: The illustration of the virtual gimbal

With the help of ROI extraction, we can implement a virtual gimbal with a virtual camera reproject. The virtual camera $v_{src}$ can be moved to $v_{dst}$, corresponding the fisheye camera $c$, shown in Fig. 4.7.

$$\begin{aligned}\mathbf{u}^c &= \pi_c(\mathbf{R}_{v_{src}}^c \pi_{v_{src}}^{-1}(\mathbf{u}^{v_{src}})),\\ \mathbf{u}^c &= \pi_c(\mathbf{R}_{v_{dst}}^c \pi_{v_{dst}}^{-1}(\mathbf{u}^{v_{dst}})),\end{aligned} \tag{4.15}$$

where $\pi_c(\cdot)$ is the projection function of the fisheye camera introduced, with $\mathbf{u}^c$ being the pixel coordinates of the fisheye image. $\pi_{v_{src}}^{-1}(\cdot)$ and $\pi_{v_{dst}}^{-1}(\cdot)$ is the back projection function of the virtual camera, original one and the destination one. $\mathbf{u}^{v_{src}}$ and $\mathbf{u}^{v_{dst}}$ is the pixel coordinates of the extracted image, and $\mathbf{R}_{v_{src}}^c$ and $\mathbf{R}_{v_{dst}}^c$ is the related orientation of the virtual camera in original camera frame.

Furthermore, a digital electronic image stabilization can be executed with a virtual camera reproject. As shown in Fig. 4.8, the real fisheye camera captured two images and the corresponding frame is $C$ and $C'$. There is a rotation $R_C^{C'}$ between $C$ and $C'$. In the frame $C$, the pixel map between the virtual camera $v$ and fisheye camera $C$ is:

$$\mathbf{u}^c = \pi_c(\mathbf{R}_v^c \pi_v^{-1}(\mathbf{u}^v)). \tag{4.16}$$

To keep the virtual camera $v$ being stable, in next image, the rotation $R_C^{C'}$ should be

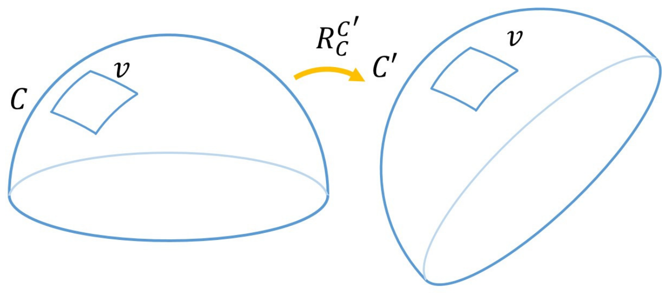


Figure 4.8: The illustration of the virtual electronic image stabilization

considered:

$$\mathbf{u}^{c'} = \pi_c(\mathbf{R}_c^{c'}\mathbf{R}_v^c\pi_v^{-1}(\mathbf{u}^v)), \tag{4.17}$$

where $\pi_c(\cdot)$ is the projection function of the fisheye camera introduced, with $\mathbf{u}^c$ being the pixel coordinates of the fisheye image. $\pi_v^{-1}(\cdot)$ is the back projection function of the virtual camera. $\mathbf{u}^v$ is the pixel coordinates of the extracted image, and $\mathbf{R}_v^c$ is the related orientation of the virtual camera in original camera frame.

# CHAPTER 5

# COMPUTER VISION ON SPHERE IMAGES

The details of the multiple view geometry in computer vision have been described in [26]. However, the main discussion focus on the perspective cameras. The definitions in perspective cameras are widely used, but not satisfy the spherical cameras. In this chapter, we modify the definitions to satisfy spherical cameras, including some essential definitions of geometry computer vision for spherical cameras include image coordinate evaluate (parallax and projection error) and the geometry computer vision (depth, epipolar constraint, homography, triangulation, and Perspective-n-Point).

## 5.1 Review of Sphere Images

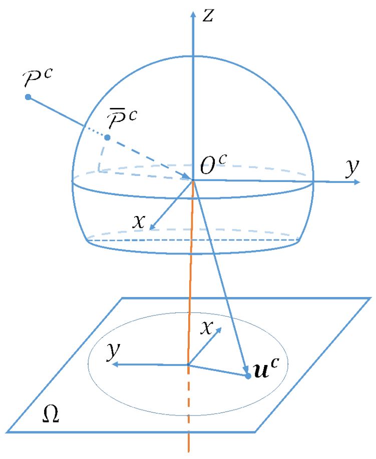


Figure 5.1: Projection from 3D point $\mathcal{P}_c$ to 2D point $\mathcal{P}_i$ . Back projection from 2D point $\mathcal{P}_i$ to 3D point on the unit sphere $\mathcal{P}_s$. $O_c$ and $O_{cl}$ are virtual points.

As shown in Fig. 5.1, the projection can be divide as three process:

1. from the point $\mathcal{P}^w$ in world frame to point in the camera frame $\mathcal{P}^c$;

2. from the point in camera frame $\mathcal{P}^c$ to incident vector $\bar{\mathcal{P}}^c\mathcal{O}^c$;

3. from the incident vector $\bar{\mathcal{P}}^c\mathcal{O}^c$ to pixel coordinate $\mathbf{u}^c = [u\ v]^T$;

For perspective cameras, all of the processes can get analytical solutions. For omnidirectional cameras and fisheye cameras, the analytical solution of process 3 is complex, even not exist. Thus it is improper to do other calculated in pixel coordinate. For perspective cameras, the projection satisfy $\mathbf{u}^c = [u \ v \ 1]^T = \mathbf{K} \left[u_x/u_z \ u_y/u_z \ 1\right]^T$, $\mathbf{K}$ is the intrinsic parameters matrix. Therefore, for perspective cameras, we can focus on the pixel coordinate. For ultra-wide filed-of-view camera and omnidirectional cameras, we need to consider the situation that $u_{iz} \in (-1, 1]$, unlike perspective camera model satisfy $u_{iz} = 1$. Thus, the geometry computer vision need to focus on the inverse vector of the incident vector $\bar{\mathcal{P}}^c\mathcal{O}^c$, that is $\mathcal{O}^c\bar{\mathcal{P}}^c = \mathbf{u} = \begin{bmatrix} u_x & u_y & u_z \end{bmatrix}^T$, a unit vector.

For one point $i$, the coordinate in world frame is $\mathrm{X}_i^w = \begin{bmatrix} X_{ix}^w & X_{iy}^w & X_{iz}^w \end{bmatrix}^T$. The projection of cameras can be describe as:

$$\lambda \mathbf{u}_i = \lambda \begin{bmatrix} u_{ix} \\ u_{iy} \\ u_{iz} \end{bmatrix} = \pi\left(\begin{bmatrix} \mathbf{R} & \mathbf{T} \end{bmatrix} \begin{bmatrix} X_{ix}^w \\ X_{iy}^w \\ X_{iz}^w \\ 1 \end{bmatrix}\right), \tag{5.1}$$

With this formulation, the perspective projection is support not only the perspective cameras, also the ultra-wide filed-of-view cameras, especially some cameras with over 270-degree filed-of-view.

## 5.2 Image Coordinate Evaluate

### 5.2.1 Parallax

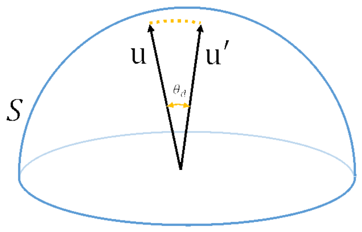


Figure 5.2: In the unit sphere $S$, the parallax of two image vector $\mathbf{u}$ and $\mathbf{u}'$ is the angle $\theta_d$.

For perspective cameras, the parallax of perspective cameras always define as the difference of two image points:

$$d = \tilde{\mathbf{u}} - \tilde{\mathbf{u}}', \tag{5.2}$$

where $\tilde{\mathbf{u}}$ and $\tilde{\mathbf{u}}'$ are the points in pixel coordinate. The parallax of spherical image is define as the parallax angle:

$$\theta_d = \arccos(\mathbf{u} \cdot \mathbf{u}'), \tag{5.3}$$

where $\mathbf{u}$ and $\mathbf{u}'$ is the image unit vectors in two frames. with the unit *rad* or degree (°), as shown in Fig. 5.2.

### 5.2.2 Projection Error

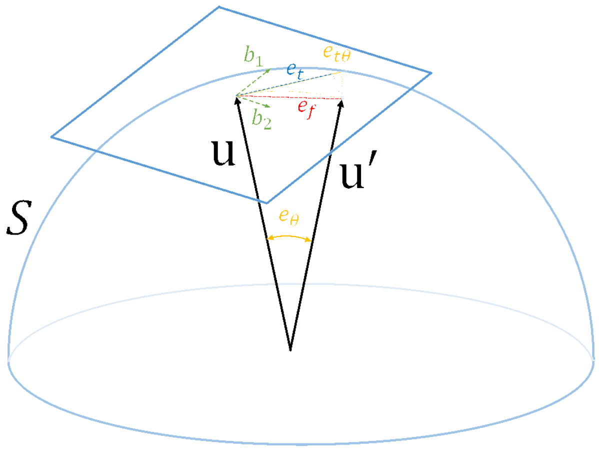


Figure 5.3: The illustration of different kinds of Projection Errors on the sphere.

For the perspective cameras, the projection error can be described as pixel coordinate error $e_u$ or the unit plane error $e_m$. For the spherical cameras, include omnidirectional cameras and the ultra-wide field-of-view cameras, the projection error is the difference between the two vectors. There are some methods to describe the projection error: angular error $e_\theta$, bearing vector difference error $e_f$, tangential error $e_t$.

The projection error is define as:

Pixel Coordinate Error The pixel coordinate error is define as the difference of two image points:

$$e_u = \tilde{\mathbf{u}} - \tilde{\mathbf{u}}' \tag{5.4}$$

where the $\tilde{\mathbf{u}}$ and $\tilde{\mathbf{u}}'$ are points in pixel coordinate.

Unit Plane Error To project the two points into a unit plane with $z = 1$, the unit plane error is define as:

$$e_m = \begin{bmatrix} \tilde{\mathbf{u}}_x/\tilde{\mathbf{u}}_z \\ \tilde{\mathbf{u}}_y/\tilde{\mathbf{u}}_z \end{bmatrix} - \begin{bmatrix} \tilde{\mathbf{u}}'_x/\tilde{\mathbf{u}}'_z \\ \tilde{\mathbf{u}}'_y/\tilde{\mathbf{u}}'_z \end{bmatrix} \tag{5.5}$$

where the $\tilde{\mathbf{u}}$ and $\tilde{\mathbf{u}}'$ are points in pixel coordinate.

Angular Error Same as Sect. 5.2.1, the projection error can be described with angular error. The parallax is define as the parallax angle:

$$e_\theta = \arccos(\mathbf{u} \cdot \mathbf{u}'), \tag{5.6}$$

with the unit *rad* or degree (°).

Vector Difference Error The vector difference error is define as the difference of two unit bearing vectors of points:

$$e_f = \mathbf{u} - \mathbf{u}' \tag{5.7}$$

where $\mathbf{u}$ and $\mathbf{u}'$ are the image unit bearing vectors of two points.

Tangential Error To project the difference of two unit bearing vectors of points onto a tangential plane, the tangential error is define as:

$$e_t = [b_1 \; b_2]\delta\mathbf{u} \tag{5.8}$$

where $\delta\mathbf{u} = \mathbf{u} - \mathbf{u}'$ is the difference of two unit bearing vectors; $b_1$ and $b_2$ are two orthogonal bases spanning the tangent plane [45].

## 5.3 Geometry Computer Vision

### 5.3.1 Depth

The depth of sphere camera is different with the perspective cameras. As shown in Fig. 5.4, for the sphere camera $C$, the depth $d$ is define as the distance between the point $X_i$ and the center of the camera:

$$d = \|OX_i\|, \tag{5.9}$$

where $O$ is the center point of the camera.

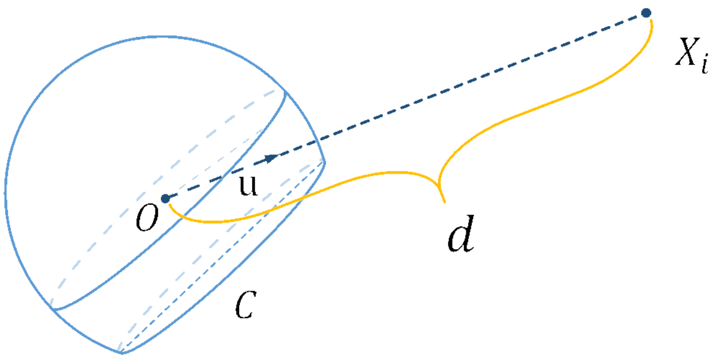


Figure 5.4: Depth of sphere camera.

## 5.3.2 Epipolar Constraint

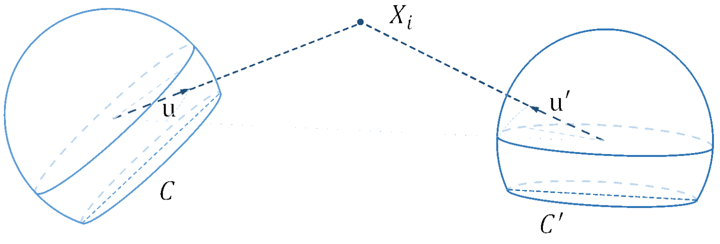


Figure 5.5: The illustration of epipolar constraint.

Epipolar Constraint We have seen that two perspective cameras observing the same points must satisfy the epipolar constraint:

$$\mathbf{u}'^T \mathrm{E} \mathbf{u} = 0, \tag{5.10}$$

where the essential matrix $\mathrm{E} = \lfloor t \rfloor_\times \mathrm{R}$. For one point $i$ captured in two frames, corresponding as $\mathbf{u}'_i = \begin{bmatrix} u'_{ix} & u'_{iy} & u'_{iz} \end{bmatrix}^T$ and $\mathbf{u}_i = \begin{bmatrix} u_{ix} & u_{iy} & u_{iz} \end{bmatrix}^T$:

$$\begin{bmatrix} u'_{ix} & u'_{iy} & u'_{iz} \end{bmatrix} \mathrm{E} \begin{bmatrix} u_{ix} \\ u_{iy} \\ u_{iz} \end{bmatrix} = 0, \tag{5.11}$$

Solve the essential matrix with epipolar constraint Rewrite matrix E:

$$\begin{bmatrix} u'_{ix} & u'_{iy} & u'_{iz} \end{bmatrix} \begin{bmatrix} e_1 & e_2 & e_3 \\ e_4 & e_4 & e_5 \\ e_7 & e_8 & e_9 \end{bmatrix} \begin{bmatrix} u_{ix} \\ u_{iy} \\ u_{iz} \end{bmatrix} = 0, \tag{5.12}$$

$$\begin{bmatrix} u_{ix}u'_{ix} & u_{iy}u'_{ix} & u_{iz}u'_{ix} & u_{ix}u'_{iy} & u_{iy}u'_{iy} & u_{iz}u'_{iy} & u_{ix}u'_{iz} & u_{iy}u'_{iz} & u_{iz}u'_{iz} \end{bmatrix} \mathrm{e} = 0, \tag{5.13}$$

where $\mathrm{e} = \begin{bmatrix} e_1 & e_2 & e_3 & e_4 & e_5 & e_6 & e_7 & e_8 & e_9 \end{bmatrix}^T$.

Consider multiple points measurements,

$$\begin{bmatrix} u_{ix}u'_{ix} & u_{iy}u'_{ix} & u_{iz}u'_{ix} & u_{ix}u'_{iy} & u_{iy}u'_{iy} & u_{iz}u'_{iy} & u_{ix}u'_{iz} & u_{iy}u'_{iz} & u_{iz}u'_{iz} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix} \mathrm{e} = 0, \tag{5.14}$$

$$\mathrm{Ae} = 0 \tag{5.15}$$

Thus, combining these equations we get an overdetermined system of linear equations that we can solve the essential matrix E with SVD.

### 5.3.3 Homography

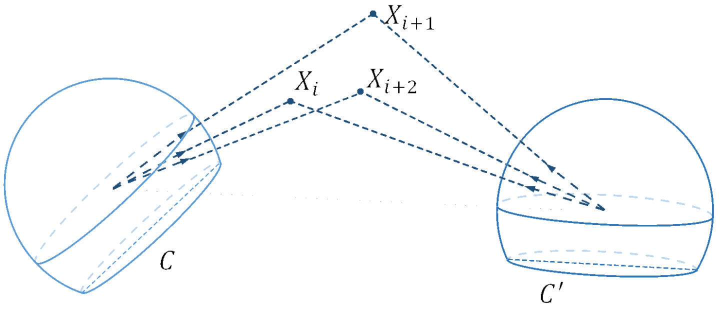


Figure 5.6: The illustration of homography.

One point $i$ is captured in two frames, corresponding as $\mathbf{u}'_i = \begin{bmatrix} u'_{ix} & u'_{iy} & u'_{iz} \end{bmatrix}^T$ and $\mathbf{u}_i = \begin{bmatrix} u_{ix} & u_{iy} & u_{iz} \end{bmatrix}^T$.

Assume that

$$\mathbf{n}^T\mathbf{P}_i + d = 0, \tag{5.16}$$

as the planar model in the camera frame $c'$, where the $\mathbf{P}_i$ is the point on the planar, $\mathbf{n}$ is the normal vector and $d$ is the distance of the camera frame and the planar.

The projection of one frame to another is:

$$\begin{aligned} \mathbf{u}_i &= \mathbf{RP}_i + \mathbf{t} \\ &= \mathbf{RP}_i + \mathbf{t}\left(-\frac{\mathbf{n}^T\mathbf{P}_i}{\mathbf{d}}\right) \\ &= \left(\mathbf{R} - \frac{\mathbf{tn}^T}{\mathbf{d}}\right)\mathbf{P}_i. \end{aligned} \tag{5.17}$$

With the scale $\lambda$, the point is $\mathbf{P}_i = \mathbf{u}'_i/\lambda$. Equation 5.17 can be write as:

$$\lambda\mathbf{u}_i = \mathrm{H}\mathbf{u}'_i, \tag{5.18}$$

Rewrite matrix H:

$$\lambda \begin{bmatrix} u_{ix} \\ u_{iy} \\ u_{iz} \end{bmatrix} = \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_4 & h_5 \\ h_7 & h_8 & h_9 \end{bmatrix} \begin{bmatrix} u'_{ix} \\ u'_{iy} \\ u'_{iz} \end{bmatrix}, \tag{5.19}$$

$$\begin{cases} \lambda u_{ix} = h_1 u'_{ix} + h_2 u'_{iy} + h_3 u'_{iz} \\ \lambda u_{iy} = h_4 u'_{ix} + h_4 u'_{iy} + h_5 u'_{iz} \\ \lambda u_{iz} = h_7 u'_{ix} + h_8 u'_{iy} + h_9 u'_{iz} \end{cases} \tag{5.20}$$

From equation 5.20, the unknown scale $\lambda$ is:

$$\lambda = \frac{h_1 u'_{ix} + h_2 u'_{iy} + h_3 u'_{iz}}{u_{ix}} \tag{5.21}$$

$$\lambda = \frac{h_4 u'_{ix} + h_4 u'_{iy} + h_5 u'_{iz}}{u_{iy}} \tag{5.22}$$

$$\lambda = \frac{h_7 u'_{ix} + h_8 u'_{iy} + h_9 u'_{iz}}{u_{iz}} \tag{5.23}$$

The scale $\lambda$ is unknown and can be written as equation 5.21, 5.22, and 5.23. In order to avoid the numerical problems, we find one of $\lambda$ equation from 5.21, 5.22, and 5.23 and solve with equation 5.20.

Consider multiple points measurements, if $\|u_{ix}\| > \|u_{iy}\|$ and $\|u_{ix}\| > \|u_{iz}\|$, solve with equation 5.21 and 5.20:

$$\begin{bmatrix} -\frac{u'_{ix}u_{iy}}{u_{ix}} & -\frac{u'_{iy}u_{iy}}{u_{ix}} & -\frac{u'_{iz}u_{iy}}{u_{ix}} & u'_{ix} & u'_{iy} & u'_{iz} & 0 & 0 & 0 \\ -\frac{u'_{ix}u_{iz}}{u_{ix}} & -\frac{u'_{iy}u_{iz}}{u_{ix}} & -\frac{u'_{iz}u_{iz}}{u_{ix}} & 0 & 0 & 0 & u'_{ix} & u'_{iy} & u'_{iz} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix} \mathrm{h} = 0, \tag{5.24}$$

if $\|u_{iy}\| > \|u_{ix}\|$ and $\|u_{iy}\| > \|u_{iz}\|$, solve with equation 5.22 and 5.20:

$$\begin{bmatrix} u'_{ix} & u'_{iy} & u'_{iz} & -\frac{u'_{ix}u_{ix}}{u_{iy}} & -\frac{u'_{iy}u_{ix}}{u_{iy}} & -\frac{u'_{iz}u_{ix}}{u_{iy}} & 0 & 0 & 0 \\ 0 & 0 & 0 & -\frac{u'_{ix}u_{iz}}{u_{iy}} & -\frac{u'_{iy}u_{iz}}{u_{iy}} & -\frac{u'_{iz}u_{iz}}{u_{iy}} & u'_{ix} & u'_{iy} & u'_{iz} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix} \mathrm{h} = 0, \tag{5.25}$$

if $\|u_{iz}\| > \|u_{ix}\|$ and $\|u_{iz}\| > \|u_{iy}\|$, solve with equation 5.23 and 5.20:

$$\begin{bmatrix} u'_{ix} & u'_{iy} & u'_{iz} & 0 & 0 & 0 & -\frac{u'_{ix}u_{ix}}{u_{iz}} & -\frac{u'_{iy}u_{ix}}{u_{iz}} & -\frac{u'_{iz}u_{ix}}{u_{iz}} \\ 0 & 0 & 0 & u'_{ix} & u'_{iy} & u'_{iz} & -\frac{u'_{ix}u_{iy}}{u_{iz}} & -\frac{u'_{iy}u_{iy}}{u_{iz}} & -\frac{u'_{iz}u_{iy}}{u_{iz}} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix} \mathrm{h} = 0, \tag{5.26}$$

where h $= \begin{bmatrix} h_1 & h_2 & h_3 & h_4 & h_5 & h_6 & h_7 & h_8 & h_9 \end{bmatrix}^T$. Thus, combining these equations we get an overdetermined system of linear equations that we can solve the homography matrix with SVD.

$$\mathrm{A}\mathrm{h} = 0 \tag{5.27}$$

### 5.3.4 Triangulation

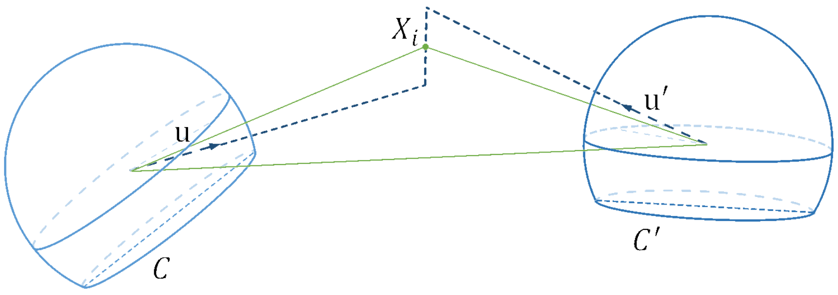


Figure 5.7: The illustration of triangulation.

The observations of a points $\mathrm{X}_i$ in multiple frames is:

$$\begin{aligned} P\mathrm{X}_i &= \mathbf{u} \\ P'\mathrm{X}_i &= \mathbf{u}' \end{aligned} \tag{5.28}$$

Linear triangulation, Minimizing the algebraic error

For one observed points perspective projection:

$$\begin{gathered} \mathbf{u}_i = P\mathrm{X}_i \\ \mathbf{u}_i \times P\mathrm{X}_i = 0 \\ \begin{bmatrix} u_{ix} \\ u_{iy} \\ u_{iz} \end{bmatrix} \times \begin{bmatrix} \mathbf{p^{1T}} \\ \mathbf{p^{2T}} \\ \mathbf{p^{3T}} \end{bmatrix} \mathrm{X}_i = 0 \\ \begin{bmatrix} u_{iy}\mathbf{p^{3T}} - u_{iz}\mathbf{p^{2T}} \\ u_{iz}\mathbf{p^{1T}} - u_{ix}\mathbf{p^{3T}} \\ u_{ix}\mathbf{p^{2T}} - u_{iy}\mathbf{p^{1T}} \end{bmatrix} \mathrm{X}_i = 0 \end{gathered} \tag{5.29}$$

This algorithm uses the two equations for perspective projection to solve for the 3D point that are optimal in a least squares sense. Each perspective camera model gives rise to there equations on the three entries of $\mathrm{X}_i$.

$$\begin{bmatrix} u_{iy}\mathbf{p^{3T}} - u_{iz}\mathbf{p^{2T}} \\ u_{iz}\mathbf{p^{1T}} - u_{ix}\mathbf{p^{3T}} \\ u_{ix}\mathbf{p^{2T}} - u_{iy}\mathbf{p^{1T}} \\ y_i'\mathbf{p'^{3T}} - z_i'\mathbf{p'^{2T}} \\ z_i'\mathbf{p'^{1T}} - x_i'\mathbf{p'^{3T}} \\ x_i'\mathbf{p'^{2T}} - y_i'\mathbf{p'^{1T}} \end{bmatrix} \mathrm{X}_i = 0 \tag{5.30}$$

$$\begin{bmatrix} u_{iy}p_{31} - u_{iz}p_{21} & u_{iy}p_{32} - u_{iz}p_{22} & u_{iy}p_{33} - u_{iz}p_{23} & u_{iy}p_{34} - u_{iz}p_{24} \\ u_{iz}p_{11} - u_{ix}p_{31} & u_{iz}p_{12} - u_{ix}p_{32} & u_{iz}p_{13} - u_{ix}p_{33} & u_{iz}p_{14} - u_{ix}p_{34} \\ u_{ix}p_{21} - u_{iy}p_{11} & u_{ix}p_{22} - u_{iy}p_{12} & u_{ix}p_{23} - u_{iy}p_{13} & u_{ix}p_{24} - u_{iy}p_{14} \\ y'_ip'_{31} - z'_ip'_{21} & y'_ip'_{32} - z'_ip'_{22} & y'_ip'_{33} - z'_ip'_{23} & y'_ip'_{34} - z'_ip'_{24} \\ z'_ip'_{11} - x'_ip'_{31} & z'_ip'_{12} - x'_ip'_{32} & z'_ip'_{13} - x'_ip'_{33} & z'_ip'_{14} - x'_ip'_{34} \\ x'_ip'_{21} - y'_ip'_{11} & x'_ip'_{22} - y'_ip'_{12} & x'_ip'_{23} - y'_ip'_{13} & x'_ip'_{24} - y'_ip'_{14} \end{bmatrix} \mathrm{X}_i = 0 \tag{5.31}$$

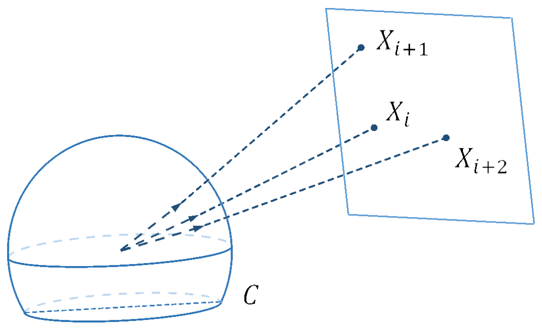
$$\mathrm{AX}_i = 0 \tag{5.32}$$

Combining these equations we get an overdetermined system of linear equations that we can solve with SVD.

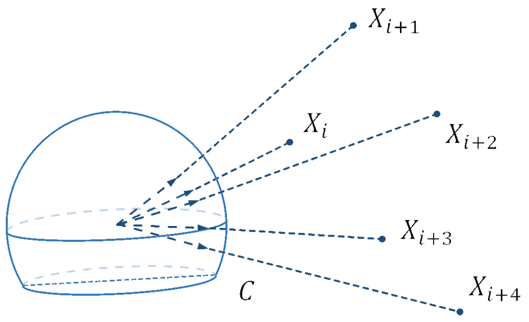
For multiple observations, the minimized algebraic error is not geometrically meaningful, but the method extends naturally to the case when $\mathrm{X}_i$ is observed in more than frames images.

$$\begin{bmatrix} u_{iy}\mathbf{p^{3T}} - u_{iz}\mathbf{p^{2T}} \\ u_{iz}\mathbf{p^{1T}} - u_{ix}\mathbf{p^{3T}} \\ u_{ix}\mathbf{p^{2T}} - u_{iy}\mathbf{p^{1T}} \\ y'_i\mathbf{p'^{3T}} - z'_i\mathbf{p'^{2T}} \\ z'_i\mathbf{p'^{1T}} - x'_i\mathbf{p'^{3T}} \\ x'_i\mathbf{p'^{2T}} - y'_i\mathbf{p'^{1T}} \\ \vdots \end{bmatrix} \mathrm{X}_i = 0 \qquad \mathrm{AX}_i = 0 \tag{5.33}$$

### 5.3.5 Perspective-n-Point



(a) the world points $\mathrm{X}^w$ is in the same planar



(b) the world points $\mathrm{X}^w$ is not in the same planar

Figure 5.8: The illustration of Perspective-n-Point.

As shown in Fig. 5.8, for the world point $\mathrm{X}_i^w = \begin{bmatrix} X_{ix}^w & X_{iy}^w & X_{iz}^w \end{bmatrix}^T$, the projection of the spherical camera is:

$$\lambda \mathbf{u}_i = \lambda \begin{bmatrix} u_{ix} \\ u_{iy} \\ u_{iz} \end{bmatrix} = \begin{bmatrix} \mathbf{R} & \mathbf{T} \end{bmatrix} \begin{bmatrix} X_{ix}^w \\ X_{iy}^w \\ X_{iz}^w \\ 1 \end{bmatrix}, \tag{5.34}$$

where $\lambda$ is unknown scale.

If all of the world points $\mathrm{X}^w$ is in the same planar, solve with homography method in Sect. 5.3.3. If all of the world points $\mathrm{X}^w$ is not in the same planar, solve with DLT method.

Direct linear transformation Rewrite matrix $\begin{bmatrix}\mathbf{R} & \mathbf{T}\end{bmatrix}$:

$$\lambda \begin{bmatrix} u_{ix} \\ u_{iy} \\ u_{iz} \end{bmatrix} = \begin{bmatrix} p_1 & p_2 & p_3 & p_4 \\ p_5 & p_6 & p_7 & p_8 \\ p_9 & p_{10} & p_{11} & p_{12} \end{bmatrix} \begin{bmatrix} X^w_{ix} \\ X^w_{iy} \\ X^w_{iz} \\ 1 \end{bmatrix}, \tag{5.35}$$

$$\begin{cases} \lambda u_{ix} = p_1 X^w_{ix} + p_2 X^w_{iy} + p_3 X^w_{iz} + p_4, \\ \lambda u_{iy} = p_5 X^w_{ix} + p_6 X^w_{iy} + p_7 X^w_{iz} + p_8, \\ \lambda u_{iz} = p_9 X^w_{ix} + p_{10} X^w_{iy} + p_{11} X^w_{iz} + p_{12}. \end{cases} \tag{5.36}$$

From equation 5.36, the unknown scale $\lambda$ is:

$$\lambda = \frac{p_1 X^w_{ix} + p_2 X^w_{iy} + p_3 X^w_{iz} + p_4}{u_{ix}} \tag{5.37}$$

$$\lambda = \frac{p_5 X^w_{ix} + p_6 X^w_{iy} + p_7 X^w_{iz} + p_8}{u_{iy}} \tag{5.38}$$

$$\lambda = \frac{p_9 X^w_{ix} + p_{10} X^w_{iy} + p_{11} X^w_{iz} + p_{12}}{u_{iz}} \tag{5.39}$$

The scale $\lambda$ is unknown and can be written as equation 5.37, 5.38, and 5.39. In order to avoid the numerical problems, we find one of $\lambda$ equation from 5.37, 5.38, and 5.39 and solve with equation 5.36.

If $\|u_{ix}\| > \|u_{iy}\|$ and $\|u_{ix}\| > \|u_{iz}\|$, solve with equation 5.37 and 5.36:

$$\begin{bmatrix} -\frac{X^w_{ix} u_{iy}}{u_{ix}} & -\frac{X^w_{iy} u_{iy}}{u_{ix}} & -\frac{X^w_{iz} u_{iy}}{u_{ix}} & -\frac{u_{iy}}{u_{ix}} & X^w_{ix} & X^w_{iy} & X^w_{iz} & 1 & 0 & 0 & 0 & 0 \\ -\frac{X^w_{ix} u_{iz}}{u_{ix}} & -\frac{X^w_{iy} u_{iz}}{u_{ix}} & -\frac{X^w_{iz} u_{iz}}{u_{ix}} & -\frac{u_{iz}}{u_{ix}} & 0 & 0 & 0 & 0 & X^w_{ix} & X^w_{iy} & X^w_{iz} & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix} \mathrm{P} = 0, \tag{5.40}$$

if $\|u_{iy}\| > \|u_{ix}\|$ and $\|u_{iy}\| > \|u_{iz}\|$, solve with equation 5.38 and 5.36:

$$\begin{bmatrix} X^w_{ix} & X^w_{iy} & X^w_{iz} & 1 & -\frac{X^w_{ix} u_{ix}}{u_{iy}} & -\frac{X^w_{iy} u_{ix}}{u_{iy}} & -\frac{X^w_{iz} u_{ix}}{u_{iy}} & -\frac{u_{ix}}{u_{iy}} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -\frac{X^w_{ix} u_{iz}}{u_{iy}} & -\frac{X^w_{iy} u_{iz}}{u_{iy}} & -\frac{X^w_{iz} u_{iz}}{u_{iy}} & -\frac{u_{iz}}{u_{iy}} & X^w_{ix} & X^w_{iy} & X^w_{iz} & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix} \mathrm{P} = 0, \tag{5.41}$$

if $\|u_{iz}\| > \|u_{ix}\|$ and $\|u_{iz}\| > \|u_{iy}\|$, solve with equation 5.39 and 5.36:

$$\begin{bmatrix} X^w_{ix} & X^w_{iy} & X^w_{iz} & 1 & 0 & 0 & 0 & 0 & -\frac{X^w_{ix}u_{ix}}{u_{iz}} & -\frac{X^w_{iy}u_{ix}}{u_{iz}} & -\frac{X^w_{iz}u_{ix}}{u_{iz}} & -\frac{u_{ix}}{u_{iz}} \\ 0 & 0 & 0 & 0 & X^w_{ix} & X^w_{iy} & X^w_{iz} & 1 & -\frac{X^w_{ix}u_{iy}}{u_{iz}} & -\frac{X^w_{iy}u_{iy}}{u_{iz}} & -\frac{X^w_{iz}u_{iy}}{u_{iz}} & -\frac{u_{iy}}{u_{iz}} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix} \mathrm{P} = 0, \tag{5.42}$$

where $\mathrm{P} = \begin{bmatrix} p_1 & p_2 & p_3 & p_4 & p_5 & p_6 & p_7 & p_8 & p_9 & p_{10} & p_{11} & p_{12} \end{bmatrix}^T$. Thus, combining these equations we get an overdetermined system of linear equations that we can solve with SVD.

$$\mathrm{AP} = 0 \tag{5.43}$$

Nonlinear Perspective-n-Point For the point $i$ in the spherical images, the reprojection error is defined as tangential error in Sect. 5.2.2:

$$\begin{aligned} e_i &= [b_{i1}\ b_{i2}](\mathbf{u}_i - \mathbf{u}'_i), \\ s.t.\mathbf{u}'_i &= \frac{\mathbf{R}\mathrm{X}^w_i + \mathbf{T}}{||\mathbf{R}\mathrm{X}^w_i + \mathbf{T}||} \end{aligned} \tag{5.44}$$

where $\mathbf{u}_i$ is the image unit bearing vector of point $i$ and $\mathbf{u}'_i$ is the normalized bearing vector in camera frame. $b_{i1}$ and $b_{i2}$ are the orthogonal bases spanning the tangent plane by vector $\mathbf{u}_i$ [45].

The nonlinear Perspective-n-Point is to minimize the sum of the Mahalanobis norm of the re-projection error of all the observed points via the rotation $\mathbf{R}$ and the translation $\mathbf{T}$:

$$\begin{aligned} &\min_{\mathbf{R},\mathbf{T}} \sum_i \|[b_{i1}\ b_{i2}](\mathbf{u}_i - \mathbf{u}'_i)\|^2, \\ &s.t.\mathbf{u}'_i = \frac{\mathbf{R}\mathrm{X}^w_i + \mathbf{T}}{||\mathbf{R}\mathrm{X}^w_i + \mathbf{T}||} \end{aligned} \tag{5.45}$$

where $i$ is the index of the observed points.

## 5.3.6 Structure From Motion

As shown in Fig. 5.9, for the world point $\mathrm{X}^w_j = \begin{bmatrix} X^w_{jx} & X^w_{jy} & X^w_{jz} \end{bmatrix}^T$ in the frame $i$, the projection of the spherical camera is equation (5.34).

The structure from motion for spherical image is to estimating the three-dimensional structures from two-dimensional image points that coupled with local motions.

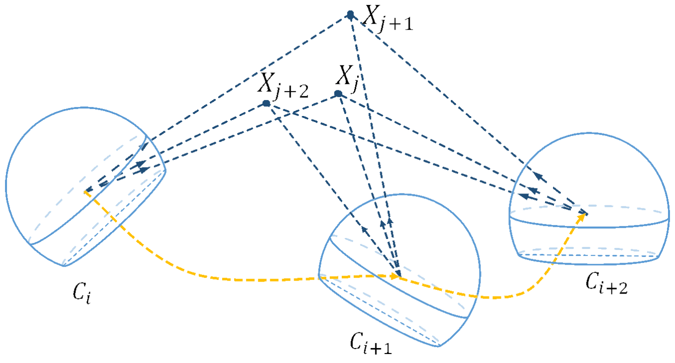


Figure 5.9: The illustration of Structure From Motion.

The cost function of the spherical structure from motion is similar to the cost function of PnP equation (5.45). The spherical structure from motion is to minimize the sum of the Mahalanobis norm of the re-projection error of all the observed points via the rotations $\mathbf{R}$ , the translations $\mathbf{T}$, and the position of the world points $\mathrm{X}^w$:

$$\begin{aligned} &\min_{\mathbf{R}_i,\mathbf{T}_i,\mathrm{X}_j^w} \sum_i \sum_j \left\| [b_{j1}\ b_{j2}](\mathbf{u}_j - \mathbf{u}_j') \right\|^2, \\ &s.t. \mathbf{u}_j' = \frac{\mathbf{R}_i \mathrm{X}_j^w + \mathbf{T}_i}{||\mathbf{R}_i \mathrm{X}_j^w + \mathbf{T}_i||} \end{aligned} \tag{5.46}$$

where $j$ is the index of the observed points in the frame $i$, $b_{i1}$ and $b_{i2}$ are the orthogonal bases spanning the tangent plane by vector $\mathbf{u}_i$ [45].

# CHAPTER 6

# DUAL-FISHEYE VISUAL-INERTIAL STATE ESTIMATION

For the autonomous flight, The state estimation is important to provide accurate states, including position, orientation, and velocity.

In this section, we improve the state-of-the-art tightly-coupled visual inertial navigation system [61] to satisfy dual-fisheye cameras, which construct with two ultra-wide FoV fisheye cameras. With ultra-wide FOV, cameras provide more spherical coverage of the surrounding environment and the probability of textureless situation is lower. Thus, the estimator is more robust than the perspective cameras.

## 6.1 Related Work

Visual-inertial estimation is an academic hotspot recently. There exist sparse visual-inertial estimation with monocular camera [28, 42, 65, 61, 41, 55, 5], stereo cameras [39], or RGB-D cameras [31]. In [11], the authors test some publicly-available VIO pipelines ( VINS-Mono[61], OKVIS[41], MSCKF[55], ROVIO[5], SVO+MSF, and SVO+GTSAM ) on different hardware configurations and evaluates the performance. Most of them are used for pinhole cameras.

In [62], the authors proposed an ego-motion estimator based on visual and inertial sensors named Omnidirectional Visual-Inertial Odometry (OVIO). The OVIO combines omnidirectional visual features with inertial measurements based on the Multi-State Constraint Kalman Filter (MSCKF). The authors extend the measurement model onto a plane tangent to the unit sphere rather than on the image plane is defined. The key hypothesis of the spherical visual inertial odometry is that a wider field of view allows the incorporation of more visual features from the surrounding environment, thereby improving the accuracy and robustness of the odometry estimation.

In [9], the authors propose a real-time, direct monocular SLAM method for omnidirectional or wide field-of-view fisheye cameras. The formulations of the direct image alignment tracking and the pixel-wise distance filtering mapping are built based on the

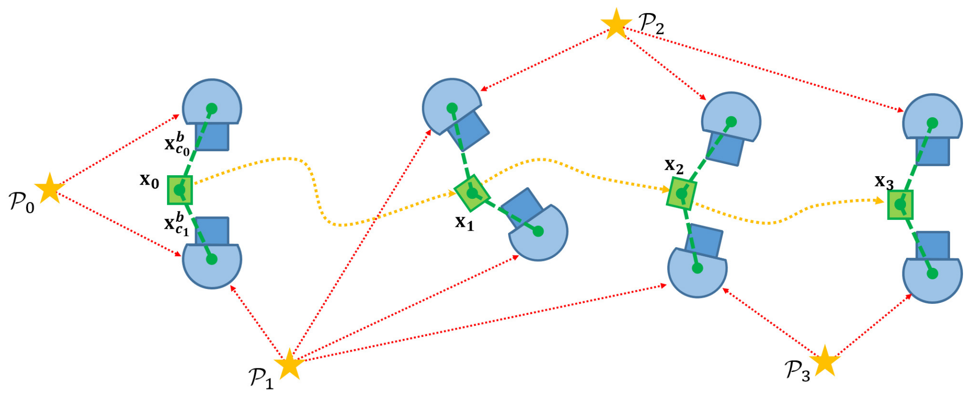


Figure 6.1: An example of the sliding window with four IMU states $\mathbf{x}_j$ and four features $\mathcal{P}_l$.

unified omnidirectional camera model, which can model central imaging devices with a large field of view. Additionally, to incremental stereo directly on distorted images with such a fast accurate mapping approach. The authors find that the system is able to observe and reconstruct a larger portion of the surrounding environments. And the system is more robust to handle the degenerate movement, such as rotation-only case.

In this paper, to achieve accurate online estimation, we use a sparse and feature-based tightly-coupled optimization-based visual-inertial framework with robust initialization. The method is to extracts and tracks features, pre-integrate IMU in the front-end, and minimizes visual and inertial geometry error in the back-end.

## 6.2 Sliding Window Formulation

The IMU and camera measurements are taken in a fixed time interval for state estimation. The algorithm achieves high-accurate state estimation by taking advantage of multi-view constraints. As shown in Fig. 6.1, several number of IMU and camera measurements are incorporated in the fixed window. Yellow lines represent the pre-integrated IMU measurements and red lines illustrate visual measurements. Note that there is a constant but unknown camera-IMU extrinsic calibration $\mathbf{x}_c^b$. All aforementioned quantities as well as IMU bias are jointly estimated in our framework.

The full state vector $\mathcal{X}$ in the sliding window with $n$ cameras, $m$ frames and $l$ features is defined as:

$$\begin{aligned}\mathcal{X} &= \left[\mathbf{x}_0,\, \mathbf{x}_1,\, \cdots\, \mathbf{x}_m,\, \mathbf{x}^b_{c_0},\, \mathbf{x}^b_{c_1},\, \cdots\, \mathbf{x}^b_{c_n},\, \lambda_0,\, \lambda_1,\, \cdots\, \lambda_l\right] \\ \mathbf{x}_i &= \left[\mathbf{p}^w_{b_i},\, \mathbf{v}^w_{b_i},\, \mathbf{q}^w_{b_i},\, \mathbf{b}^b_a,\, \mathbf{b}^b_g\right], k \in [0, m] \\ \mathbf{x}^b_{c_j} &= \left[\mathbf{p}^b_{c_j},\, \mathbf{q}^b_{c_j}\right], j \in [0, n]\end{aligned} \tag{6.1}$$

where $\mathbf{x}_i$ is the $i^{th}$ frame state, which contains position, velocity, and rotation in the world frame and acceleration bias and gyroscope bias in the IMU frame, treated as body frame. For concise representation, we use quaternion to denote rotation matrix here. $\mathbf{x}^b_{c_j}$ contains translation and rotation, which is the transformation from the $j^{th}$ camera frame to the IMU frame, extrinsic parameter. We parameterize $l^{th}$ feature in its inverse depth $\lambda_l$ form the first unit sphere observation.

## 6.3 Measurement Preprocessing

We consider two kinds of measurements in the state estimation. One is the sparse features of images, and the other is the IMU measurements. They are preprocessed then incorporated into the estimation. The features are detected and tracked from the image sequence in the feature processing front-end. And the IMU measurements are pre-integrated.

### 6.3.1 Feature Processing Front-End

For the dual-fisheye omnidirectional stereo system, there are two ultra-wide-FOV images cover the whole spherical sensing scope with both monocular, binocular and stereo, as shown in Fig. 1.2.

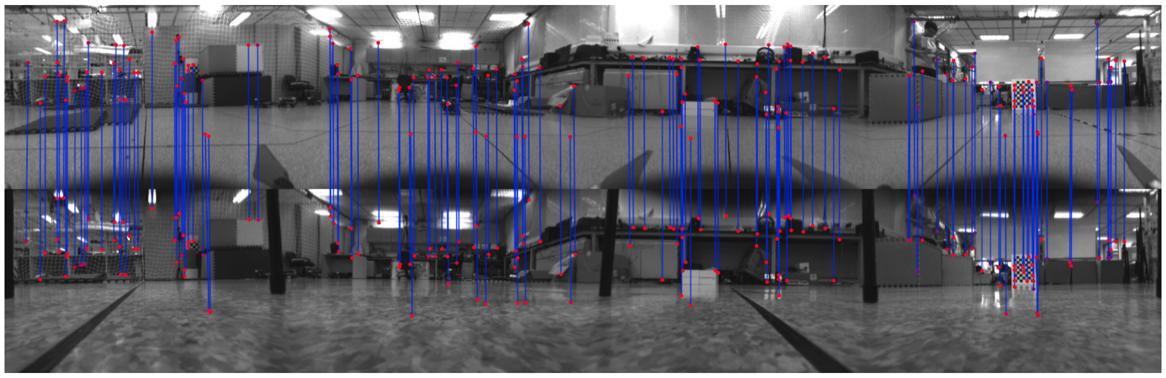


Figure 6.2: Tracked features of overlapping regions.

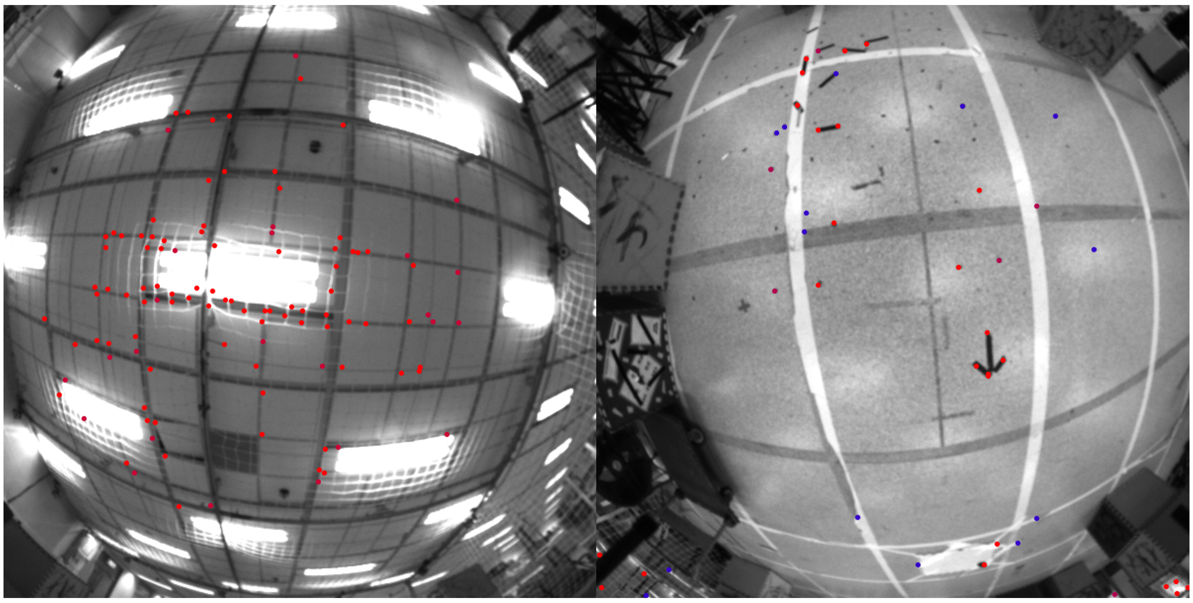


Figure 6.3: Tracked features in the monocular regions.

For each pair of new images, the ROI extraction preprocess will translate the two spherical images to 10 images. Two of them cover up-ward and down-ward with monocular vision, as shown in Fig. 6.3, and 8 of them cover front-ward, left-ward, right-ward, and back-ward with stereo vision, as shown in Fig. 6.2. The existing features are tracked by the KLT sparse optical flow algorithm [49] and new corner features are detected to maintain a minimum number of features $(150 - 250)$ in the whole sphere totally. Features are projected to two unit sphere using the camera model, one for the upward camera and another one for the downward camera. Outlier rejection performs by the RANSAC step.

During the feature tracing and detecting, the average parallax angle is calculated in order to select keyframes beyond a certain threshold. Also, new pair frames are treated as a keyframe if the number of tracked features goes below a certain threshold.

### 6.3.2 IMU Pre-Integration

The IMU measurements are angular velocity $\hat{\boldsymbol{\omega}}_t^b$ and linear acceleration $\hat{\mathbf{a}}_t^b$ in IMU frame, affected by bias $\mathbf{b}$ and noise $\boldsymbol{\eta}$,

$$\begin{aligned}\hat{\boldsymbol{\omega}}_t^b &= \boldsymbol{\omega}_t^b + \mathbf{b}_g + \boldsymbol{\eta}_g \\ \hat{\mathbf{a}}_t^b &= \mathbf{R}(\mathbf{q}_t^w)^{\mathrm{T}}(\mathbf{a}_t^w + \mathbf{g}^w) + \mathbf{b}_a + \boldsymbol{\eta}_a.\end{aligned} \tag{6.2}$$

For two time instants $k$ and $k+1$ that correspond to image frame $b_k$ and $b_{k+1}$, the linear acceleration and angular velocity in the local frame $b_k$ can be pre-integrated as:

$$
\begin{aligned}
\boldsymbol{\alpha}_{b_{k+1}}^{b_k} &= \iint_{t\in[k,k+1]} \boldsymbol{\gamma}_{b_t}^{b_k} \hat{\mathbf{a}}_t dt^2 \\
\boldsymbol{\beta}_{b_{k+1}}^{b_k} &= \int_{t\in[k,k+1]} \boldsymbol{\gamma}_{b_t}^{b_k} \hat{\mathbf{a}}_t dt \\
\boldsymbol{\gamma}_{b_{k+1}}^{b_k} &= \int_{t\in[k,k+1]} \boldsymbol{\gamma}_{b_t}^{b_k} \otimes \begin{bmatrix} 0 \\ \frac{1}{2}\hat{\boldsymbol{\omega}}_t \end{bmatrix} dt,
\end{aligned}
\tag{6.3}
$$

where $\otimes$ denotes the quaternion multiplication operation. The pre-integration part is obtained solely with IMU measurements within $[k, k+1]$ since its independence for pose and velocity.

## 6.4 Initialization

We do need a good initial guess to bootstrap the whole non-linear tightly-coupled visual-inertial optimization system. And the IMU pre-integration result cannot be directly used as the initial guess without the known of the initial attitude and velocity. Furthermore, the estimation is dramatically affected by the unknown biases of the IMU measurements.

A loosely-coupled sensor fusion method is adopted to get the initialization. For stereo overlapping of our system, a stereo visual odometry is able to initialize itself. And we find that stereo vision SLAM has a good property of initialization. A stereo visual-only system can bootstrap itself by a derived initial guess. Through loosely-coupled alignment between the stereo visual odometry and IMU metric pre-integration information, the gravity, velocity, and gyroscope bias can be roughly recovered. They satisfy with bootstrapping a nonlinear system.

We first construct the stereo visual odometry of the sliding window, including pose and feature position. Then the information of the visual odometry is aligned with the IMU information, to calculate the velocity, gravity vector and the bias of the gyroscope. The visual odometry is nonlinear, but it can bootstrap itself stably and the scale is approximate since the stereo baseline for sure. The second step is in linear form.

The extrinsic parameter $\mathbf{x}_{c_i}^b$ between camera $i$ and IMU is needed in the initial step. Note that only an initial guess about the extrinsic parameter is needed here, and accurate estimation will be carried out in non-linear optimization.

### 6.4.1 Visual Only Odometry in Sliding Window

In this step, we try to construct the vision only structure (up-to-scale camera pose for the monocular region, scaled camera pose for the stereo region, and feature position) within the window.

For the monocular region, we firstly choose two keyframes which contain sufficient feature parallax in the sliding window. Next, we use the Five-point method [57] to recover the relative rotation and up-to-scale translation between these two frames, then fix the scale of translation and triangulate the features observed in these two frames. For the stereo region, since existing the feature match between different cameras, and an approximate translation between the two cameras is known, feature parallax exists in the two images and every frame can be chosen as a keyframe. we use stereo triangulate to estimate the position of observed features. Based on these triangulated features, Perspective-n-Point method is performed to estimate all frame poses in the sliding window in Sect. 5.3.5.

A global full Bundle Adjustment [72] is then applied to minimize the total re-projection error of all feature observations in all frames. After that, we get all keyframe poses ($\bar{\mathbf{p}}_{c_i}^{c_0}$,$\mathbf{q}_{c_i}^{c_0}$) and feature positions. Here, $\bar{(\cdot)}$ denotes up to scale variables. We write all poses in the local frame $(\cdot)^{c_0}$, which is the first camera frame in the window. Since the extrinsic parameter ($\mathbf{p}_c^b$, $\mathbf{q}_c^b$) between camera frame and IMU (body) frame is known, all variables are transformed into the IMU frame,

$$\mathbf{q}_{b_i}^{b_0} = \mathbf{q}_c^b \otimes \mathbf{q}_{c_i}^{c_0}, \tag{6.4}$$

For the feature points in the stereo overlapping region, the keyframe poses is

$$\bar{\mathbf{p}}_{b_i}^{b_0} = \mathbf{q}_c^b \bar{\mathbf{p}}_{c_i}^{c_0} + \mathbf{p}_c^b. \tag{6.5}$$

Different with the stereo case, if all feature points are observed in monocular region, the keyframe poses is up-to-scale:

$$s\bar{\mathbf{p}}_{b_i}^{b_0} = s\mathbf{q}_c^b \bar{\mathbf{p}}_{c_i}^{c_0} + \mathbf{p}_c^b, \tag{6.6}$$

where $s$ is the unknown scale, which will be solved in the next section.

### 6.4.2 Visual-Inertial Alignment

IMU Bias Initialization Assume the IMU gyroscope bias $\mathbf{b}_g$ is constant in current window. Considering two consecutive frames $b_k$ and $b_{k+1}$ in the window, we have the relative

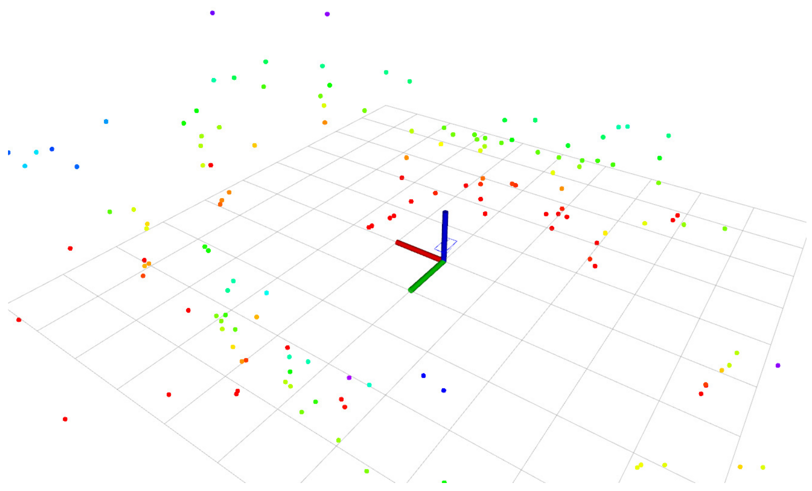


Figure 6.4: Initialization features of overlapping regions.

rotation $\mathbf{q}_{b_k}^{b_0}$ and $\mathbf{q}_{b_{k+1}}^{b_0}$ from the visual structure, as well as rotation propagation result $\hat{\gamma}_{b_{k+1}}^{b_k}$ from the IMU pre-integration. We estimate the gyroscope bias by minimizing the error between these two terms:

$$\begin{aligned} &\min_{b_g} \sum_{k \in \mathcal{B}} \left\| {\mathbf{q}_{b_{k+1}}^{b_0}}^{-1} \otimes \mathbf{q}_{b_k}^{b_0} \otimes \boldsymbol{\gamma}_{b_{k+1}}^{b_k} \right\|^2 \\ &\boldsymbol{\gamma}_{b_{k+1}}^{b_k} \approx \hat{\boldsymbol{\gamma}}_{b_{k+1}}^{b_k} \otimes \begin{bmatrix} 1 \\ \frac{1}{2} \frac{\partial \boldsymbol{\gamma}_{b_{k+1}}^{b_k}}{\partial \mathbf{b}_g} \mathbf{b}_g \end{bmatrix}, \end{aligned} \tag{6.7}$$

where $\mathcal{B}$ indexes the IMU measurement in the window. By solving this least square problem, we can get the estimation of $\mathbf{b}_g$. Then we update $\hat{\boldsymbol{\alpha}}_{b_{k+1}}^{b_k}, \hat{\boldsymbol{\beta}}_{b_{k+1}}^{b_k}$ with respect to $\mathbf{b}_g$.

As for the accelerometer bias, it is hard to solve it at the initialization procedure, since sufficient rotation movements are needed to distinguish the accelerometer bias and the gravity, when we are solving $\mathbf{g}^{b_0}$ at the same time. However, giving a coarse initial guess is adequate because we will continuously refine the bias after the initialization. Hence, we treat accelerometer bias $\mathbf{b}_a$ as zero in the initialization step.

Velocity and Gravity Vector Initialization For the window contains features of the stereo region, we solve the velocity and gravity vector. We define the variables that we would like to estimate as

$$\mathcal{X}_I = \left[\mathbf{v}_{b_0}^{b_0}, \mathbf{v}_{b_1}^{b_0}, \cdots \mathbf{v}_{b_n}^{b_0}, \mathbf{g}^{b_0}\right], \tag{6.8}$$

where $s$ is the scale parameter that aligns the visual structure to the actual metric scale implicitly provided by IMU measurements. Based on the Newton kinematic, we can get the equation as the following form shown:

$$\hat{\mathbf{z}}^{b_k}_{b_{k+1}} = \begin{bmatrix} \hat{\boldsymbol{\alpha}}^{b_k}_{b_{k+1}} \\ \hat{\boldsymbol{\beta}}^{b_k}_{b_{k+1}} \end{bmatrix} = \mathbf{H}^{b_k}_{b_{k+1}} \mathcal{X}_I + \mathbf{n}^{b_k}_{b_{k+1}} \approx \begin{bmatrix} -\mathbf{q}^{b_k}_{b_0} \Delta t_k & \mathbf{0} & \frac{1}{2}\mathbf{q}^{b_k}_{b_0} \Delta t_k^2 \\ -\mathbf{q}^{b_k}_{b_0} & \mathbf{q}^{b_k}_{b_0} & \mathbf{q}^{b_k}_{b_0} \Delta t_k \end{bmatrix} \begin{bmatrix} \mathbf{v}^{b_0}_{b_k} \\ \mathbf{v}^{b_0}_{b_{k+1}} \\ \mathbf{g}^{b_0} \end{bmatrix}. \tag{6.9}$$

Velocity, Gravity Vector and Metric Scale Initialization For the window contains features of the monocular region, we solve the velocity, gravity vector, and metric scale since the pose is up-to-scale. We define the variables that we would like to estimate as

$$\mathcal{X}_I = \left[\mathbf{v}^{b_0}_{b_0}, \mathbf{v}^{b_0}_{b_1}, \cdots \mathbf{v}^{b_0}_{b_n}, \mathbf{g}^{b_0}, s\right], \tag{6.10}$$

where $s$ is the scale parameter that aligns the visual structure to the actual metric scale implicitly provided by IMU measurements. Based on the Newton kinematic, we can get the equation as the following form shown:

$$\hat{\mathbf{z}}^{b_k}_{b_{k+1}} = \mathbf{H}^{b_k}_{b_{k+1}} \mathcal{X}_I + \mathbf{n}^{b_k}_{b_{k+1}} \approx \begin{bmatrix} -\mathbf{q}^{b_k}_{b_0} \Delta t_k & \mathbf{0} & \frac{1}{2}\mathbf{q}^{b_k}_{b_0} \Delta t_k^2 & \mathbf{q}^{b_k}_{b_0}(\bar{\mathbf{p}}^{b_0}_{b_{k+1}} - \bar{\mathbf{p}}^{b_0}_{b_k}) \\ -\mathbf{q}^{b_k}_{b_0} & \mathbf{q}^{b_k}_{b_0} & \mathbf{q}^{b_k}_{b_0} \Delta t_k & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{v}^{b_0}_{b_k} \\ \mathbf{v}^{b_0}_{b_{k+1}} \\ \mathbf{g}^{b_0} \\ s \end{bmatrix}. \tag{6.11}$$

In the above formula, $\mathbf{q}^{b_0}_{b_k}, \bar{\mathbf{p}}^{b_0}_{b_k}, \bar{\mathbf{p}}^{b_0}_{b_{k+1}}$ are obtained from the visual structure with respect to body frame $(\cdot)^{b_0}$. $\Delta t_k$ is the time interval between two consecutive keyframes. By solving the following least square problem:

$$\min_{\mathcal{X}_I} \sum_{k \in \mathcal{B}} \left\| \hat{\mathbf{z}}^{b_k}_{b_{k+1}} - \mathbf{H}^{b_k}_{b_{k+1}} \mathcal{X}_I \right\|^2, \tag{6.12}$$

we can get the velocities in every local frame, and the gravity vector in the visual base frame $(\cdot)^{b_0}$, as well as the scale parameter. The translational components $\bar{\mathbf{p}}^{b_i}$ from the visual structure will be scaled to the metric units. We rotate all variables from frame $(\cdot)^{b_0}$ to the world frame $(\cdot)^w$ where the gravity vector is vertical. At this point, the initialization procedure is completed and these metric values will be fed for a tightly-coupled nonlinear visual-inertial estimator.

## 6.5 Tightly-Coupled Visual-Inertial Localization

After state initialization, we proceed with a sliding window nonlinear estimator for high-accuracy state estimation. This is an extension of our earlier work [69, 73] by including

IMU bias calibration in the nonlinear optimization framework.

## 6.5.1 Formulation

We minimize the sum of the Mahalanobis norm of all measurement residuals to obtain a maximum of a posteriori estimation:

$$\min_{\mathcal{X}} \left\{ \|\mathbf{r}_p - \mathbf{H}_p \mathcal{X}\|^2 + \sum_{k \in \mathcal{B}} \left\| r_{\mathcal{B}}(\hat{\mathbf{z}}^{b_k}_{b_{k+1}}, \mathcal{X}) \right\|^2_{\mathbf{P}^{b_k}_{b_{k+1}}} + \sum_{(l,j) \in \mathcal{C}} \left\| r_{\mathcal{C}}(\hat{\mathbf{z}}^{c_j}_{l}, \mathcal{X}) \right\|^2_{\mathbf{P}^{c_j}_{l}} \right\}, \tag{6.13}$$

where $r_{\mathcal{B}}(\hat{\mathbf{z}}^{b_k}_{bk+1}, \mathcal{X})$ and $r_{\mathcal{C}}(\hat{\mathbf{z}}^{c_j}_{l}, \mathcal{X})$ are measurement residuals for the IMU and camera respectively. $\mathcal{B}$ is the set of all IMU measurements, and $\mathcal{C}$ is the set of feature observations in the corresponding frame. Corresponding measurement models are defined in Sect. 6.5.2 and Sect. 6.5.3. $\{\mathbf{r}_p, \mathbf{H}_p\}$ is the prior information, which will be discussed in Sect. 6.5.4. We use error-state representation to linearize the nonlinear system (6.13) and solve it via Gauss-Newton algorithm.

## 6.5.2 IMU Measurement Model

Following the kinematics theory, the residual of a pre-integrated IMU measurement can be defined as

$$r_{\mathcal{B}}(\hat{\mathbf{z}}^{b_k}_{b_{k+1}}, \mathcal{X}) = \begin{bmatrix} \delta\boldsymbol{\alpha}^{b_k}_{b_{k+1}} \\ \delta\boldsymbol{\beta}^{b_k}_{b_{k+1}} \\ \delta\boldsymbol{\gamma}^{b_k}_{b_{k+1}} \\ \delta\mathbf{b}_{a}{}^{b_k}_{b_{k+1}} \\ \delta\mathbf{b}_{g}{}^{b_k}_{b_{k+1}} \end{bmatrix} = \begin{bmatrix} \mathbf{q}^{b_k}_{w}(\mathbf{p}^{w}_{b_{k+1}} - \mathbf{p}^{w}_{b_k} + \mathbf{g}^{w}\Delta t^2/2) - \mathbf{v}^{b_k}_{b_k}\Delta t - \hat{\boldsymbol{\alpha}}^{b_k}_{b_{k+1}} \\ \mathbf{q}^{b_k}_{w}(\mathbf{q}^{w}_{b_{k+1}}\mathbf{v}^{b_{k+1}}_{b_{k+1}} + \mathbf{g}^{w}\Delta t) - \mathbf{v}^{b_k}_{b_k} - \hat{\boldsymbol{\beta}}^{b_k}_{b_{k+1}} \\ 2\left[\hat{\gamma}^{b_k^{-1}}_{b_{k+1}} \otimes \mathbf{q}^{b_w^{-1}}_{b_k} \otimes \mathbf{q}^{b_w}_{b_{k+1}}\right]_{xyz} \\ \mathbf{b}_{a}{}^{b_{k+1}}_{b_{k+1}} - \mathbf{b}_{a}{}^{b_k}_{b_k} \\ \mathbf{b}_{g}{}^{b_{k+1}}_{b_{k+1}} - \mathbf{b}_{g}{}^{b_k}_{b_k} \end{bmatrix}, \tag{6.14}$$

where $\left[\cdot\right]_{xyz}$ extracts the vector part of the quaternion $\mathbf{q}$, which is the approximation of error state representation. $[\hat{\boldsymbol{\alpha}}^{b_k}_{b_{k+1}}, \hat{\boldsymbol{\beta}}^{b_k}_{b_{k+1}}, \hat{\gamma}^{b_k}_{b_{k+1}}]^{\mathrm{T}}$ is the pre-integrated IMU measurement using only noisy accelerometer and gyroscope measurements, which is related to accelerometer and gyroscope bias and independent of initial velocity and attitude.

The covariance matrix $\mathbf{P}^{b_k}_{b_{k+1}}$ can then be calculated by first-order discrete-time propagation within the time interval $[k, k+1]$.

### 6.5.3 Camera Measurement Model

To utilize the benefit of large FOV camera, we define the camera measurement residual on an unit sphere as proposed in [46], which matches the fisheye camera model. As shown in Fig 6.5, the camera residual for the observation of the $l^{th}$ feature which first captured by the ${k_m}^{th}$ camera, in the $j^{th}$ frame captured by the ${k_n}^{th}$ camera is defined as

$$r_{\mathcal{C}}(\hat{\mathbf{z}}_l^{c_j}, \mathcal{X}) = \begin{bmatrix}\mathbf{b}_1 & \mathbf{b}_2\end{bmatrix}^T \cdot (\hat{\mathcal{P}}_l^{c_j^{k_n}} - \frac{\mathcal{P}_l^{c_j^{k_n}}}{\|\mathcal{P}_l^{c_j^{k_n}}\|}), \tag{6.15}$$

where

$$\hat{\mathcal{P}}_l^{c_j^{k_n}} = \begin{bmatrix}\hat{x}_l^{c_j^{k_n}} \\ \hat{y}_l^{c_j^{k_n}} \\ \hat{z}_l^{c_j^{k_n}}\end{bmatrix}, \tag{6.16}$$

is the observation of the $l^{th}$ feature in the $j^{th}$ image, $k_n^{th}$ camera.

The general formulation of the camera measurements of the $l^{th}$ feature which first captured by the ${k_m}^{th}$ camera, in the $j^{th}$ frame captured by the ${k_n}^{th}$ camera is

$$\mathcal{P}_l^{c_j^{k_n}} = -\mathbf{R}_{c^{k_n}}^{b^{-1}}\mathbf{t}_{c^{k_n}}^b + \mathbf{R}_{c^{k_n}}^{b^{-1}}\left(-\mathbf{R}_{b_j}^{w^{-1}}\mathbf{t}_{b_j}^w + \mathbf{R}_{b_j}^{w^{-1}}(\mathbf{t}_{b_i}^w + \mathbf{R}_{b_i}^w(\mathbf{t}_{c^{k_m}}^b + \mathbf{R}_{c^{k_m}}^b\frac{1}{\lambda_l}\cdot\begin{bmatrix}x_l^{c_i^{k_m}} \\ y_l^{c_i^{k_m}} \\ z_l^{c_i^{k_m}}\end{bmatrix}))\right), \tag{6.17}$$

Especially, as for the feature of monocular camera, the first captured camera is the captured camera $m = n$. Thus the camera measurements for the $l^{th}$ feature in the $j^{th}$ frame captured by the $k^{th}$ camera is:

$$\mathcal{P}_l^{c_j^k} = -\mathbf{R}_{c^k}^{b^{-1}}\mathbf{t}_{c^k}^b + \mathbf{R}_{c^k}^{b^{-1}}\left(-\mathbf{R}_{b_j}^{w^{-1}}\mathbf{t}_{b_j}^w + \mathbf{R}_{b_j}^{w^{-1}}(\mathbf{t}_{b_i}^w + \mathbf{R}_{b_i}^w(\mathbf{t}_{c^k}^b + \mathbf{R}_{c^k}^b\frac{1}{\lambda_l}\cdot\begin{bmatrix}x_l^{c_i^k} \\ y_l^{c_i^k} \\ z_l^{c_i^k}\end{bmatrix}))\right), \tag{6.18}$$

where $[x_l^{c_i^k}\ y_l^{c_i^k}\ z_l^{c_i^k}\ ]^T$ is the noiseless first observation of the $l^{th}$ feature that happens in the $i^{th}$ image, $k^{th}$ camera. To simply the representation, we omit homogeneous term in above equations. $\mathbf{T}_c^b$ is the transformation from the camera frame to the IMU (body) frame, and its inverse transforms from the IMU frame to the camera frame. $\mathbf{T}_{b_i}^w$ transforms from the $i^{th}$ IMU frame to the world frame. $\pi_c^{-1}(\cdot)$ is the back projection function which outputs

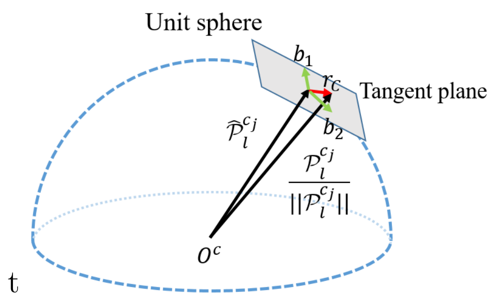

Unit sphere
$b_1$
$r_C$
Tangent plane
$b_2$
$\hat{\mathcal{P}}_l^{c_j}$
$\frac{\mathcal{P}_l^{c_j}}{\|\mathcal{P}_l^{c_j}\|}$
$O^c$
t

Figure 6.5: An illustration of the camera measurement residual on the unit sphere.

the unit vector in 3D space. Since the degree of freedom of the vision residual is two, we project the unit vector residual into the tangent plane, which guarantees the right degree of freedom. $\mathbf{b}_1, \mathbf{b}_2$ are two arbitrarily selected orthogonal bases which span the tangent plane of $\hat{\mathcal{P}}_l^{c_j}$.

### 6.5.4 Marginalization

In order to bound the computational complexity of graph optimization-based methods, marginalization is incorporated. We selectively marginalize out IMU states $\mathbf{x}_k$ and features $\lambda_l$ from the sliding window, meanwhile convert measurements corresponding to the marginalized states into a prior.

In our previous work [73, 69], we use the two-way-marginalization scheme to selectively remove recent or old states based on the scene parallax test. A new frame is added to the sliding window if it is a keyframe, and the oldest frame states are marginalized out with its corresponding measurement. Otherwise, if a non-keyframe comes, we marginalize out the second newest frame. This marginalization scheme can keep spatial keyframes in the window, meanwhile bound the uncertainty for pre-integrated IMU measurements.

However, one drawback of our previous strategy is that marginalizing out the second newest frame generates a dense prior, which will destroy the sparsity of the system, increasing the computational complexity. To avoid this, we slightly modify the strategy on marginalizing the second newest frame. When a non-keyframe comes, instead of marginalizing out all measurement, we throw the visual measurements and keep the IMU measurements corresponding to the second newest frame in the window. Although we drop some visual measurements, these measurements are not significant since they correspond to small motion which do not affect the visual structure. One potential drawback is that we extend the time interval of pre-integration which will induce large covariance into

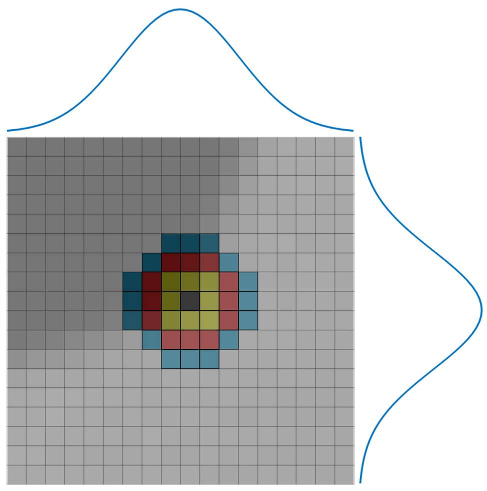


Figure 6.6: An illustration of the feature measurements.

the sliding window. However, this strategy maintains the sparsity of the system, making it efficient and work well on the a computation-limited platform.

We construct a new prior based on marginalized measurements related to the removed state. The marginalization is carried out using the Schur complement. Intuitively, by marginalization, important information of the removed states is kept and computation complexity is bounded.

## 6.6 Parameter Selection

In the estimation, we process the sensors as Gaussian process.

### 6.6.1 Vision Measurement Variance

The vision measurements are provide by the sparse feature detector and the feature tracking in pixel coordinate. The vision measurements contains the additive noise errors that fluctuates, as "write noise". The vision measurement $\mathbf{u}_m$ is therefore written as:

$$\mathbf{u}_m = \mathbf{u} + n(\mathbf{u}), \tag{6.19}$$

where the $\mathbf{u} = [u, v]^T$ is the feature in 2D pixel coordinate. The $n(\mathbf{u})$ is the "write noise" due to the performance of the feature front-end system.

Assume the noise of the vision measurement is Gaussian, the noise is modelled with a zero-mean, independent, continuous-time white Gaussian noise process $n(\mathbf{u})$ of strength

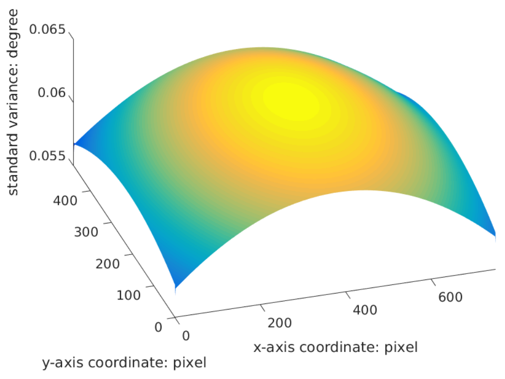
$\sigma_{\mathbf{u}}$:

$$E[n(\mathbf{u})] \equiv [0,0]^T,$$
$$\sigma_{\mathbf{u}} = \begin{bmatrix} \sigma_u & 0 \\ 0 & \sigma_u \end{bmatrix} \tag{6.20}$$

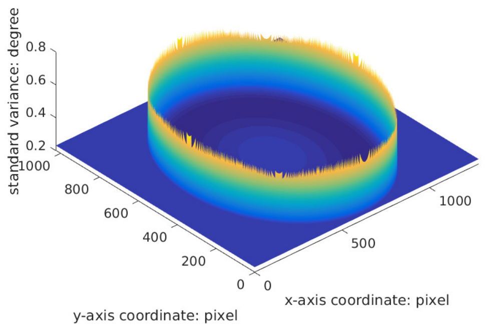
where $\sigma_u$ is the variance in the pixel coordinate.

However, in our estimation system, the vision measurements are transformed from the pixel coordinate to the bearing vector and the residuals are in tangent space. Thus the variance of vision measurements should be described as noise error angle so that satisfy the bearing vector and residual in tangent space.

After camera intrinsic calibration, we get the relationship between the bearing vectors and the pixels. The corresponding noise error angle per pixel can be calculated via enumerating all of the pixels. A map will be built for each camera which describes the relationship between the pixel domain and the error angle domain. The sample of the maps are shown in Fig. 6.7. During the nonlinear optimization, the noise of visual measurements will be set in the pixel domain at first, then replaced with the noise error angle by look-up-table.



(a) a perspective camera



(b) a fisheye camera

Figure 6.7: The relationship map between the pixel domain and the error angle domain.

### 6.6.2 IMU Variance

The variance of the IMU data is calculated, analyzed and modeled using the Allan variance [15]. The noise of IMU always modeled as quantization noise, angle random walk, bias instability, rate random walk, and rate ramp, described by the Allan variance.

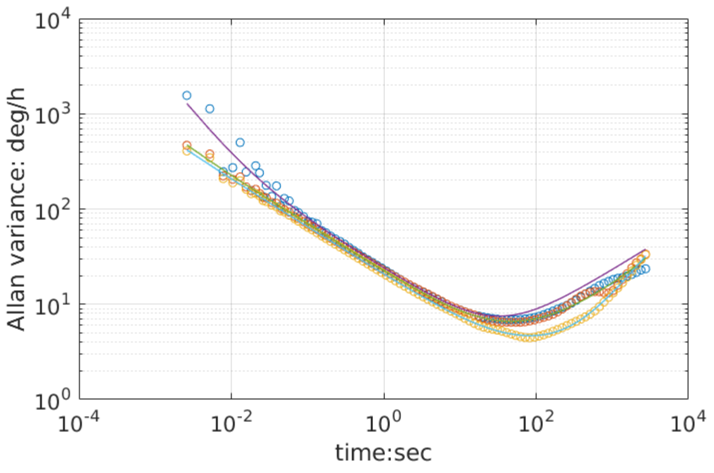


Figure 6.8: The Allan variance of a IMU.

For a series of IMU measurements $\Omega(t)$ with time period $\tau$ and total $N$ consecutive data, a group of $n$ consecutive data points (with $n < N/2$ ) are formed as a cluster. The cluster average is defined as

$$\bar{\Omega}(T) = \frac{1}{n} \sum_{j=i}^{i+n-1} \Omega(t), \tag{6.21}$$

where associated with each cluster is a time $T$ , which is equal to $n\tau$. The Allan variance is defined as

$$\sigma_A^2(T) = \frac{1}{2(N-2n)} \sum_{k=1}^{N-2n} (\bar{\Omega}_{k+n} - \bar{\Omega}_k)^2. \tag{6.22}$$

The variance parameters are gotten from the Allan variance [15].

The source code of the Allan variance calculation tool is open sourced at: https://github.com/gaowenliang/imu_utils.

## 6.7 IMU Propagation for Feedback Control

For the visual-inertial system, the frequency of the IMU measurements (400Hz) is much higher than the visual feature measurements (10 Hz). Consider the time cost of feature tracking and nonlinear optimization, the moment of the newest optimized keyframe is almost 100ms later than the current IMU frame. To satisfy the controller performance for real-time control, the optimized odometry of the estimator are directly propagated with the IMU measurements until current now, which serves as the high-frequency feedback in the control loop.

# CHAPTER 7

# DUAL-FISHEYE OMNIDIRECTIONAL DENSE MAPPING

As the core of the autonomous quadrotor navigation, the mapping module provides a perception to reconstruct the surrounding environment. This reconstruction is the backbone for safe and efficient navigation around obstacles.

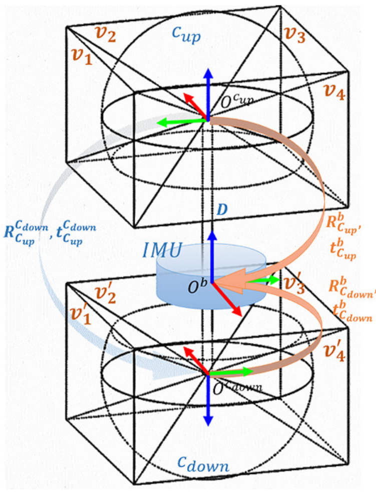
To achieve the best coverage with the fewest number of cameras, we proposed a stereo setup with two oppositely installed fisheye camera, provide an omnidirectional stereo view. In this section, we will introduce the model of the omnidirectional spatial stereo. This is followed by semi-global smoothing to for outlier removal and depth propagation on texture-less environments. We utilize the GPUs onboard the NVIDIA TX2 and with the help of CUDA[1], the system is able to provide omnidirectional depth images in real time. Local depth images are fused using truncated signed distance field (TSDF) to provide a global map that is directly used for trajectory planning.

## 7.1 Related Work

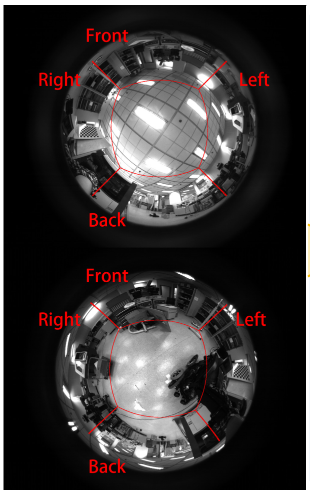
Extensive scholarly work exists in the field of dense mapping, and a large variety of sensors, such as RGB-D cameras, stereo cameras, and monocular cameras are used. In an RGB-D camera setting, KinectFusion [56] simultaneously solves the localization and mapping problem. It uses iterative closest point (ICP) to solve the camera pose, then reconstructs a global map by TSDF fusion. Autonomous flight using an RGB-D sensor is proposed in [58, 59]. With a known mesh map constructed by a Kinect, the authors use direct semi-dense tracking for pose estimation. However, RGB-D sensors have its own limitation, which severely limits their usage in outdoor environments.

Spatial stereo camera is another popular sensing configuration. A semi-global matching stereo method has been presented in [29]. By combining pixelwise cost and smoothness constraints together, the author proposes a global energy function to solve the disparity

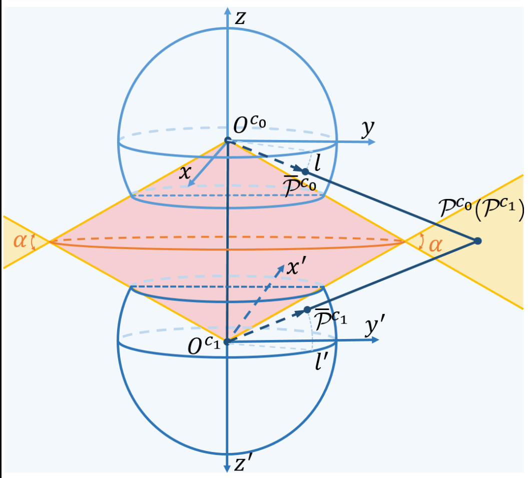
[1]https://developer.nvidia.com/cuda-toolkit


$v_1$
$v_2$
$c_{up}$
$v_3$
$v_4$
$O^{c_{up}}$
$D$
$IMU$
$O^b$
$R^{c_{down}}_{c_{up}}, t^{c_{down}}_{c_{up}}$
$R^b_{c_{up}}, t^b_{c_{up}}$
$R^b_{c_{down}}, t^b_{c_{down}}$
$v'_1$
$v'_2$
$v'_3$
$v'_4$
$O^{c_{down}}$
$c_{down}$

(a) The sensor configuration



(b) ROI extraction


$z$
$O^{c_0}$
$y$
$x$
$l$
$\overline{\mathcal{P}}^{c_0}$
$\mathcal{P}^{c_0}(\mathcal{P}^{c_1})$
$\alpha$
$\alpha$
$x'$
$\overline{\mathcal{P}}^{c_1}$
$y'$
$O^{c_1}$
$l'$
$z'$

(c) The omnidirectional stereo configuration

Figure 7.1: Illustration of dual-fisheye omnidirectional stereo.

map and optimizes its approximated version by dynamic programming. The stereo matching cost can be further improved using machine learning-based approaches, as proposed in [74]. Spatial stereo can easily suffer the calibration issues as it conditions on known camera extrinsic parameters. Moreover, the baseline requirement of stereo-based methods fundamentally limits its capability for small-scale platforms.

## 7.2 Dual-Fisheye Omnidirectional Stereo

The omnidirectional stereo camera system is constructed with dual oppositely installed ultra-wide FOV fisheye cameras, one upward-facing, and the other downward-facing, and a fixed baseline, as shown in Fig. 8.4(a). The overlapping field of view between the two fisheye cameras naturally forms a ring-shaped omnidirectional spatial stereo setup, as indicated in Fig. 7.1(c). The yellow area represents the 65-degree overlap. The red area is the blind spot of the stereo. The blue area is the field of monocular view. Using fisheye cameras with a $\gamma$ FOV, we obtain FOV around $(\gamma - 180)$-degree vertical and 360-degree horizontal.

For each fisheye camera, four different $\mathbf{R}^c_v$ and $\pi_v^{-1}(\cdot)$ are used in order to crop and rectify the original image to four direction undistorted images, as shown in Fig. 7.1(a) and Fig. 7.1(b). The red distorted lines demarcate the ROI of the upward-facing and downward-facing images.

The back projection functions of these two cameras $c_0$ and $c_1$ are $\pi_{c_0}^{-1}(\cdot)$ and $\pi_{c_1}^{-1}(\cdot)$.

Suppose a 3D point $j$ is observed both in the upward-facing camera and downward-facing camera. Its coordinates in each frame are $\mathcal{P}_j^{c_0}$ and $\mathcal{P}_j^{c_1}$. The projections on the image frame are $\mathbf{u}_j^{c_0}$ and $\mathbf{u}_j^{c_1}$, and the corresponding depths are $\lambda_j^{c_0}$ and $\lambda_j^{c_1}$ respectively. With the camera model, we denote the normalized pixel coordinates of $\mathcal{P}_j^{c_0}$ and $\mathcal{P}_j^{c_1}$ as

$$\bar{\mathcal{P}}_j^{c_0} = \pi_{c_0}^{-1}(\mathbf{u}_j^{c_0}), \quad \bar{\mathcal{P}}_j^{c_1} = \pi_{c_1}^{-1}(\mathbf{u}_j^{c_1}), \tag{7.1}$$

where $\mathcal{P}_j^{c_0} = \lambda_j^{c_0}\bar{\mathcal{P}}_j^{c_0}$ and $\mathcal{P}_j^{c_1} = \lambda_j^{c_1}\bar{\mathcal{P}}_j^{c_1}$. The frame of the upward-facing and downward-facing cameras are related to the extrinsic parameters

$$\mathcal{P}_j^{c_1} = \mathbf{R}\mathcal{P}_j^{c_0} + \mathbf{t}, \tag{7.2}$$

where the $\mathbf{R} \in \mathbb{R}^3$ is the rotation in the downward-facing camera frame, and $\mathbf{t}$ is the translation $O^{c_1}O^{c_0}$ in the downward-facing camera frame. The essential matrix $\mathbf{E}$ is defined as

$$\mathbf{E} = \lfloor \mathbf{t}\times \rfloor \mathbf{R}, \tag{7.3}$$

where the $\lfloor \cdot\times \rfloor$ is a skew-symmetric function which transforms a 3D vector into a $3 \times 3$ skew-symmetric matrix.

According to (10.2), with ROI extraction, (7.1) can be formulated as

$$\bar{\mathcal{P}}_j^{c_0} = \mathbf{R}_k^{c_0}\pi_k^{-1}(\mathbf{u}_j^{k}), \quad \bar{\mathcal{P}}_j^{c_1} = \mathbf{R}_{k'}^{c_1}\pi_{k'}^{-1}(\mathbf{u}_j^{k'}), \tag{7.4}$$

where $\mathbf{u}_j^k$ and $\mathbf{u}_j^{k'}$ are the image coordinates of the virtual camera $k$ and $k'$ towards the same direction as camera $c_0$ and $c_1$. (7.2) can be formulated as

$$\lambda_j^{c_0}\left[\mathbf{R}_k^{c_0}\pi_k^{-1}(\mathbf{u}_j^{k})\right] = \lambda_j^{c_1}\mathbf{R}\left[\mathbf{R}_{k'}^{c_1}\pi_{k'}^{-1}(\mathbf{u}_j^{k'})\right] + \mathbf{t}. \tag{7.5}$$

For ROI extraction of the downward-facing camera, using $\mathbf{R}\mathbf{R}_{k'}^{c_1}$ instead of $\mathbf{R}_{k'}^{c_1}$ in (10.2), which will rectify each vertical stereo camera. Thus, the traditional stereo matching approach is satisfied with eight cropped images taken from four direction virtual stereo systems.

### 7.2.1 Extrinsic Self-Calibration of Omnidirectional Stereo

To achieve extrinsic self-calibration from the epipolar constraint, an optimizable error is formulated with the stereo extrinsic parameters. Our proposed method is based on our previous work [45]. For points on the line $O^{c_0}\mathcal{P}^{c_0}$, the epipolar line is

$$l_j = \mathbf{E}\bar{\mathcal{P}}_j^{c_0}. \tag{7.6}$$

The epipolar constraint is

$$(\bar{\mathcal{P}}_j^{c_1})^T l_j = (\bar{\mathcal{P}}_j^{c_1})^T \mathbf{E} \bar{\mathcal{P}}_j^{c_0} = 0, \tag{7.7}$$

which is satisfied with scaling factor $\mathbf{t}$ multiplies. The epipolar error of each pair of matched feature points $\bar{\mathcal{P}}_j^{c_0}$ and $\bar{\mathcal{P}}_j^{c_1}$ is defined as

$$||(\bar{\mathcal{P}}_j^{c_1})^T \mathbf{E} \bar{\mathcal{P}}_j^{c_0}||^2, \tag{7.8}$$

which is the square of the distance between point $\bar{\mathcal{P}}_j^{c_0}$ and epipolar line $l$ in view of the epipolar geometry. Since the observation of detected feature points is corrupted with noise and matching error exists, the epipolar constraint cannot be satisfied perfectly. Our approach is to solve for $\mathbf{R}$ and $\mathbf{t}$ on the manifold that minimizes the overall epipolar error for all the matched feature correspondences of each virtual camera. Detailed treatment of the self-calibration approach can be found in [45]. According to (7.4), the total process is

$$\begin{aligned} &\min_{\mathbf{R},\mathbf{t}} \sum_k \sum_j \left\| \left[\mathbf{R}_{k'}^{c_1} \pi_{k'}^{-1}(\mathbf{u}'_j)\right]^T \lfloor \mathbf{t} \times \rfloor \mathbf{R} \left[\mathbf{R}_k^{c_0} \pi_k^{-1}(\mathbf{u}_j)\right] \right\|^2, \\ &s.t. \; ||\mathbf{t}|| = 1 \end{aligned} \tag{7.9}$$

where $\mathbf{E}$ is substituted by $\lfloor \mathbf{t} \times \rfloor \mathbf{R}$ using equation (7.3), $\mathbf{u}_j$ and $\mathbf{u}'_j$ are the corresponding features. The unit norm constraint $||\mathbf{t}|| = 1$ refines the stereo calibration as 5-DOF optimization, three for rotation $\mathbf{R} \in SO(3)$ and two for translation $\mathbf{t} \in \mathbb{S}^2$ [45].

### 7.2.2 Semi-Global Optimization

Temporal cost aggregation aims to decrease the sensitivity of image noise by using multiple measurements. But still, the simple winner-takes-all strategy will not get reliable results since mismatch can easily occurred due to repetitive patterns. In addition, in texture-less regions, the cost volume will contain similar cost among different depths, which leads ambiguity in depth estimation. To this end, a smoothing-based optimization method is adopted in our depth estimator.

Since we use enumerated planes to represent the depth image, the depth optimization problem now becomes a plane index optimization problem. Denote $\boldsymbol{\mathcal{K}}$ as the plane index map that we try to optimize. The global energy function $E(\boldsymbol{\mathcal{K}})$ which combines the pixelwise cost and smoothness constraints is expressed as:

$$E(\boldsymbol{\mathcal{K}}) = \sum_{\mathbf{u}\in\Omega}(E_{SAD}(\mathbf{u}, \boldsymbol{\mathcal{K}}_{\mathbf{u}}) + P_1 \cdot \sum_{\mathbf{u}_q\in\mathcal{N}(\mathbf{u})} T[|\boldsymbol{\mathcal{K}}_{\mathbf{u}} - \boldsymbol{\mathcal{K}}_{\mathbf{u}_q}| = 1] + P_2 \cdot \sum_{\mathbf{u}_q\in\mathcal{N}(\mathbf{u})} T[|\boldsymbol{\mathcal{K}}_{\mathbf{u}} - \boldsymbol{\mathcal{K}}_{\mathbf{u}_q}| > 1]), \tag{7.10}$$

where $\Omega$ is the keyframe image domain and $\boldsymbol{\mathcal{K}}_{\mathbf{u}}$ is the enumerated plane index at pixel $\mathbf{u}$. $T[\cdot]$ is the indicator function which returns 1 if the expression inside is true, and returns 0 otherwise. The energy function for each pixel consists of three kinds of terms: one data term directly extracted from the cost volume and two regularization terms. The first regularization term $P_1$ penalizes the energy by neighbor pixels with which the enumerated depth difference is only 1. The second regularization term gives higher penalty $P_2$ penalty to larger depth differences..

Since the global minimization is a NP-complete problem which can not be solved in polynomial time, a semi-global matching (SGM) is proposed in [29]. In this work, we use 4-path SGM as it gives a good trade-off between speed and accuracy.

$$S(\boldsymbol{\mathcal{K}}) = \sum_{\mathbf{u}\in\Omega}\sum_{\mathbf{r}} L_{\mathbf{r}}(\mathbf{u}, \boldsymbol{\mathcal{K}}_{\mathbf{u}}), \mathbf{r} \in \left\{[0\ 1]^T, [1\ 0]^T, [0\ -1]^T, [-1\ 0]^T\right\} \tag{7.11}$$

$$L_{\mathbf{r}}(\mathbf{u}, \boldsymbol{\mathcal{K}}_{\mathbf{u}}) = E_{SAD}(\mathbf{u}, \boldsymbol{\mathcal{K}}_{\mathbf{u}}) + \min \begin{Bmatrix} L_{\mathbf{r}}(\mathbf{u}-\mathbf{r}, \boldsymbol{\mathcal{K}}_{\mathbf{u}}) \\ L_{\mathbf{r}}(\mathbf{u}-\mathbf{r}, \boldsymbol{\mathcal{K}}_{\mathbf{u}}-1) + P_1 \\ L_{\mathbf{r}}(\mathbf{u}-\mathbf{r}, \boldsymbol{\mathcal{K}}_{\mathbf{u}}+1) + P_1 \\ \min_i \{L_{\mathbf{r}}(\mathbf{u}-\mathbf{r}, i) + P_2\} \end{Bmatrix} - \min_j L_{\mathbf{r}}(\mathbf{u}-\mathbf{r}, j), \tag{7.12}$$

where the last term $\min_j L_{\mathbf{r}}(\mathbf{u}-\mathbf{r}, j)$ is used to reduce the numerical value since it is smaller than the sum of first two terms and it is the same if $\mathbf{u}$ and $\mathbf{r}$ does not change which guarantees the $L_{\mathbf{r}}(\mathbf{u}, \boldsymbol{\mathcal{K}}_{\mathbf{u}})$ would not be affected. Hence, the problem is simplified as a 1D problem along the direction $\mathbf{r}$. The approximated cost function can be solved in parallel by using GPU.

Usually, a pixel with higher gradient gains higher probability of the depth discontinuity. To support this edge-aware smoothing, we adjust the penalities $P_1$ and $P_2$ by means of the pixel gradient $G$ along the SGM scaning direction:

$$P_1 = \begin{cases} P_1' \cdot Q_1 & \text{if } G <= G_\tau \\ P_1' & \text{otherwise} \end{cases} \qquad P_2 = \begin{cases} P_2' \cdot Q_2 & \text{if } G <= G_\tau \\ P_2' & \text{otherwise} \end{cases}, \tag{7.13}$$

where $G_\tau$ is a certain gradient threshold, and $P_1'$, $P_2'$, $Q_1$, $Q_2$ are contant parameters. Note that $Q_1$ and $Q_2$ should be set greater than 1.

### 7.2.3 Post-Processing of Depth Images

After semi-global optimization, simple winer-takes-all strategy can be used to obtain depth by choosing the lowest cost $S(\boldsymbol{\mathcal{K}})$ at each pixel. However, since depth is sampled at a discrete setting, we use parabola interpolation to obtain a better depth, which is the extrema point between two depth samples.

$$f(\boldsymbol{\mathcal{K}}_{\mathbf{u}}) = S(\mathbf{u}, \boldsymbol{\mathcal{K}}_{\mathbf{u}}) = a\boldsymbol{\mathcal{K}}_{\mathbf{u}}^2 + b\boldsymbol{\mathcal{K}}_{\mathbf{u}} + c \tag{7.14}$$

$$f(\boldsymbol{\mathcal{K}}_{\mathbf{u}} - 1) = S(\mathbf{u}, \boldsymbol{\mathcal{K}}_{\mathbf{u}} - 1) = a(\boldsymbol{\mathcal{K}}_{\mathbf{u}} - 1)^2 + b(\boldsymbol{\mathcal{K}}_{\mathbf{u}} - 1) + c \tag{7.15}$$

$$f(\boldsymbol{\mathcal{K}}_{\mathbf{u}} + 1) = S(\mathbf{u}, \boldsymbol{\mathcal{K}}_{\mathbf{u}} + 1) = a(\boldsymbol{\mathcal{K}}_{\mathbf{u}} + 1)^2 + b(\boldsymbol{\mathcal{K}}_{\mathbf{u}} + 1) + c \tag{7.16}$$

$$\boldsymbol{\mathcal{K}}_{\mathbf{u}}^* = \boldsymbol{\mathcal{K}}_{\mathbf{u}} - \frac{f(\boldsymbol{\mathcal{K}}_{\mathbf{u}} + 1) - f(\boldsymbol{\mathcal{K}}_{\mathbf{u}} - 1)}{2\left(f(\boldsymbol{\mathcal{K}}_{\mathbf{u}} - 1) - f(\boldsymbol{\mathcal{K}}_{\mathbf{u}}) + f(\boldsymbol{\mathcal{K}}_{\mathbf{u}} + 1) - f(\boldsymbol{\mathcal{K}}_{\mathbf{u}})\right)}, \tag{7.17}$$

where $a, b, c$ are the coefficients of the parabola $f(\cdot)$, $\boldsymbol{\mathcal{K}}_{\mathbf{u}}$ is the plane index which reaches the lowest cost at pixel $\mathbf{u}$, $\boldsymbol{\mathcal{K}}_{\mathbf{u}}^*$ is the interpolated plane index at the extreme point. Thus the estimated depth of pixel $\mathbf{u}$ can be calculated by $d_{\mathbf{u}} = \frac{1}{\boldsymbol{\mathcal{K}}_{\mathbf{u}}^* \cdot \lambda_{min}}$.

Two criteria are used to remove the outliers. First, for the pixel with some sampling inverse depth which we can not compute the similarity cost since the corresponding pixel projected in the measurement frame at that inverse depth is unknown. For those pixels $\mathbf{u}$, we mark it as undefined in the 3D cost volume at the corresponding inverse depth and later set $L_{\mathbf{r}}(\mathbf{u}, \boldsymbol{\mathcal{K}}_{\mathbf{u}})$ as zero in the SGM step to remove the influence of the missing cost. And finally those pixels at the output depth image are set as undefined. Secondly, we drop the depth if it is not the absolute minimal of the fitting parabola. Specifically, the minimal cost locating at the $0^{th}$ or the $(L-1)^{th}$ plane is needed to be ignored, in this case, the pixel's depth will be set as undefined. Also, if the sum of the cost of two neighbor planes is smaller than the twice of the minimal cost, the corresponding pixel will be set as undefined as well.

## 7.3 Global Dense Mapping

We fuse all depth images obtained at different camera poses into a global dense map using an uncertainty-aware truncated signed distance field (TSDF) fusion approach. Our method is developed from the open source CHISEL TSDF implementation [37]. Improvements include uncertainty-aware depth fusion (Sect. 7.3.2), and algorithm parallelization (Sect. 7.3.3).

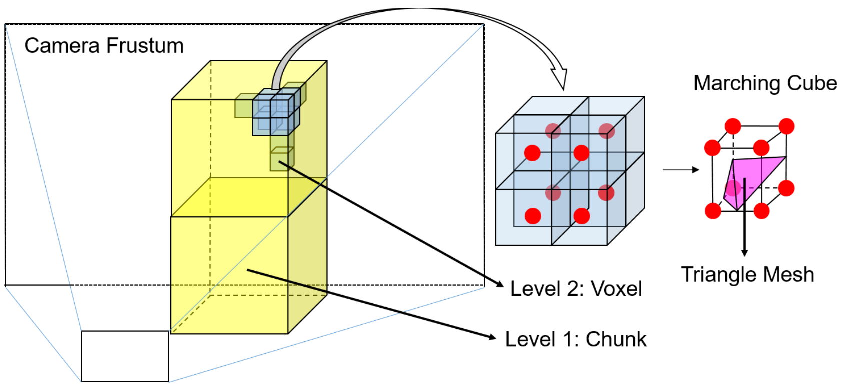


Figure 7.2: A general view of TSDF.

### 7.3.1 Truncated Signed Distance Field (TSDF)

The whole environment can be modeled as a 3D volumetric signed distance field. The signed distance function $\phi(x) : \mathbb{R}^3 \rightarrow \mathbb{R}$ is the distance from voxel $x$ to the its nearest surface, signed positive if the voxel is between the surface and the camera, negative otherwise. Since we are only interesting in reconstructing the surface, we use truncated signed distance field:

$$\phi_\tau(x) = \begin{cases} \phi(x) & \text{if } |\phi(x)| \leq \tau \\ \text{undefined} & \text{otherwise} \end{cases}, \tag{7.18}$$

where $\tau \in \mathbb{R}$ is the truncation distance.

Besides truncated signed distance field, we store two more values at each 3D voxel. $C(x) : \mathbb{R}^3 \rightarrow \mathbb{R}$ is the photometric intensity which is maintained like $\phi_\tau(x)$. $W(x) : \mathbb{R}^3 \rightarrow \mathbb{R}$ is the weight representing the confidence among measurements.

To achieve better memory efficiency, CHISEL uses a two-level hybrid data structure. The first level chunk is a spatially-hashed map, which consists of a fixed grid of $N_v^3$ voxels. A chunk is allocated dynamically from a growing heap and indexed in a spatial 3D hash map based on their integer coordinates. The second level is the 3D cube voxel with size $\mathcal{L} \times \mathcal{L} \times \mathcal{L}$ where $\mathcal{L}$ is the mapping resolution in metric. The sketch is shown in Fig. 7.2.

Two depth scan fusion schemes are provided in CHISEL. Since we use the virtual pinhole camera in depth image estimation, projection mapping based on the camera frustum gives better efficiency than raycasting. The camera frustum is an axis-aligned bounding box which is depended on the camera intrinsic and the nearest plane and farthest plane we tend to fuse. We firstly check each chunk intersected with the camera frustum and allocated the chunk if it is the first time being checked. Then, every voxel in the chunks

Algorithm 1 Voxel projection mapping

```
1: procedure Voxel update
2:     for $v \in V$ do
3:         $z \leftarrow ||v - o||$
4:         $d \leftarrow \text{DepthImage}(\pi(K, v))$
5:         $u \leftarrow d - z$
6:         $\tau \leftarrow \Psi(d)$                                  ▷ Dynamic truncation distance
7:         if $u \in [-\tau, \tau]$ then
8:             $\phi_\tau(v) \leftarrow \frac{W(v)\phi_\tau(v)+\alpha(u)u}{W(v)+\alpha(u)}$
9:             $C_\tau(v) \leftarrow \frac{W(v)C_\tau(v)+\alpha(u)u}{W(v)+\alpha(u)}$
10:            $W(v) \leftarrow W(v) + \alpha(u)$
11:        if $u \in [\tau + \epsilon, d]$ then                    ▷ Space carving
12:            $\phi_\tau(v) \leftarrow$ undefined
13:            $C_\tau(v) \leftarrow$ undefined
14:            $W(v) \leftarrow 0$
```

will be enumerated in the update procedure.

In projection mapping method, each voxel would be projected onto the camera plane and compare with the depth value from the depth image. We denote the difference as $e_x \in \mathbb{R}^+$, only difference within the truncation distance $[-\tau, \tau]$ would be updated, otherwise it would be carved. The voxel update algorithm is detailed in Alg. 1.

### 7.3.2 Uncertainty-Aware Fusion

Since long range visual depth estimation usually comes with larger depth uncertainty, we encode the uncertainty by using the dynamic truncation distance $\Psi(d)$. Define $\sigma_\lambda$ as the standard deviation of the depth measurement in the inverse depth $\lambda$, then the standard deviation of the depth measurement in depth $d$ can be approximated using linearization:

$$\sigma_d = \sqrt{(\frac{\partial d}{\partial \lambda})^2 \sigma_\lambda^2}, \tag{7.19}$$

where $\sigma_\lambda$ is set to be $\lambda_{min}$, which is the minimal value in the inverse depth field. Thus, the truncation function is

$$\Psi(d) = \sigma_d \cdot S_{truncation}, \tag{7.20}$$

where $S_{truncation}$ is some constant truncation scale. Consequently, the truncation distance is dynamic and it is able to increase as the depth observation becomes larger.

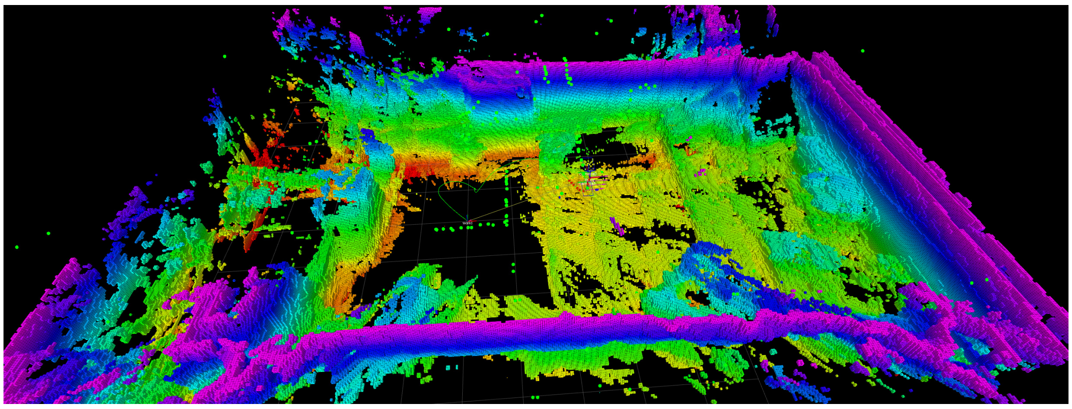


Figure 7.3: A reconstructed grid map with 0.05$m$ resolution.

### 7.3.3 Parallelization

Note that the whole TSDF fusion computation complexity is approximately equal to the voxel update complexity, which is proportional to the voxel count. The update procedure for each voxel is independent, which allows us to accelerate the process by multi-threading. We also decouple the map updating and publishing processes in order to guarantee the availability of the most recent map for trajectory planning. Map update runs as soon as a depth image (can be from either of the two ROI) comes, while the map publisher output the map to the trajectory planning at a constant frequency of 10 Hz. We use mutex lock to guarantee the shared memory would not be conflicted.

### 7.3.4 Map Rendering for Visualization

Although a visually appealing map is not necessary for autonomous navigation, we still provide a map rendering module that runs offline for visualization purpose. The Marching Cube algorithm [47] is used for map rendering. For every eight voxels, we take their centers to form a cube and the signed distance field (SDF) value of each center point is approximately equal to its voxel's SDF value. Consider the sign of each SDF value, the whole cube has 256 different possible constitutions which leads 256 different ways to separate a cube into several triangle meshes (see Fig. 7.2). A sample reconstruction result is visualized in Fig. 7.4.

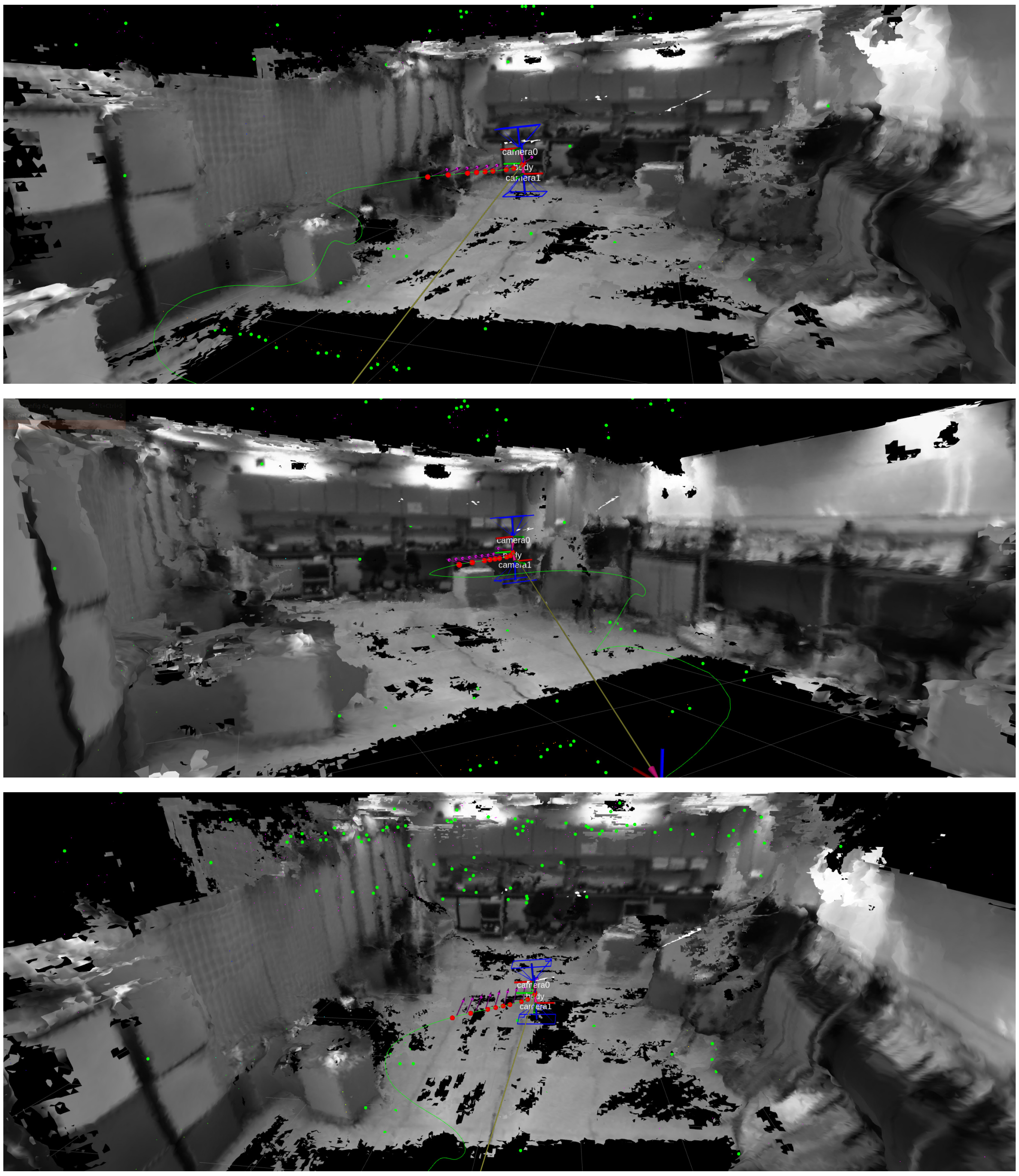


Figure 7.4: A reconstructed mesh map.

# CHAPTER 8

# EXPERIMENTAL RESULTS

In this chapter, we show the experiment results of our system. The first experiment compares the state estimation with a motion capture system namely OptiTrack [1]. The rest of the experiments tests the whole autonomous flight system in both indoor and outdoor environments. Both trajectories and the 3D reconstruction results are presented. More details can be found in our experiment videos.

## 8.1 Feedback Control Using Dual-Fisheye VINS

In this experiment, we test the performance of autonomous trajectory tracking with VINS in aggressive motion. The quadrotor is commanded to track a figure eight pattern with each circle being 1.0 meters in radius, as shown in Fig. 8.1. The quadrotor follow this trajectory six times continuously during experiment. The 400 Hz state estimation results achieve the real-time feedback to control the quadrotor to follow the pre-designed aggressive trajectory, as we mentioned in Sect. 6.7.

The robustness and accuracy are of vital importance to this real-time onboard experiment. The linear velocity reaches 2.3 m/s in this experiment, and the maximum tails of quadrotor are more than 25 degrees during flying. The final drift is 0.35 m, relative to the total 75.3 m path length, by comparing our proposed real-time estimation with OptiTrack. The final percentage of drift is 0.46%. The details of the translation and rotation as well as their corresponding error are shown in Fig. 8.2.

[1]http://optitrack.com/

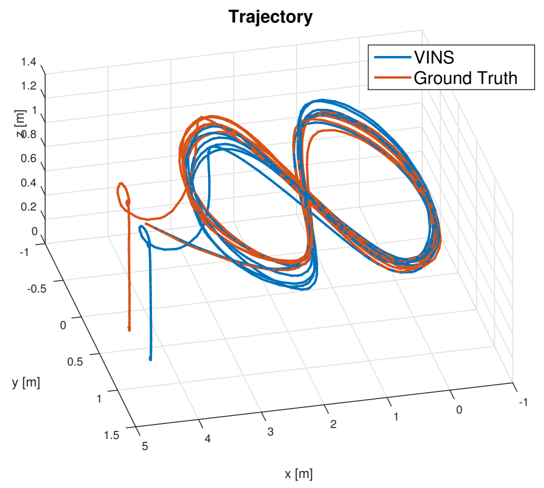


Figure 8.1: The trajectory compared VINS outputs with the motion capture.

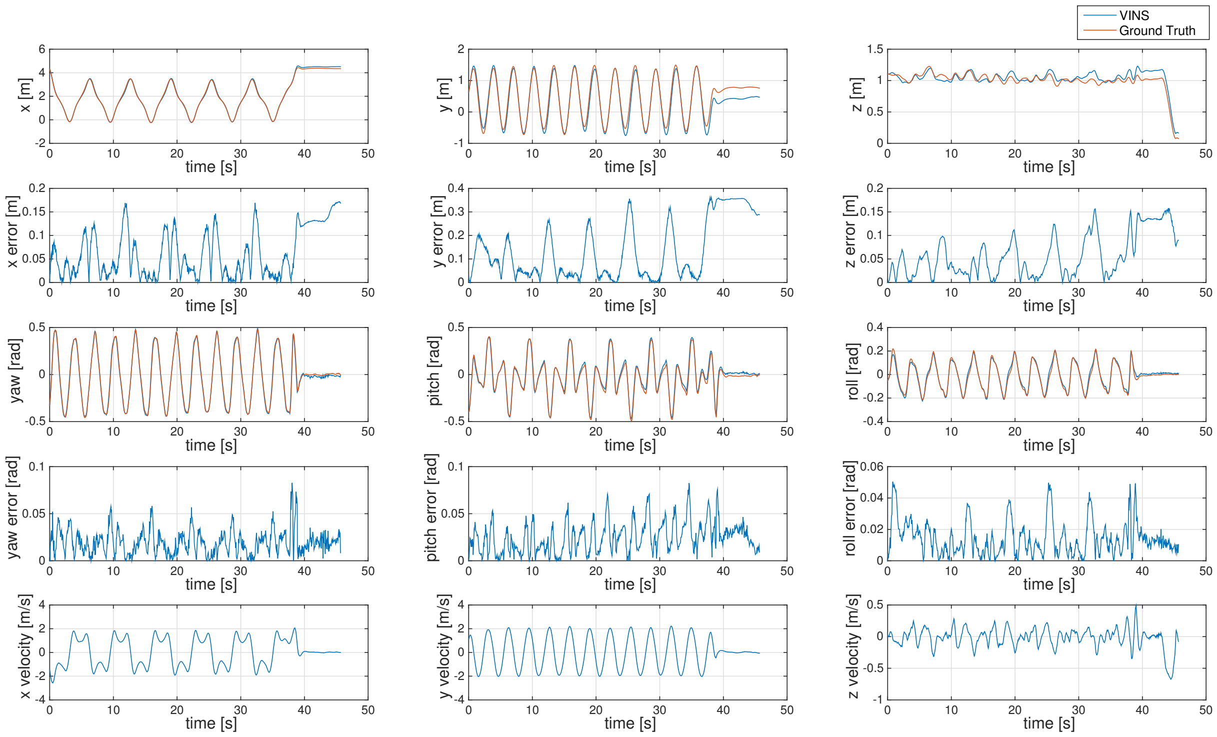


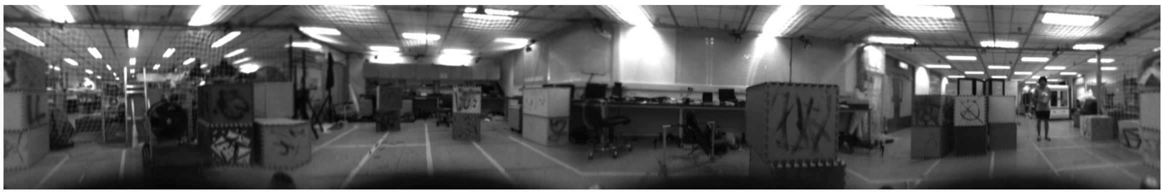
Figure 8.2: The results compared VINS outputs with the motion capture. The top four rows are translation and orientation values and their corresponding error. The bottom row is the velocity plot.

## 8.2 Dense Depth Estimation

Instead of directly evaluating the precision of the dense 3D reconstruction, we do comparison on depth images which mainly affect the performance of the reconstruction.

We first show the raw images, the corresponding omnidirectional ROI images, disparity map and the point loud in one capture in Fig. 8.4. Fig. 8.4(a) shows the dual-fisheye omnidirectional stereo system, only construct with two fisheye images with 245-degree FoV; Fig. 8.4(b) shows the original grayscale fisheye images with $1280 \times 960$ resolution; Fig. 9.19(a) shows 4 rectified pinhole images of the overlapping views, only images from upward camera is shown; Fig. 9.19(d) shows 4 disparity images obtained by semi-global block matching (SGBM) algorithm. Fig. 8.4(e) shows the omnidirectional pointcloud obtained from the disparity images. Floor and ceiling are removed for easy viewing.

The omnidirectional ROI images and the corresponding depth images are shown in Fig. 8.3. The depth images are noisy because of the disgusting distortion and the image intensity contrast, via the lens quality.


(a) Rectified ROI images of the upward-facing camera

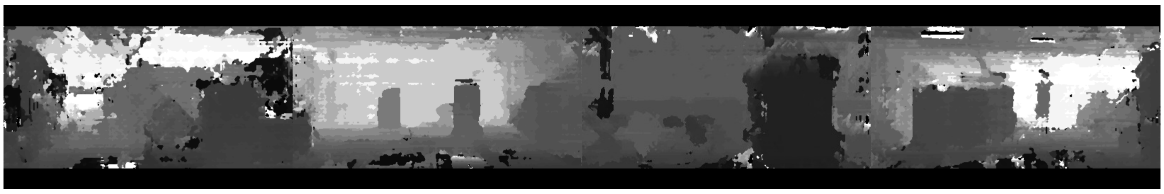
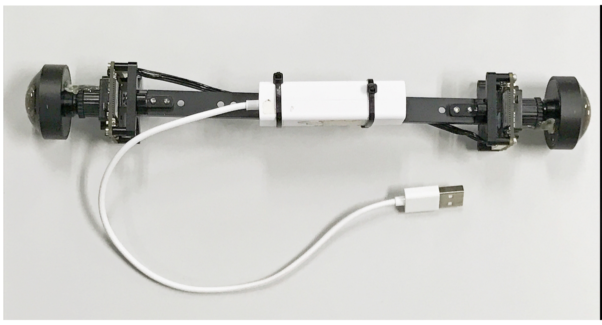
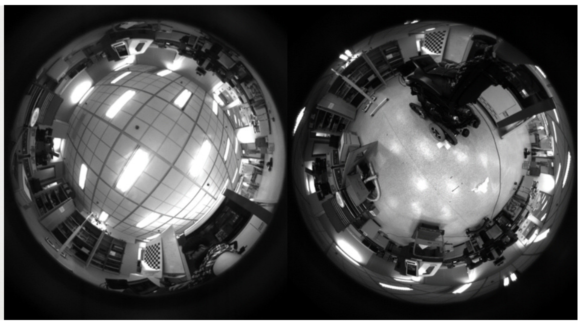
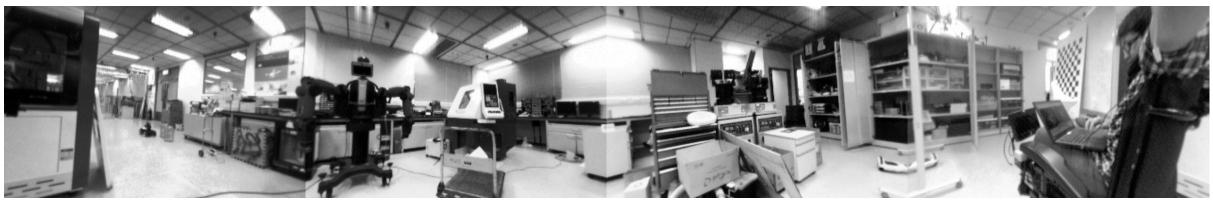

(b) Depth image of the Fig. 8.3(a)

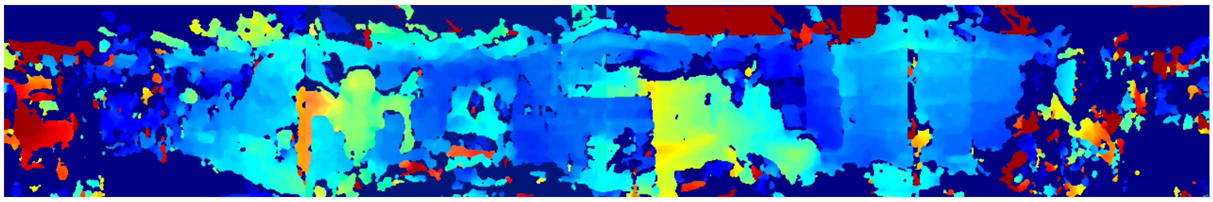
Figure 8.3: Image of indoor experiments.

The disparity map and the point cloud generated with one capture are shown in Fig.8.5: (a) is the real scenes; (b) is the disparity map from the first person view; (c) is the point cloud from the same view of (b); (d) is the point cloud from the top-down view, shows the whole omnidirectional environments; (e) is the point cloud from Bird's-eye view of different capture; (f) is the point cloud of the same scene by 3D LiDAR, as the reference. In the Fig.8.5, “A”, “B” and “C” figure out the corresponding objects.


(a) Omnidirectional stereo system


(b) Raw images


(c) ROI images of upward camera


(d) Disparity images

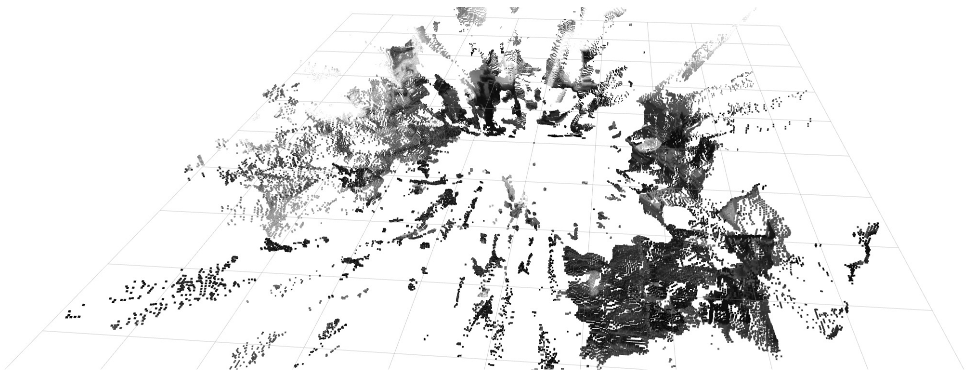

(e) Pointcloud

Figure 8.4: Results of the omnidirectional stereo.

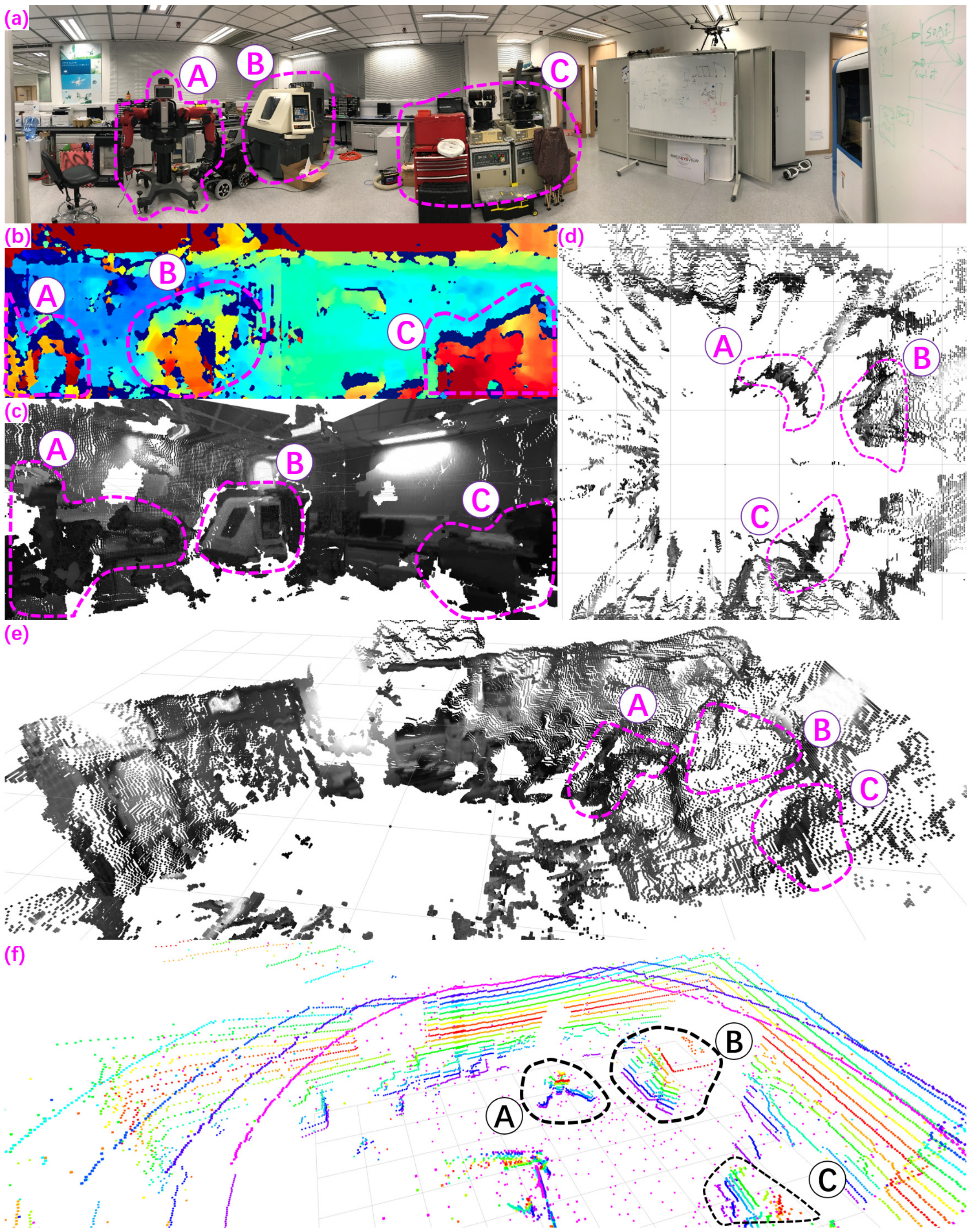
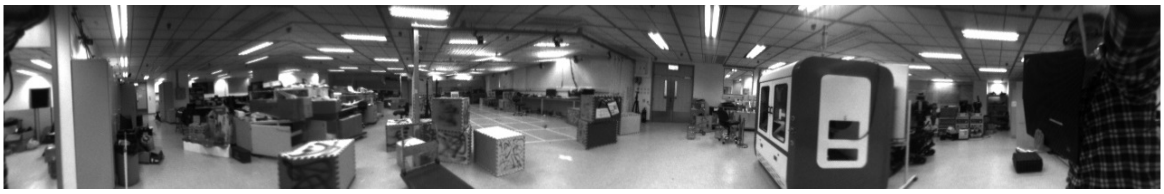
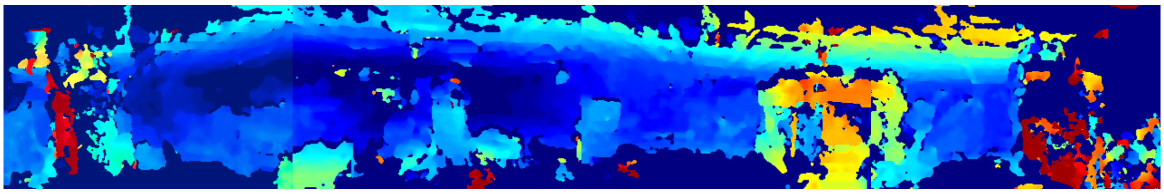


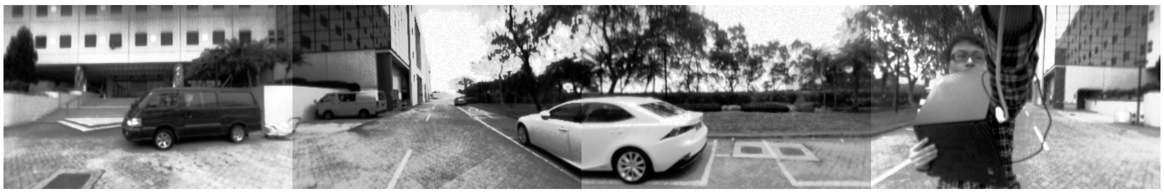
Figure 8.5: Stereo reconstruction result using a pair of image from a single capture.



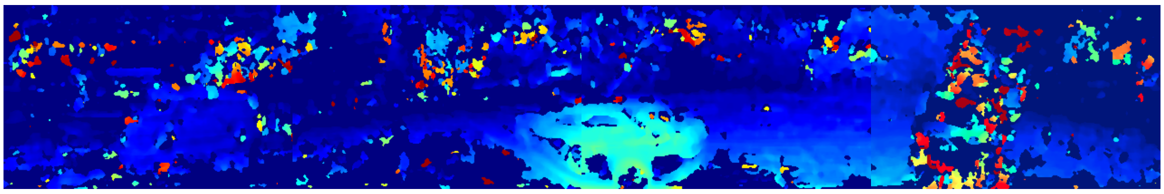
(a) ROI images of laboratory envieonment



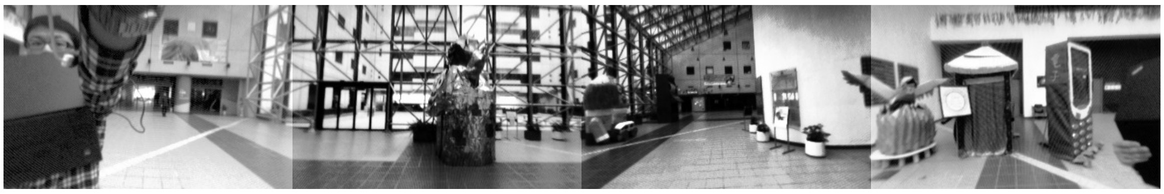
(b) Disparity images of laboratory envieonment



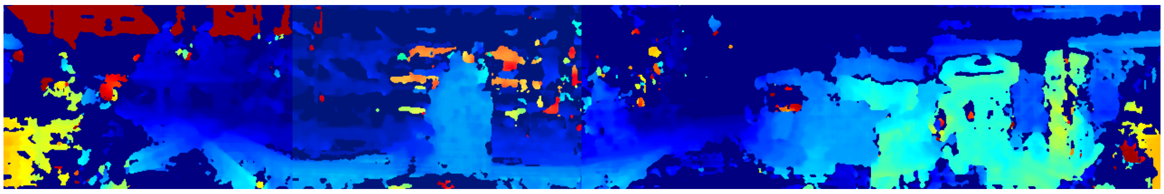
(c) Disparity images of outdoor envieonment



(d) Disparity images of outdoor envieonment



(e) Disparity images of hall envieonment



(f) Disparity images of hall envieonment

Figure 8.6: Indoor and outdoor experiments using SGBM for stereo matching.

The fused global map is shown in Fig. 7.4. The mesh visualization is rendered offline with 0.05$m$ voxel map resolution. The dense map is not very accurate for particulars reconstruction, still, it is sufficiently dense and satisfies autonomous navigation.

## 8.3 Indoor Experiment

In order to substantiate that our system is able to handle small-scale indoor autonomous navigation in complex environments, we randomly position some obstacles in the experiment area. The nearest distance between obstacles is reached 1 m, as shown in Fig, 8.7. After setting the destinations, the quadrotor flies through the feasible space, pass by while avoiding surrounding obstacles with full autonomy.

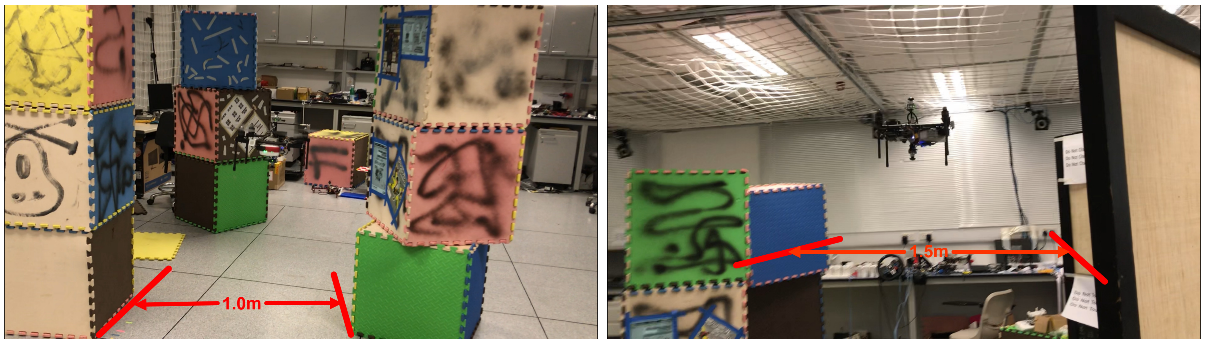
For onboard testing in the indoor environments, a receding horizon replanner based on the local control property of B-spline is used to generate the navigation trajectory in [14]. The local planning parameters are set as follows: the planning time step is 0.35$s$; the maximum velocity and maximum acceleration are set to 1.2$m/s$ and 2.0$m/s$ respectively. The number of features in the VINS module is set as 175, covered the whole spherical perception region. In the mapping module, the voxel map resolution is set as 0.15$m$ so that satisfying the high efficiency because of less mapping details requirement in the motion planning module. The average time consuming of the full system is measured in Table 8.1.

All the modules including localization, mapping, and planning can be launched on the ground and initialize itself. After taking off, the omnidirectional mapping module senses the surrounding environments without any prior knowledge. The quadrotor starts to fly to the destination point following the optimized trajectory autonomously. The generated 3D construct of the surrounding environments is fed into the trajectory planning module via the voxel map in real time so that the trajectories are generated satisfy both safety and smoothy. The perception system keeps sensing and if the newly updated map has a

Table 8.1: Approximated Timing Statistics

| Module | Time Consumption (ms) | Frequency (Hz) |
|---|---|---|
| ROI extraction | 8 | 15 |
| VINS - Front-End | 35 | 15 |
| VINS - Back-End | 55 | 10 |
| SGBM Depth Map | 20 | 10 |
| TSDF Fusion | 40 | 10 |

collision possibility with the executing trajectory, a new safe trajectory will be generated immediately by the replanning system. More details of the trajectory planning and re-plan can be found in [14]. A series of snapshots of the realtime flight experimental results are visualized in Fig. 8.8 and Fig.8.9.

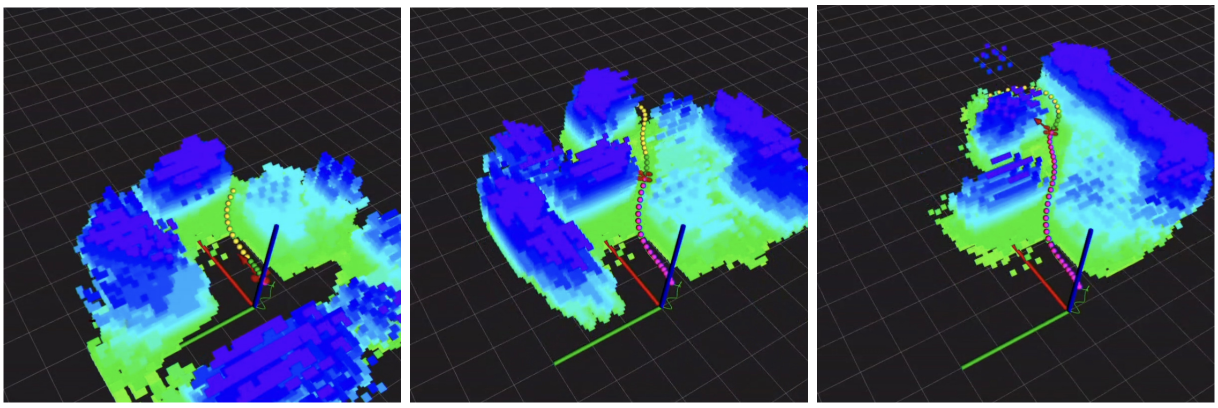


(a) (b)

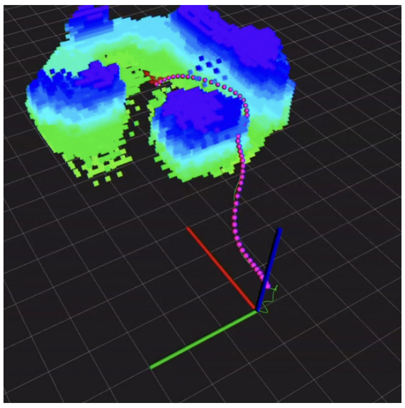
Figure 8.7: The narrow environment in the indoor experiments.

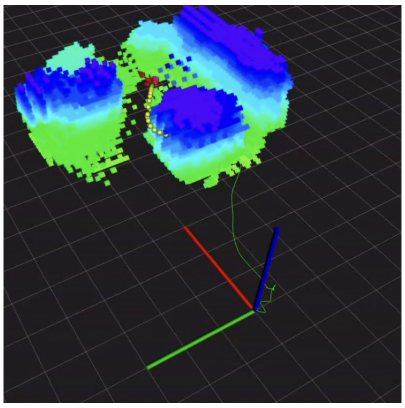
The average speed during this experiment is around $1.1m/s$, while the highest velocity is about $2.2m/s$. For a robust autonomous application, we critically concern the system delay property. As we mentioned in Table 8.1, overall average delay of the perception system is around $180ms$, which is enough for our demonstration of autonomous navigation but still can be improved. The VINS pose estimation runs at 10Hz, however, a faster 400 Hz estimated poses from IMU propagation can be used to feedback control which will gain less delay. Without considering the VINS pose estimation, the controller earns a real-time feedback. The whole system delay depends on the mapping module, since calculating four depth images and fuse them will cost a lot.

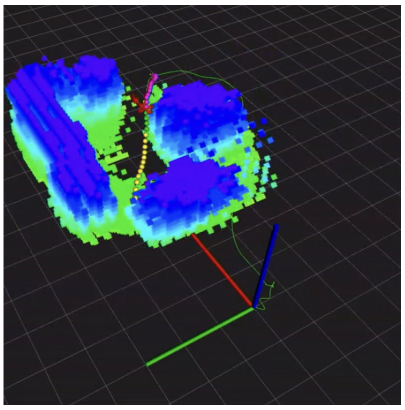
More detail about the indoor fast obstacle avoidance experiments can be found in our video (https://gaowenliang.github.io/).

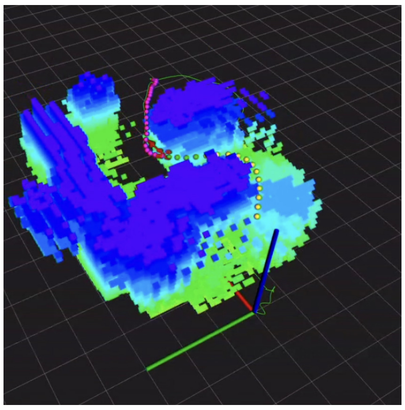


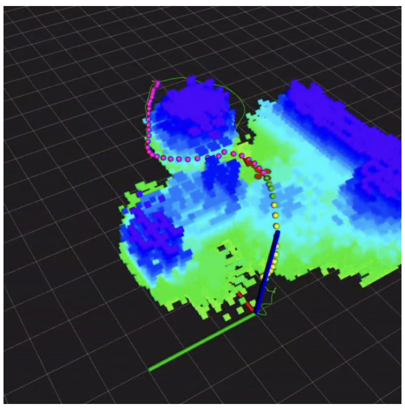
(a) 1s (b) 8s (c) 11s


(d) 20s, arrived the target


(e) 23s


(f) 28s


(g) 31s


(h) 36s

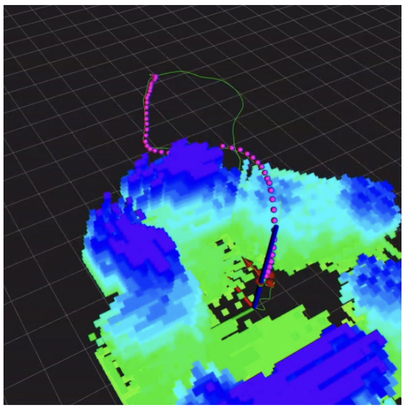
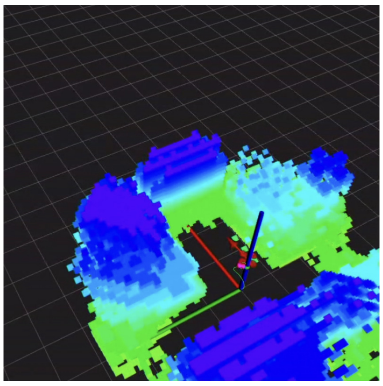

(i) 42s, back to the start point

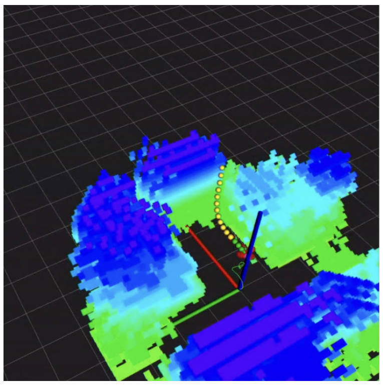
Figure 8.8: The snapshots of the map and the trajectories during the flying mission.

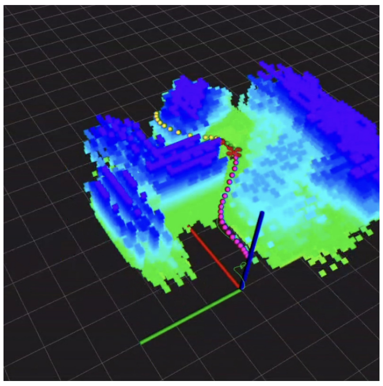
We verify our system performance in both localization, mapping, and planning in both sparse and complex environments. Dozens of experiments have been demonstrated to show the robustness of our system.

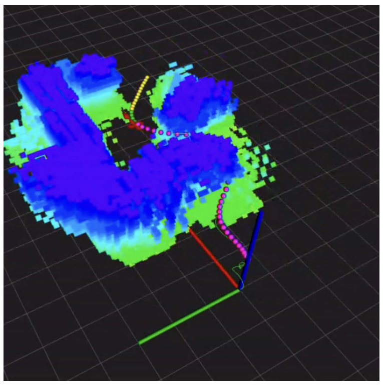
Keep working in the complex and narrow environments is a challenge even for most state-of-the-art autonomous micro aerial robots [44]. Since most of the feasible space is occluded by obstacles, more perception coverage course more safety, especially for exploration. With the dual-fisheye omnidirectional system, all the surrounding obstacles can be detected even if such obstacles are parallel or in the back of the quadrotor.

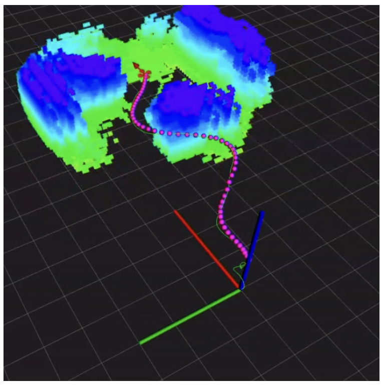
Thanks to the omnidirectional perception, our system is able to travel in any direction safely. We demonstrate the moving frontward and backward. In the experiments, the aerial robot flies to the destination then comes back to the start point, as shown in Fig. 8.8 and Fig. 8.9. The back moving trajectory is generated in real-time with considering the backward perception, not only just follow the inverse of the frontward moving path. As shown in Fig. 8.8, the waypoints are different in frontward and backward segments.

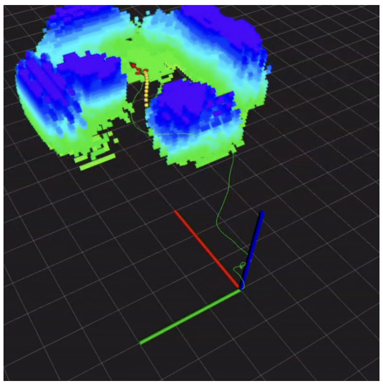

(a) 0s

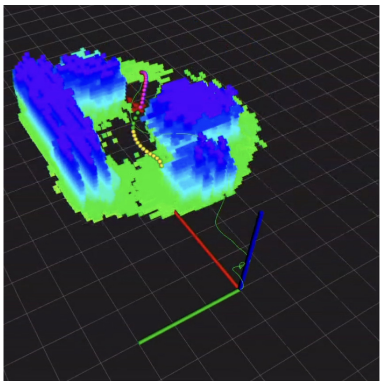

(b) 2s

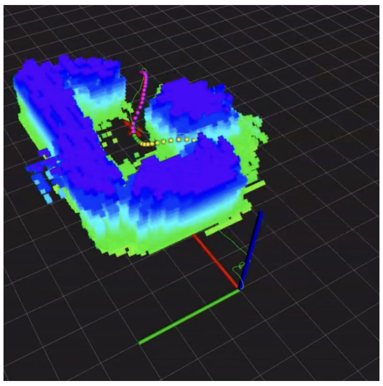

(c) 9s

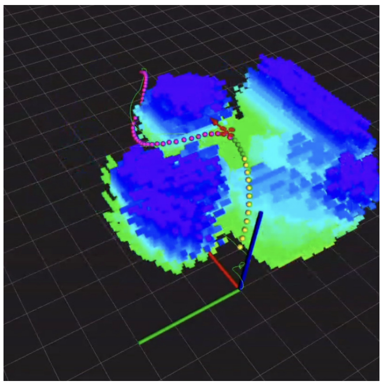

(d) 15s


(e) 21s, arrived the target


(f) 52s


(g) 27s


(h) 29s


(i) 37s

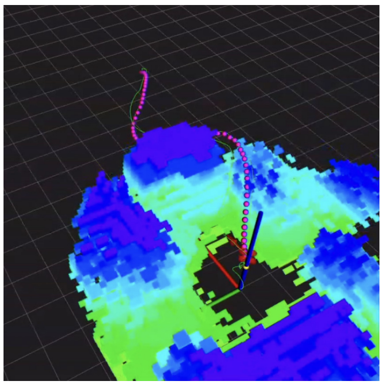
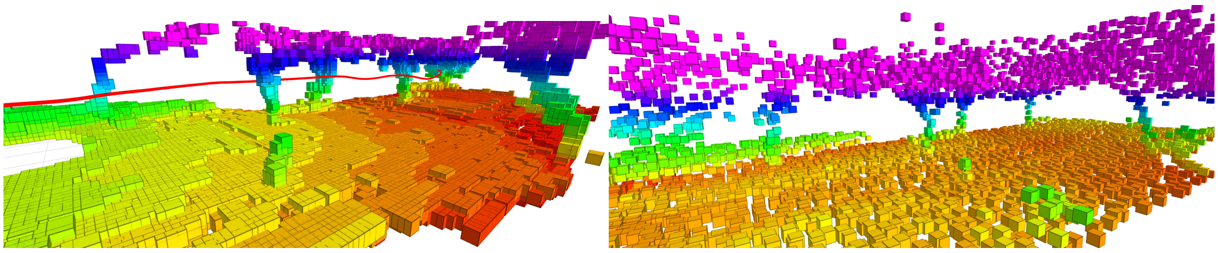

(j) 41s, back to the start point

Figure 8.9: Another experiment results in the same environments as in Fig. 8.8.

## 8.4 Autonomous Flight in Outdoor Environments

We validate the performance of our autonomous flight system in a forest-like outdoor environment. Since most of the obstacles in the forest-like environments are trees, the size of the obstacles may be smaller than in the indoor environments.

We first demonstrate the grid map generated by our proposed dual-fisheye omnidirectional perception system in Fig. 8.10(a). The environments are constructed with 0.15$m$ voxel, which benefits calculation efficient and also satisfies safety navigation. We compared the map results with a 3D LiDAR system in [76] (shown in Fig. 8.10(b)). The proposed dual-fisheye system provides a high-quality map for autonomous navigation and denser than the LiDAR method.



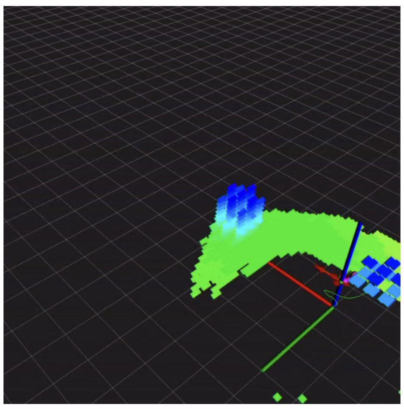
(a) Map by dual-fisheye system (b) Map by LiDAR in [76]

Figure 8.10: The raw grid map in a forest-like outdoor environment.

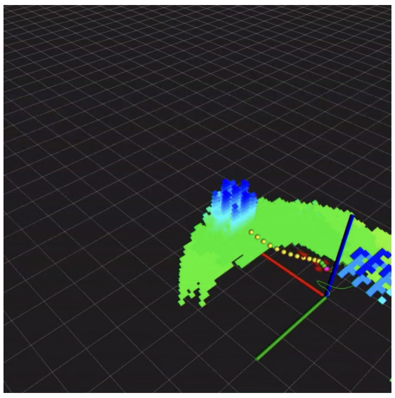
The aerial robot detects and avoids the thin tree branches and follows the trajectory which is generated in real-time from planning module.

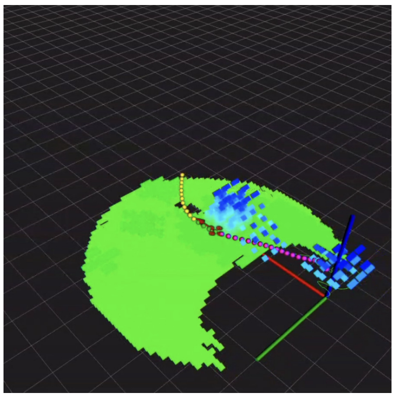
More experimental trials and online visualization can be found in the video attachment.

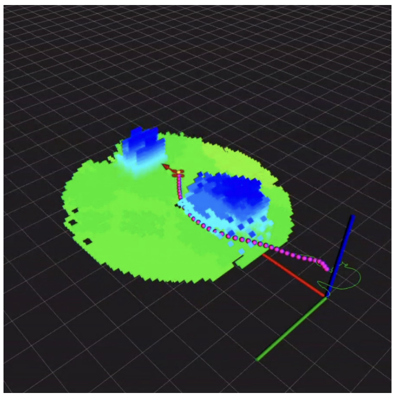
(a) A closeup during shuttling trees

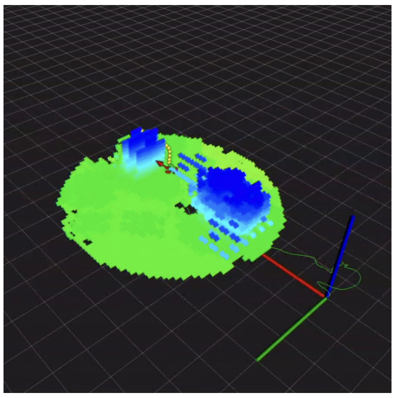


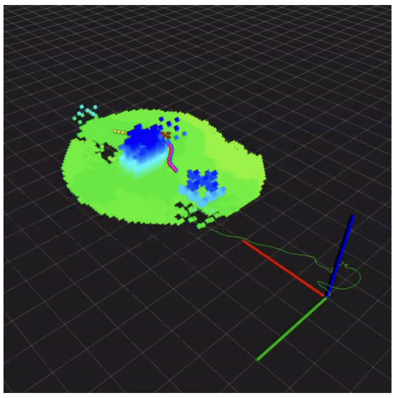
(b) 0s

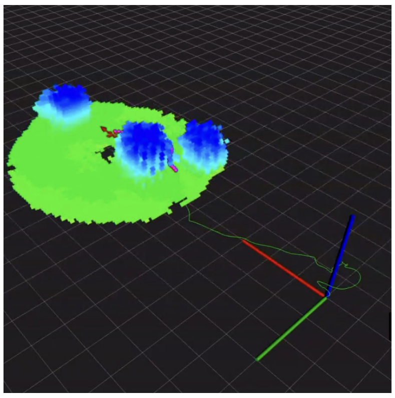

(c) 2s

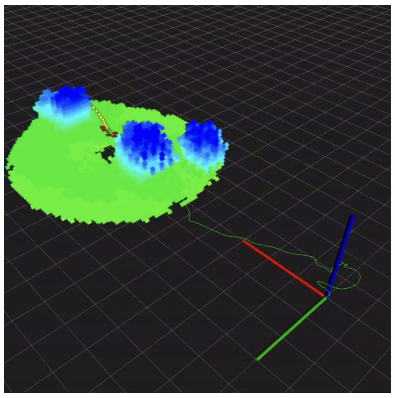

(d) 10s

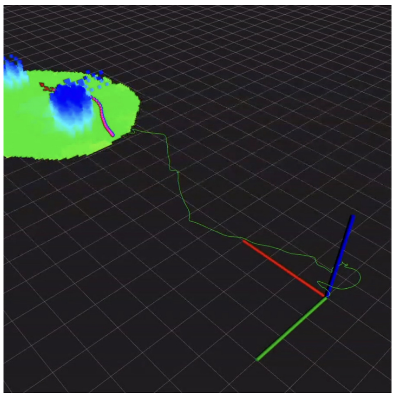

(e) 16s

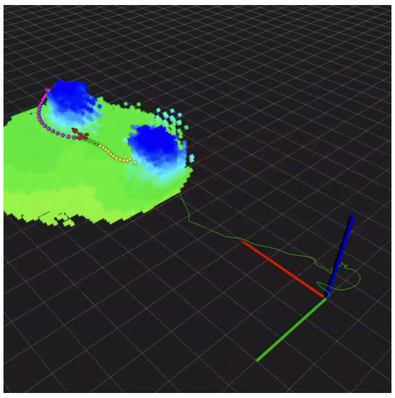

(f) 19s

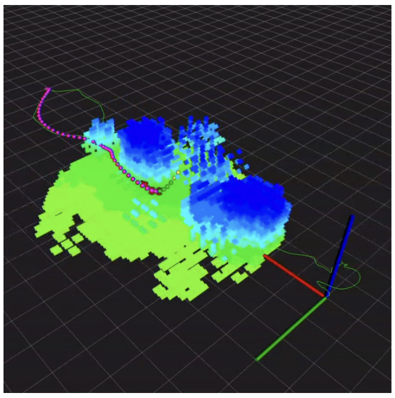

(g) 24s

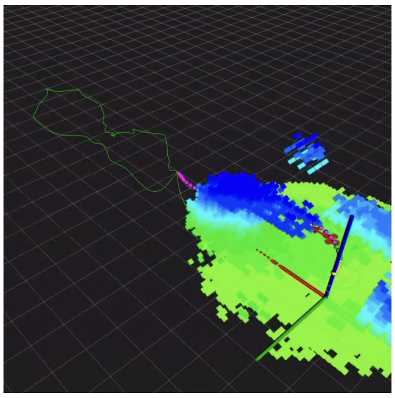

(h) 30s


(i) 31s


(j) 43s, arrived the farest target


(k) 54s


(l) 62s


(m) 78s

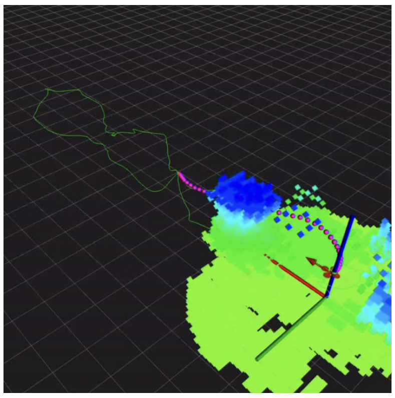
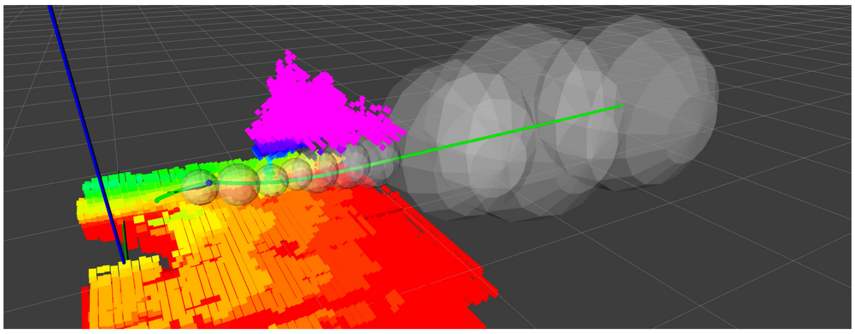
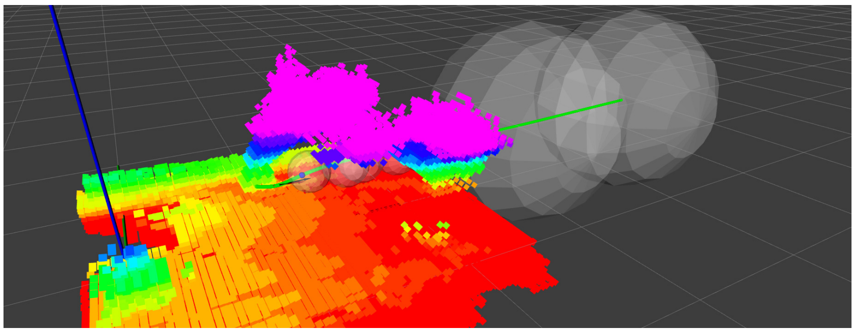
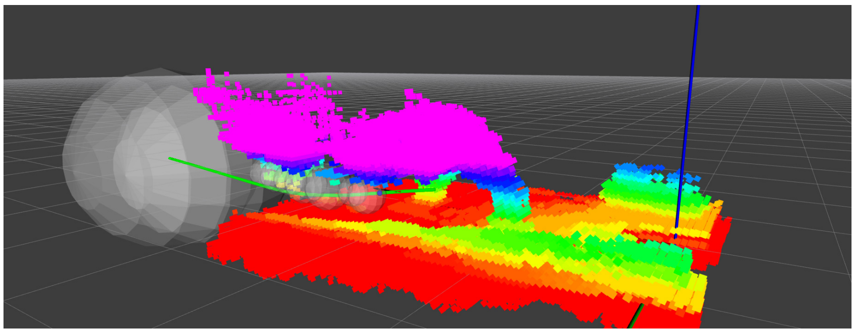

(n) 81s, back to the start

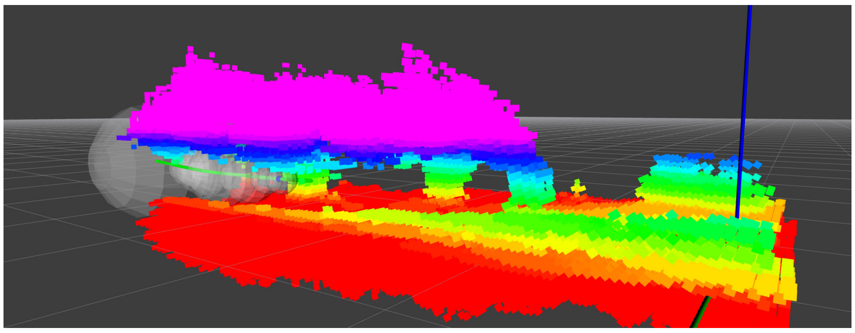
Figure 8.11: A experiment shuttling trees in the forest-like environments.

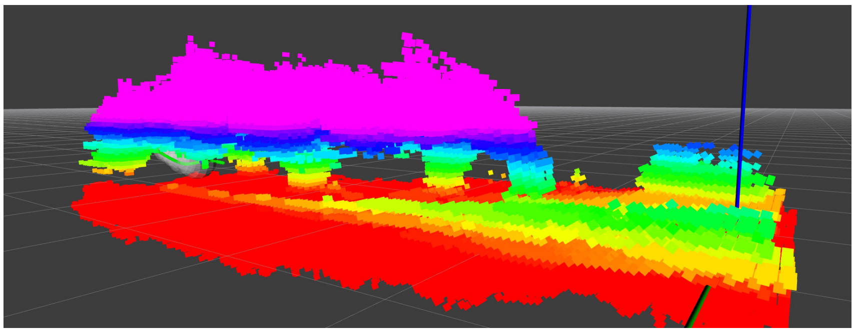
(a) A closeup during shuttling trees

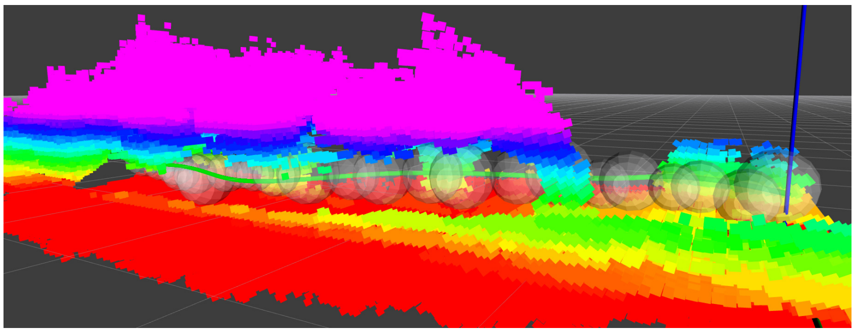

(b) 1s

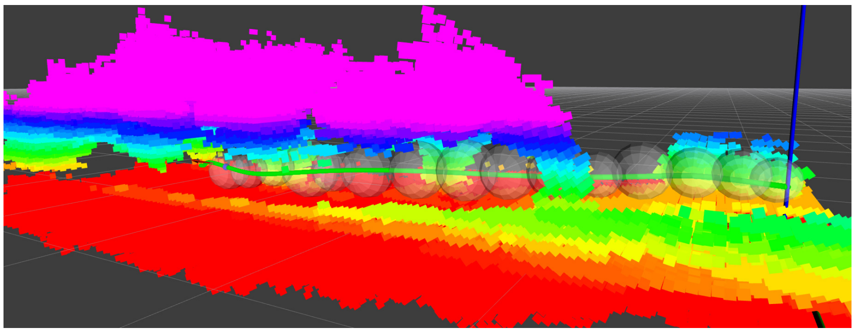

(c) 10s


(d) 20s, view point changed


(e) 30s


(f) 43s, arrived the target


(g) 69s, replan return trajectory


(h) 86s

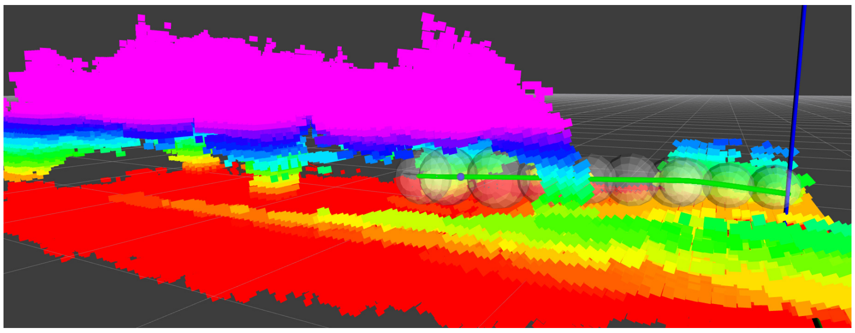
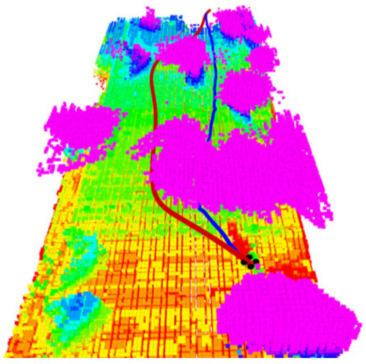

(i) 108s

Figure 8.12: Autonomous navigation in a forest-like outdoor environments.

Autonomous flight test with an vison-based quadrotor. The trajectory generated for the navigation is shown in Fig. 8.13(a), and the overview of the experiment is shown in Fig. 8.13(b).


(a) Trajectory generated in flight.

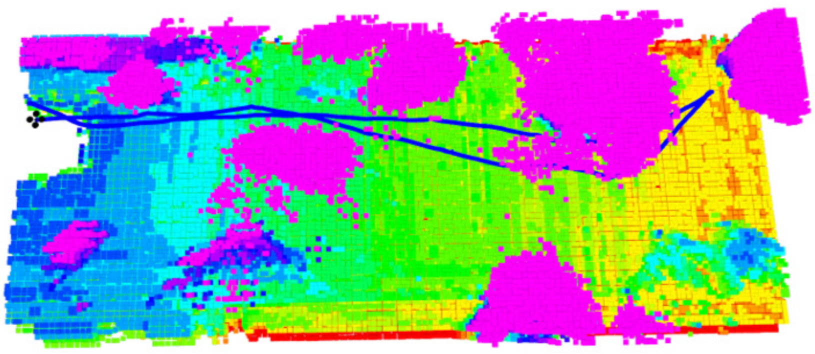
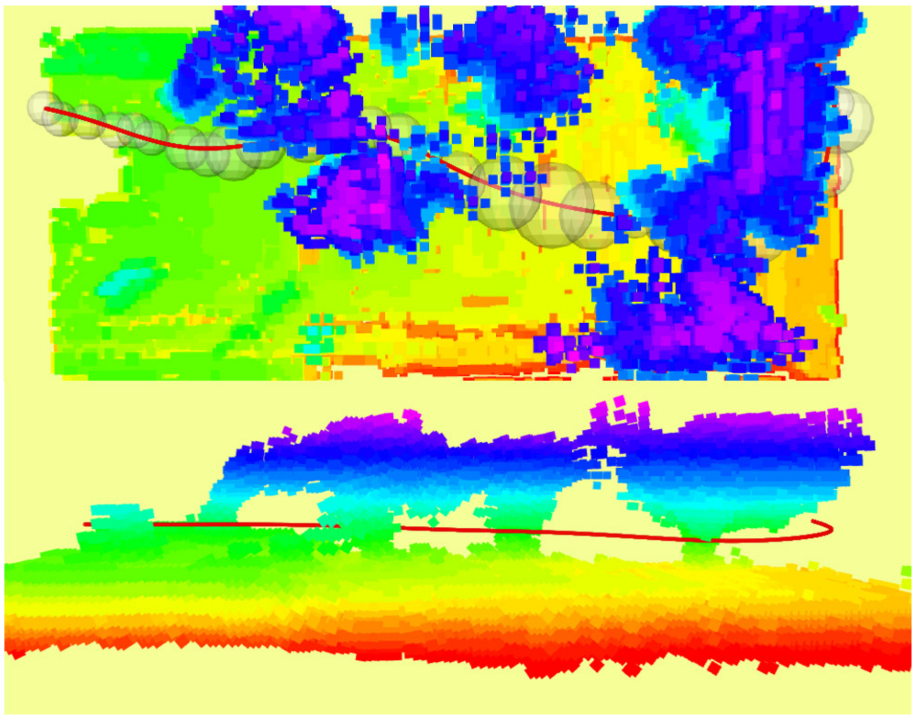
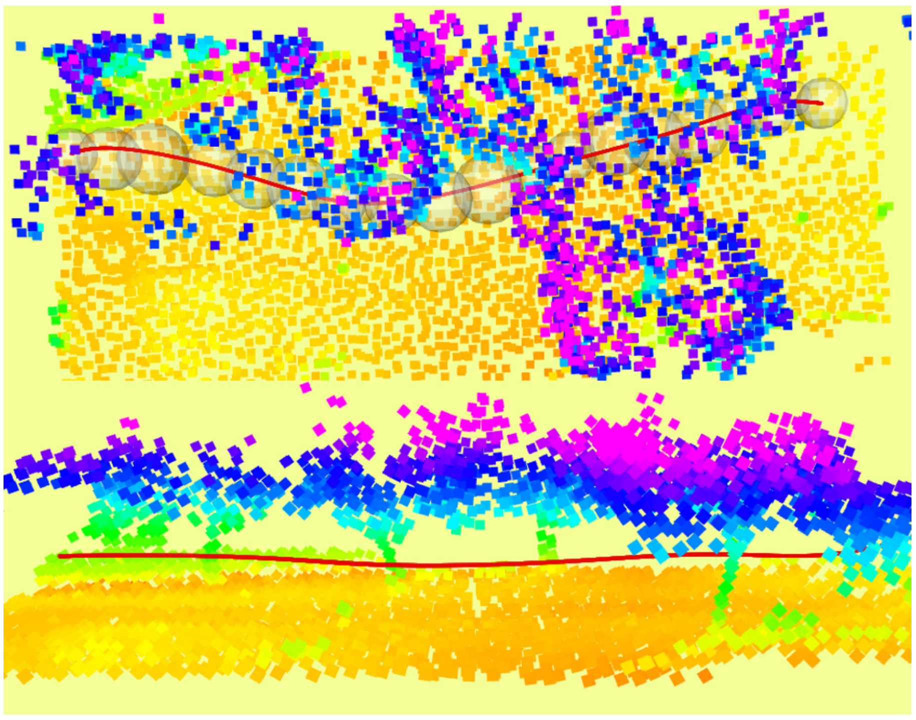

(b) Overview of the quadrotor path.

Figure 8.13: Autonomous flight test with an vison-based quadrotor.

For the same outdoor environments, we compare the flight corridor and the trajectory generated using the dual fisheye cameras in Fig. 8.14(a), and using the Velodyne LiDAR in Fig. 8.14(b). The vision-based method provide a more dense grid map of the environment. However, outliers in the vision-based mapping system occupy much free space and result in a longer flight corridor and trajectory.



(a) Planning with the dual fisheye cameras.



(b) Planning with the LiDAR.

Figure 8.14: Comparison of the flight corridor and the trajectory generated using the dual fisheye cameras, and using the Velodyne LiDAR.

# CHAPTER 9

# CAMERA CALIBRATION

In this chapter, we will introduce the camera parameters calibration of the camera models in Chapter 4, including camera intrinsic coefficients, vignetting coefficients and extrinsic parameters between cameras.

## 9.1 Camera Intrinsic Calibration

With the camera model, five projection parameter $A_{11}$, $A_{12}$, $A_{22}$, $c_x$ and $c_y$, and six polynomial parameters, $\eta_2, \eta_3, \eta_4, \eta_5, \eta_6$ and $\eta_7$, forming the intrinsic parameter $\kappa$, need to be optimized:

$$\kappa = [A_{11}\ A_{12}\ A_{22}\ c_x\ c_y\ \eta_2\ \eta_3\ \eta_4\ \eta_5\ \eta_6\ \eta_7]^T. \tag{9.1}$$

With the formulation of the camera projection function $\pi(\cdot)$, the reprojection error of a 3D point is defined as the error corresponding to the distance between a projected point and a measured point in the image coordinates

$$e = \mathbf{u} - \pi(\mathbf{R}\mathcal{P}_w + \mathbf{T}), \tag{9.2}$$

where $\mathcal{P}_w$ is a point in the world frame, which projects to $\mathbf{u}$; $\mathbf{R}$ is $\mathbf{R}_w^c$, the orientation from world frame to the camera frame, and $\mathbf{T}$ is $\mathbf{T}_w^c$, the translation in the camera frame. Such parameters will be estimated by the *PnP* method during calibration. Camera calibration is to minimize the sum of the Mahalanobis norm of the re-projection error of all the collected points:

$$\min_{\kappa,\mathbf{R},\mathbf{T}} \sum_j \left\| \mathbf{u}_j - \pi(\mathbf{R}\mathcal{P}_{wj} + \mathbf{T}) \right\|^2, \tag{9.3}$$

where $j$ is the index of points used for calibration.

In particular, the polynomial coefficient $\eta_i$ can be initially guessed by fitting the "F-Theta" distortion curve with a polynomial in (4.3) before calibration. During camera calibration, the initial guess of the polynomial coefficients will not be far from the true value.

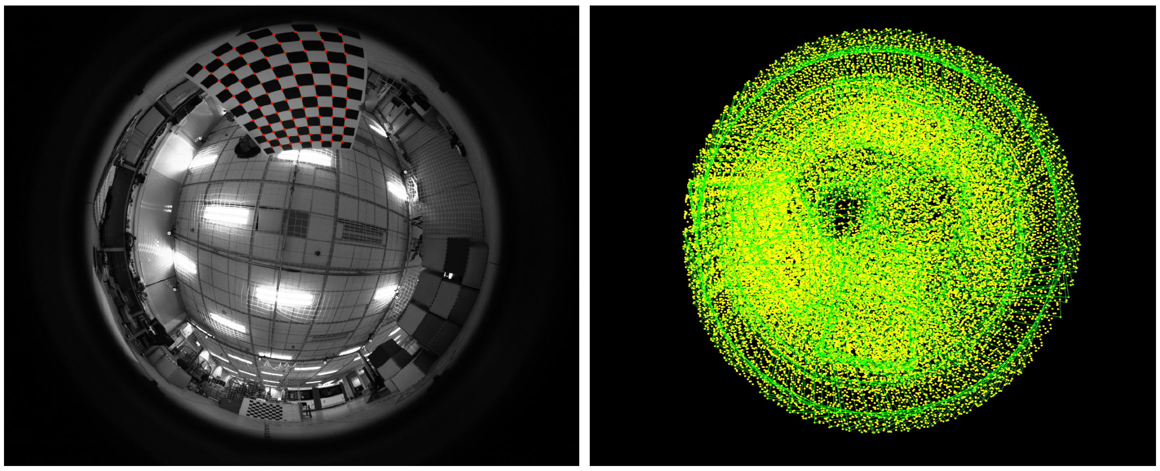

(a) (b)

Figure 9.1: The collection images for the camera intrinsic calibration

After analyzing the performance of the camera intrinsic calibration, we find that the most error of calibration is coming from chessboard images collection and chessboard points detection. We first implement a collection tool to pick out the detected images from the image array and save. Than do calibration with a two times calibration.

Images Collection Tool The images collection tool is to detect the chessboard form an image array and save the detected images. The distribution of collected chessboard points is drawn in a image to make sure the collected points are uniform and full of the camera imaging area. The collection images for the camera calibration is shown in Fig. 9.1. Fig. 9.1(a) shows a sample image for the camera calibration. The red points is the detected chessboard points. The Fig. 9.1(b) shows the sample distributed of collected chessboard points of four hundred images.

Two times calibration Assume that most of the chessboard points in the collected images can be detected normally, and few of the points cannot be detected correctly. After the first calibration, we get the optimized intrinsic parameters. The parameters may be influenced by the incorrect-detected points but still good enough for formulating the projection. With such intrinsic parameters, the reprojection error of the correct-detected points is much smaller than the incorrect-detected points. Thus we set a reprojection error threshold and drop the points whose reprojection error is bigger than the threshold then calibrate again. This process is called two times calibration. The sample image is shown in Fig. 9.2. The green points are detected chessboard points and the red points are the reprojected estimate points. First calibration, we find that exist wrong detection

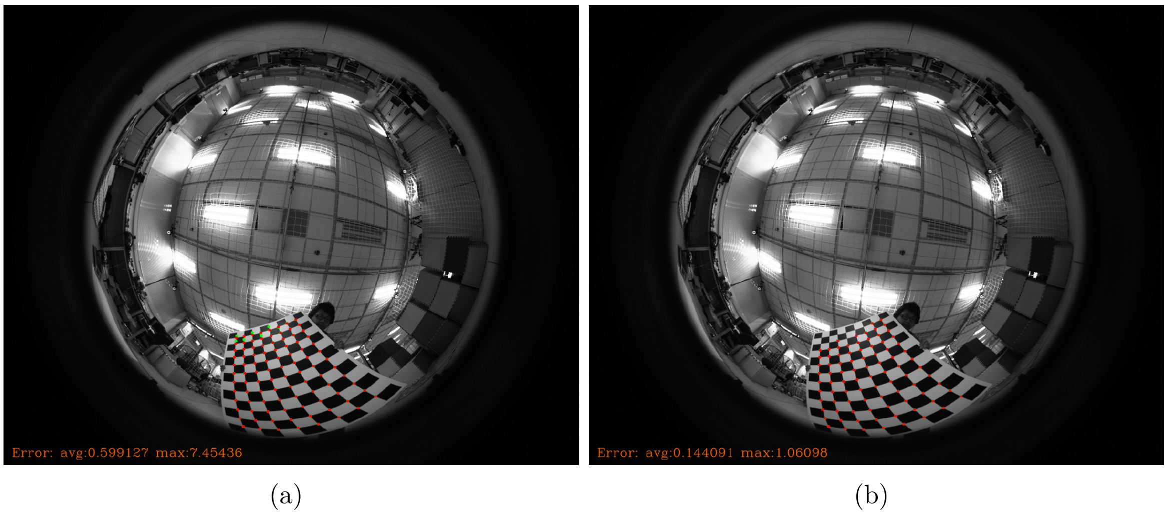

Error: avg:0.599127 max:7.45436
Error: avg:0.144091 max:1.06098
(a)
(b)

Figure 9.2: The sample image of two times calibration.

as shown in Fig. 9.2(a). Remove the wrong detection points as shown in Fig. 9.2(b) and calibration again.

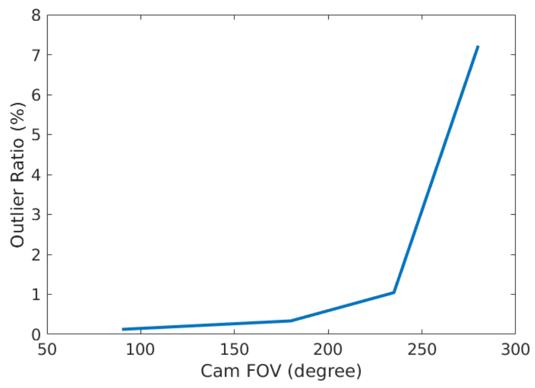


Figure 9.3: The chessboard points detect outlier percentage via the field-of-view.

The chessboard points detect outlier percentage via the field-of-view is shown in Fig. 9.3. While the FOV is less than 150 degrees, there are few outlier points. However, the outlier percentage will increase with the field-of-view. As for the camera with 280-degree FOV, the outlier ratio is almost around 7%.

## 9.2 Camera Vignetting Calibration

With the camera vignetting model in equation (4.13), six order polynomial with three none-zero coefficients is able to fit the vignetting well, thus $k = 3$. For a typical camera lens, three vignetting parameter $\beta_1, \beta_2, \beta_3$ and the intensity of the camera principle point

$I_0$, forming the vignetting parameter $\kappa$, need to be optimized:

$$\kappa = [I_0 \; \beta_1 \; \beta_2 \; \beta_3]^T. \tag{9.4}$$

With the formulation of the pixel intensity, the intensity error is defined as the error corresponding to the distance between the real pixel intensity of calibration pattern and the estimated pixel intensity:

$$e = \mathbf{I}(r) - I_0(1 + \sum_{i=1}^{3} \beta_i r^{2i}), \tag{9.5}$$

Vignetting calibration is to minimize the sum of the Mahalanobis norm of the intensity error of all the collected points:

$$\min_{\kappa} \sum_{j} \left\| \mathbf{I}(r_j) - I_0(1 + \sum_{i=1}^{3} \beta_i r_j^{2i}) \right\|^2, \tag{9.6}$$

where $j$ is the index of points used for calibration.

Consider the vignetting model formulate the intensity of each pixel, the calibration relies on the environment with a static lighting and the pattern which is a flat white object with Lambertian reflectance. Therefore the difficulty of the vignetting model calibration is how to collect data.

Data Collection According to [2], a sheet of printer paper attached to a desk in a typical office is suitable. However, if we collect the points from the image of a sheet of white paper, it is not easy to detect the pixel is the image of paper or not. Considering the intensity data collection issue, we use chessboard as the calibration pattern. The white grid of the normal chessboard can be regarded as flat Lambertian reflectance surface. And we just need to control the light of the environment as static and unified as possible. The chessboard is easy to detect and we collect the intensity of the center of every white grid. The image set for vignetting collection is almost the same as the calibration set of camera intrinsic calibration. As shown in Fig. 9.4, the chessboard grids are detected and the green points figure the points which will be used to do vignetting calibration and the red points will be removed. With a large number of images, the white grid points are more sparse than collect with white paper. It is helpful to avoid noise and calculation acceleration.

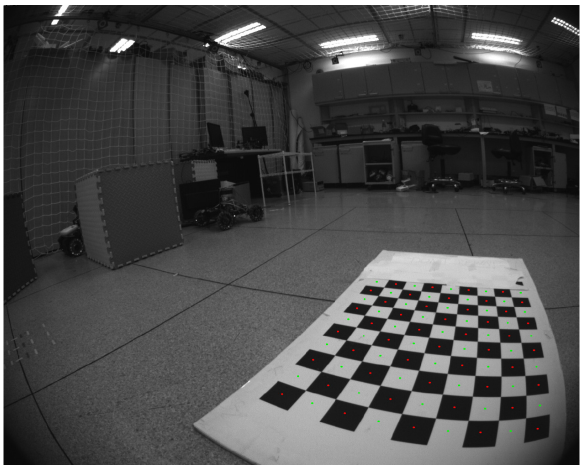


Figure 9.4: A sample image of vignetting calibration data.

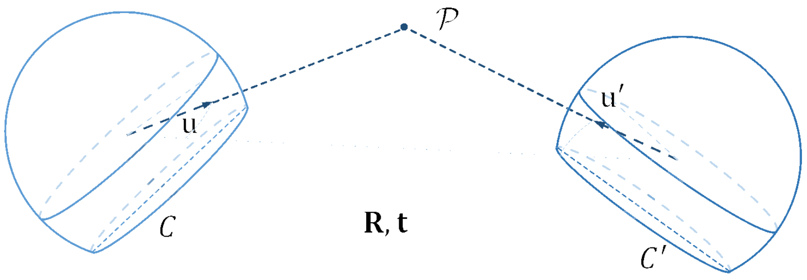


Figure 9.5: A sample image of vignetting calibration data.

## 9.3 Camera Extrinsic Calibration

To achieve extrinsic self-calibration from the epipolar constraint, an optimizable error is formulated with the stereo extrinsic parameters. Our proposed method is based on our previous work [45]. As shown in Fig. 9.5. for points on the line $O^{c_0}\mathcal{P}^{c_0}$, the epipolar line is

$$l_j = \mathbf{E}\bar{\mathcal{P}}_j^{c_0}. \tag{9.7}$$

The epipolar constraint is

$$(\bar{\mathcal{P}}_j^{c_1})^T l_j = (\bar{\mathcal{P}}_j^{c_1})^T \mathbf{E}\bar{\mathcal{P}}_j^{c_0} = 0, \tag{9.8}$$

which is satisfied with scaling factor $\mathbf{t}$ multiplies. The epipolar error of each pair of matched feature points $\bar{\mathcal{P}}_j^{c_0}$ and $\bar{\mathcal{P}}_j^{c_1}$ is defined as

$$||(\bar{\mathcal{P}}_j^{c_1})^T \mathbf{E}\bar{\mathcal{P}}_j^{c_0}||^2, \tag{9.9}$$

which is the square of the distance between point $\bar{\mathcal{P}}_j^{c_0}$ and epipolar line $l$ in view of the epipolar geometry. Since the observation of detected feature points is corrupted with noise and matching error exists, the epipolar constraint cannot be satisfied perfectly. Our approach is to solve for **R** and **t** on the manifold that minimizes the overall epipolar error for all the matched feature correspondences of each virtual camera. Detailed treatment of the self-calibration approach can be found in [45]. According to (7.4), the total process is

$$\begin{aligned}&\min_{\mathbf{R},\mathbf{t}} \sum_k \sum_j \left\| \left[\mathbf{R}_{k'}^{c_1} \pi_{k'}^{-1}(\mathbf{u}'_j)\right]^T \lfloor \mathbf{t} \times \rfloor \mathbf{R} \left[\mathbf{R}_k^{c_0} \pi_k^{-1}(\mathbf{u}_j)\right] \right\|^2, \\ &s.t. \, ||\mathbf{t}|| = 1\end{aligned} \tag{9.10}$$

where **E** is substituted by $\lfloor \mathbf{t} \times \rfloor \mathbf{R}$ using equation (7.3), $\mathbf{u}_j$ and $\mathbf{u}'_j$ are the corresponding features. The unit norm constraint $||\mathbf{t}|| = 1$ refines the stereo calibration as 5-DOF optimization, three for rotation $\mathbf{R} \in SO(3)$ and two for translation $\mathbf{t} \in \mathbb{S}^2$ [45].

## 9.4 Camera Intrinsic Calibration with Screen

The visual systems are used in the industrial applications. For the industrial users, the camera calibration is an essential technique. However, with the developing of the autonomous technology, the vision-based system will be used widely such as autonomous driving and others. It is necessary to improve a calibration approach which can work without specially trained staff.

As for traditional camera intrinsic calibration, there exist a gap: The raw data for calibration is collected by artificial capture. Thus, the distribution of calibration points is all controlled by a human.

In order to collect enough data for calibration, there would be some movements of the camera or the calibration markers (chessboard, random point board), to change the relative pose. That means the calibration result is unstable because of without the uniform collection movements. And if with the movements of cameras, the rolling shutter cameras cannot get good images. With the movements of the calibration, markers would cause blur and geometry distortion.

For the industry application, the calibration data should be reliable. And for the cameras used in factories and consumer products, the properties of the cameras are similar. Thus, as for a production line, the intrinsic calibration station can be specially designed for the cameras of products.

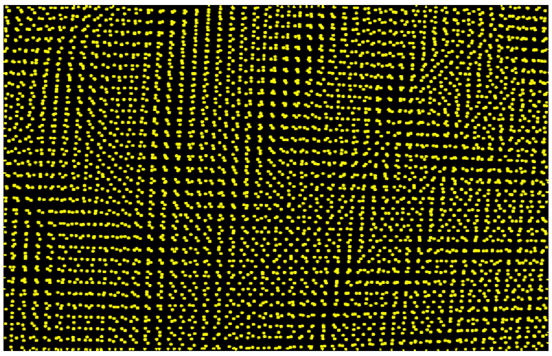


Figure 9.6: The points data distribution of screen calibration.

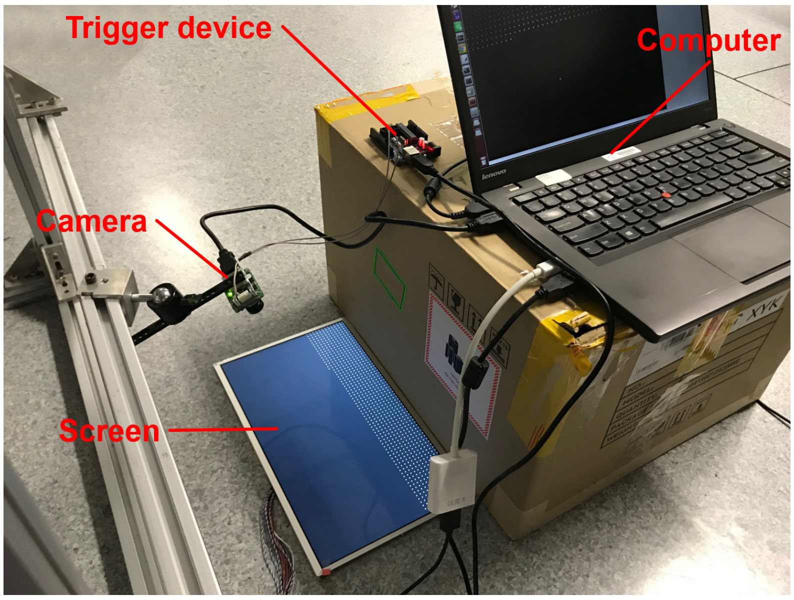


Figure 9.7: Screen calibration system.

In this work, we present a novel solution to camera calibration constructed with screens. With screens display pixel points one by one, the whole calibration pattern is the dense points captured by the camera. The calibration accuracy is comparable with the transitional method with chessboard pattern, so the camera model can be abstract with a spline mesh, map the relationship between the ray vector and pixel. The model is able to describe all kinds of camera distortions, including radial distortion, tangent distortion, or other distortions. Unlike a variety of existing methods, the calibration operation of our method needs little artificial interaction. Thus it is easy to operation, and do not need a special training for users. Also, the calibration of our method requiring only a limited amount of space, so it can be well applied in industrial environments.

Data Collection The camera intrinsic calibration is to refine the parameters of a camera model which describe the relationship between the 3D world points and the 2D pixel

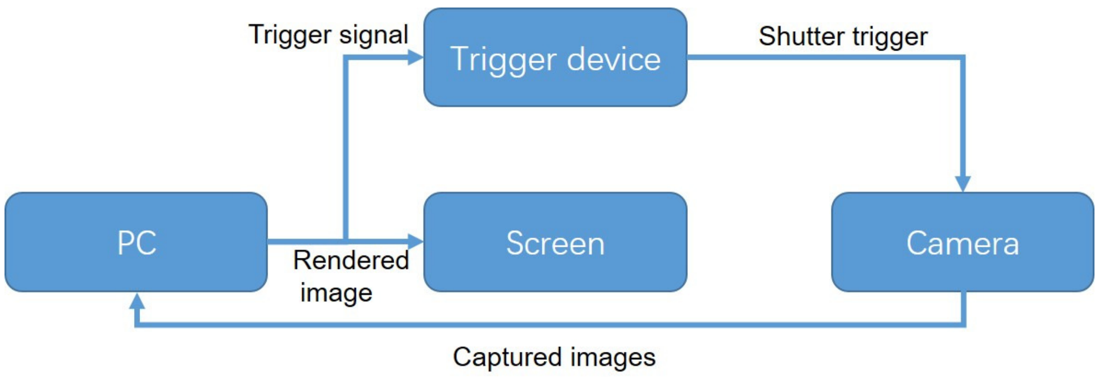


Figure 9.8: The pipeline of screen calibration.

coordinates. The data for camera intrinsic calibration is a series of images capturing a manufactured calibration pattern. The 2D pixel coordinates are detected from the images and the 3D world points are known because of the calibration pattern.

According to Fig. 9.1(b), it is not easy to get a unified 2D points distribution for a traditional chessboard pattern method. As for nonlinear optimization problem, which is the essence of camera calibration, the data distribution influences the results by weight: the data denser, the weight greater. Using a screen as a calibration pattern, it is obvious to generate a unified data distribution with the screen control. We can light one pixel of the screen and detect the light pixel from the image. It is clear to create the point-to-point correlation. And the data can be denser and more unified: a screen can provide a maximum 2 million pixel-wise calibration pattern with a $1920 \times 1080$ resolution. As for a $12 \times 8$ chessboard, the number is 96. We downsample the screen points data and use thousands of points for calibration, the distributed image is shown in Fig. 9.6.

The system is shown in Fig. 9.7, the rotation and the translation between the camera and the screen is fixed. The screen calibration system includes a screen, an embedded trigger device, and the camera, all connect to a computer. The computer renders an image with a light point with known coordinate at first, shows in the screen. Then a trigger signal is sent to the embedded trigger device. The embedded trigger device control the camera to capture an image with the shutter trigger. Finally, the captured image is collected by the computer. The system pipeline is shown in Fig. 9.8.

The screen renders frequency is around 60-144 Hz. Thus during the render images changed, it is reasonable to capture an image mixed with this image and the last image because of the inaccurate trigger control. To avoid this, we render the images with incremental light points: there is one more light point shown in the screen than the last

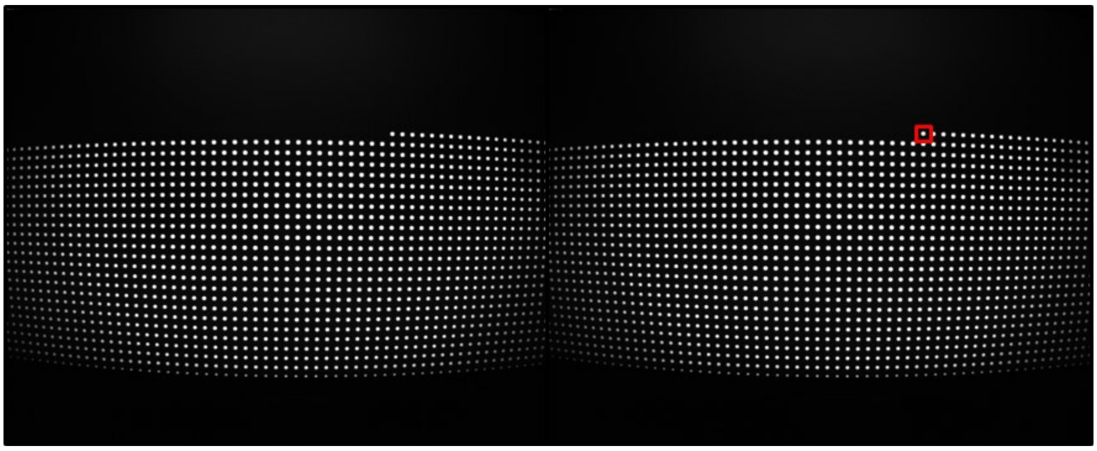


Figure 9.9: The raw images of screen calibration.

image. The sample of raw images are shown in Fig. 9.9, the left image is $k^{th}$ captured image $\mathbf{M}_k$ and the right image is the $k+1^{th}$ captured image $\mathbf{M}_{k+1}$. The $k+1^{th}$ light point in the image $\mathbf{u}_{k+1}$ can be detected in the difference image $\mathbf{M}^d_{k+1} = \mathbf{M}_{k+1} - \mathbf{M}_k$.

The difference image requires to be preprocessed because of the image intensity noise in order to precisely detect the light points:

$$\mathbf{M}^p_{k+1} = \alpha \mathbf{M}^d_{k+1} + \beta, \tag{9.11}$$

where $\alpha$ is the image intensity gain and $\beta$ is the image intensity bias.

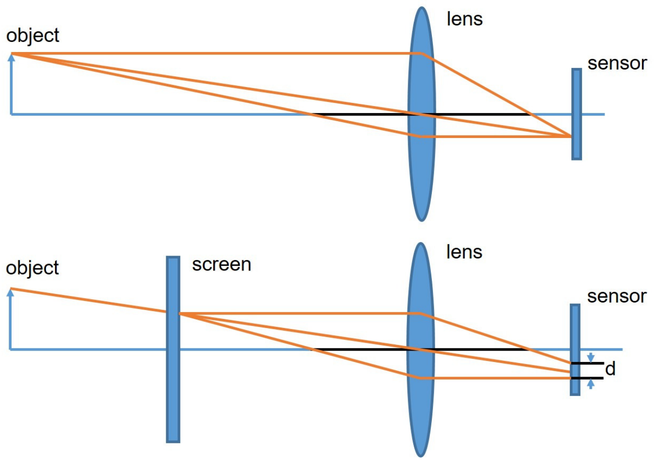
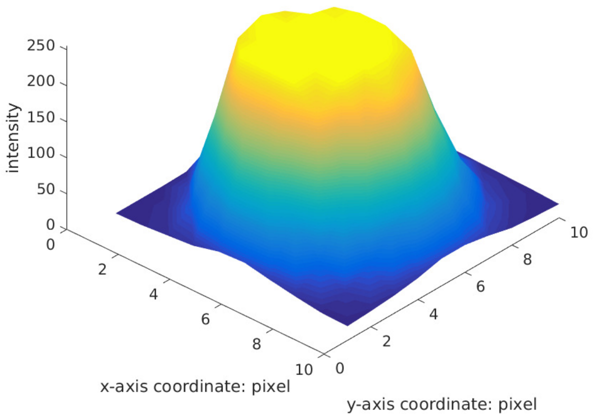


Figure 9.10: The illustration of lens projection.

Light Point Detect In the calibration system, the camera is mounted close to the screen, the distance is less than $1m$. Since the camera is focused at a farther distance ($\approx 10m$), the projection of the light pixel points will amount to circular disks on the sensor instead of a sharp point, named as circles of confusion (CoC) [33]. The size and shape of the CoC are depended on the distance of the holes to the sensor and the focal plane and the shape and size of the camera aperture.

The lens distortion might influence the shape of the CoC such that it deviates from a perfect circle. Consider the shape of CoC is influenced via the distortion, the projected location of a point is estimated by fitting a 2D Gaussian function to the image blob. Fig 9.11 shows a sample of a fitted Gaussian function and the estimated the center of one blob. Since the screen pixel object can be assumed to have infinitesimally small size, as shown in Fig. 9.10, this method does not suffer from foreshortening effects known to pose challenges for location estimation using spatial fiducials such as circular feature points [36].



(a)

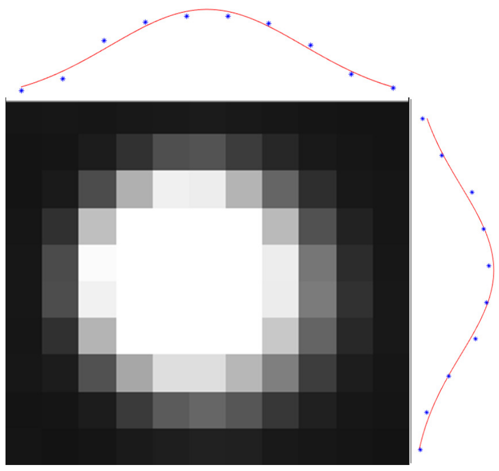

(b)

Figure 9.11: The image of the projected point.

The intensity of the CoC is described as a Gaussian distribution:

$$\begin{aligned}\mathbf{G}[u_c|\sigma_u] &= \frac{1}{\sigma_u\sqrt{2\pi}}e^{-u_c^2/2\sigma_u^2},\\ \mathbf{G}[v_c|\sigma_v] &= \frac{1}{\sigma_v\sqrt{2\pi}}e^{-v_c^2/2\sigma_v^2},\end{aligned} \tag{9.12}$$

where the $u_c$ and $v_c$ is $\mathbf{u}_c = [u_c\ v_c]^T$, the sub-pixel image coordinate of the light point, $\sigma_u$ and $\sigma_v$ is the standard deviation of the Gaussian function. For each light point, we estimate the $\mathbf{u}_c$ by fitting a Gaussian function of the blurred intensity profile.

The back-end of the screen calibration is to fit the model with 3D points and the detected 2D points, similar with the process with chessboard methods. The calibration is to minimize the sum of the Mahalanobis norm of the re-projection error of all the collected points:

$$\min_{\kappa,\mathbf{R},\mathbf{T}} \sum_{j} \|\mathbf{u}_j - \pi(\mathbf{R}\mathcal{P}_{wj} + \mathbf{T})\|^2 , \tag{9.13}$$

where $j$ is the index of points used for calibration.

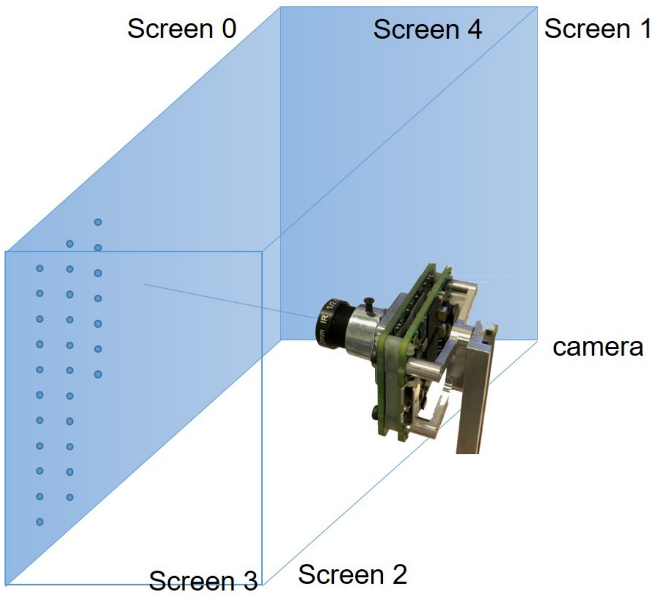


Figure 9.12: The illustration of fisheye calibration.

Since the calibration pattern should cover all of the sensing region, for camera with ultra-wide filed-of-view such as fisheye cameras, only one screen is not enough. A system with 5 to 6 screen as shown in Fig. 9.12.

## 9.5 Calibration Results

### 9.5.1 Intrinsic Calibration Results

Our proposed monocular camera model and its calibration approach conform to the ultra-wide FOV fisheye lens. We test the calibration, analyze the calibration result and show a uniform distribution of the error.

We show calibration results for two different cameras, an mvBlueFOX-MLC202bG[1] with an image resolution of $1280 \times 960$ and a 235-degree FOV lens, and an mvBlueFOX-MLC202aG with an image resolution of $1280 \times 1024$ and a 245-degree FOV lens. The

[1]https://www.matrix-vision.com

points for calibration are detected and collected from images of a chessboard with $12 \times 9$ grid and squares of 70 $mm$. We use the open source chessboard detector from OpenCV[2], as shown in Fig. 9.1(a). We modify the CamOdoCal in [27] as our calibration toolbox, based on Ceres Solver[3]. The difference between the polynomial-fitted designed radial

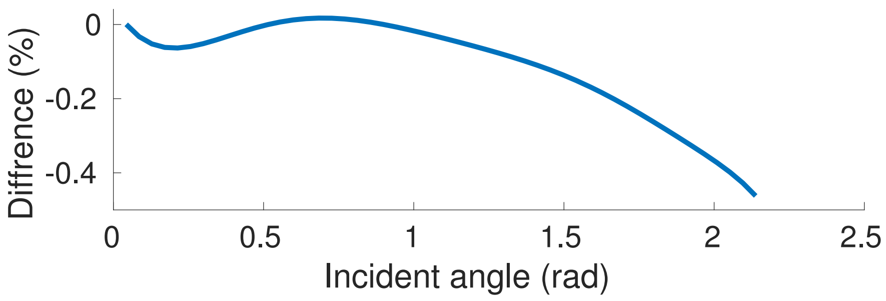


Figure 9.13: The difference between design and calibration, less than 0.5%.

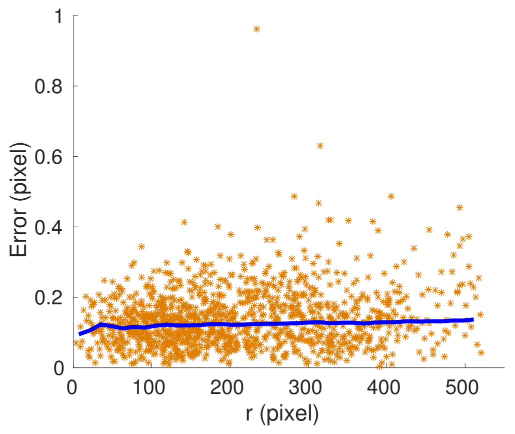
distance curve (4.3) and calibrated one is less than 0.5%, as shown in Fig. 9.13, and the calibrated radial distance is shown in Fig.4.3(b). The exact accuracy of the polynomial fit is unknown due to the lacking of the ground truth.

The ultimate goal of the calibration is to improve projection and back projection and offer accurate undistorted process. After calibration with 10 images, the radial distribution of the error versus the projected radial distance of both cameras is shown in Fig. 9.14. Fig. 9.14(a) and Fig. 9.14(b) shows the reprojection error of the 235-degree FOV camera and the 245-degree FOV camera. The blue curve depicts the median residual error in different segments of the radial distance.

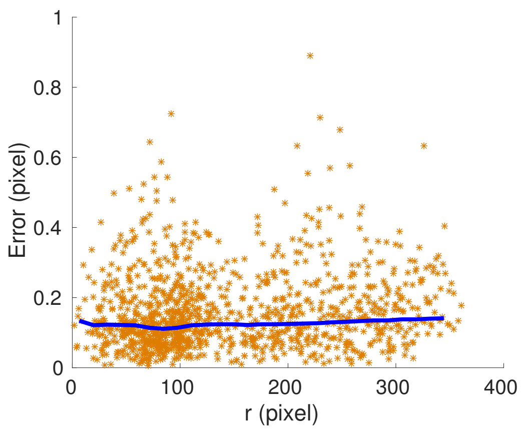
The convergence of the average reprojection error and root mean square (RMS) reprojection error is shown in Fig. 9.14(c) and Fig. 9.14(d), and we also add the result of a unified camera model of Mei [52] as the reference. We test the calibration with 10 images to 50 images. Both Mei model and proposed model are accuracy enough since the average reprojection error is less than 0.4 pixels, however, our proposed model has lower reprojection error for ultra-wide FOV fisheye camera. The average reprojection error of our proposed model of each camera is around 0.2 pixels. The calibration result is stable, and the fluctuation in the average reprojection error of each camera is less than 0.05 pixels. The RMS residual error fluctuates when insufficient images are used since the chessboard detection and camera pose estimation are not accurate due to the distortion in some images.

[2]http://opencv.org/

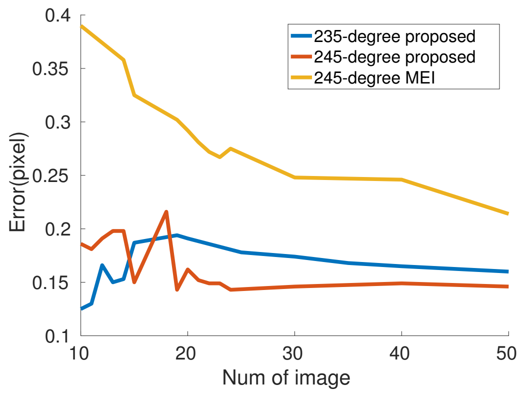
[3]http://ceres-solver.org/



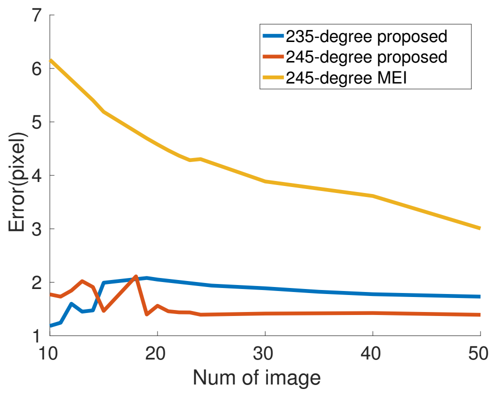
(a) Error of 235-degree FOV lens.



(b) Error of 245-degree FOV lens.



(c) Average error.



(d) RMS error.

Figure 9.14: Fisheye camera intrinsic calibration result.

As discussed in Sect. 9.1, the two times calibration is needed, especially for wide FOV cameras. The reprojection error of the two times calibration is shown in Fig. 9.15. Fig. 9.15(a) shows the average reprojection error in pixel coordinate and Fig. 9.15(b) shows the RMS error. Both errors decrease with the outlier removed in second calibration and the wider FOV, the more improvements.

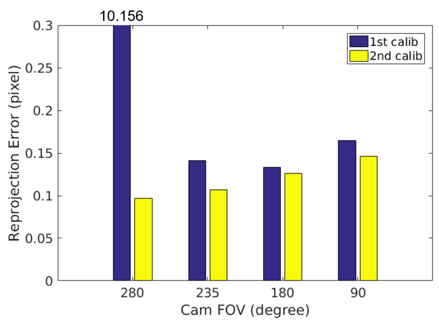
### 9.5.2 Vignetting Calibration Results

We show the vignetting calibration results for two different cameras, an mono camera PointGrey CM3-U3-13Y3M-S-BD[4] and a color camera PointGrey CM3-U3-13Y3C-S-BD[5] with an image resolution of $1280 \times 1024$.

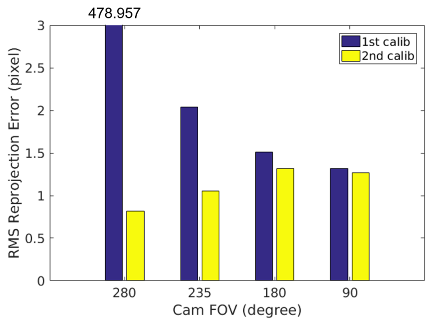
The data collection is shown in Fig .9.4, almost same as camera intrinsic calibration.

[4]https://www.ptgrey.com/chameleon3-13-mp-mono-usb3-vision-board-level-on-semi-python-1300

[5]https://www.ptgrey.com/chameleon3-13-mp-color-usb3-vision-board-level-on-semi-python-1300



(a)



(b)

Figure 9.15: The reprojection error of some sample cameras.

However, the light should be sure as static and unified as possible. After optimized, the parameters of mono camera and color camera is shown in Table. 9.1. The results are visualized inFig. 9.16. Fig. 9.16(a) and Fig. 9.16(c) show the vignetting gain curve of the mono camera and the color camera. Fig. 9.16(b) and Fig. 9.16(d) show the vignetting distribution in the whole images of the mono camera and the color camera due to the calibration results.

After the vignetting calibration, the vignetting of camera can be removed with the reverse process of equation (4.12) and the vignetting gain compensation, as shown in Fig. 9.17. The source image with huge vignetting as shown in Fig. 9.17(a), whose center is light and periphery is dark. The image removed vignetting is shown in Fig. 9.17(b), the intensity is more uniform. The vignetting rectified color image is shown in Fig. 9.17(d) from raw image Fig. 9.17(c). However, because of the huge vignetting and the digital sampling of the camera, the corners are so dark that the information almost lost and cannot be restored.

Consider the vignetting rectified during ROI extraction discussion in Sect. 4.4, the ROI image is better as shown in Fig. 9.17. The raw fisheye image is shown in Fig. 9.17(e) and the ROI image is shown in Fig. 9.17(f). The top image is the ROI image of Fig. 9.17(e) without vignetting rectified, and the bottom image is the vignetting compensated ROI

Table 9.1: Vignetting Parameters of Mono Camera and Color Camera

| Camera | Channel | $I_0$ | $\beta_1$ | $\beta_2$ | $\beta_3$ |
|---|---|---|---|---|---|
| Mono | - | 177.428 | $3.7255 \times 10^{-4}$ | $-3.2368 \times 10^{-9}$ | $-5.2368 \times 10^{-16}$ |
| Color | Bule | 253.664 | $8.1332 \times 10^{-5}$ | $-9.4155 \times 10^{-10}$ | $6.9572 \times 10^{-16}$ |
| Color | Green | 252.190 | $1.6402 \times 10^{-4}$ | $-1.5552 \times 10^{-9}$ | $1.4890 \times 10^{-15}$ |
| Color | Red | 251.705 | $1.5510 \times 10^{-4}$ | $-1.0805 \times 10^{-9}$ | $5.7441 \times 10^{-16}$ |

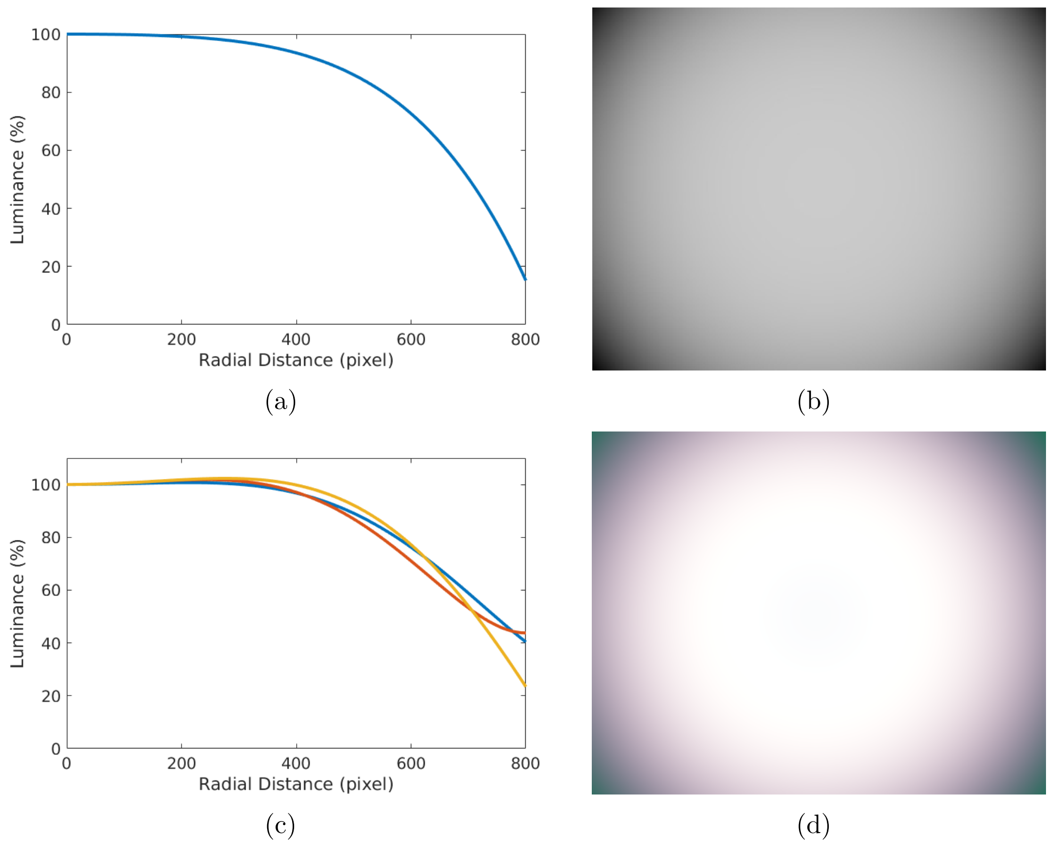


Figure 9.16: Vignetting calibration results of color camera with three channels.

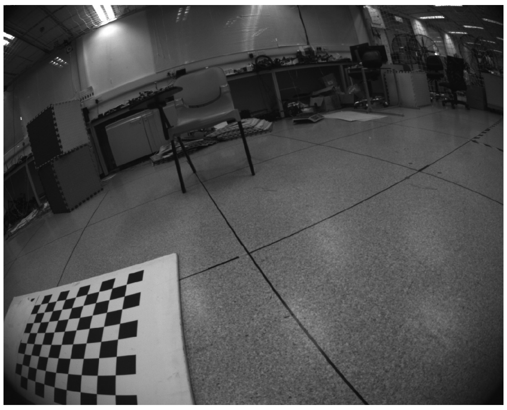
image of upper one.

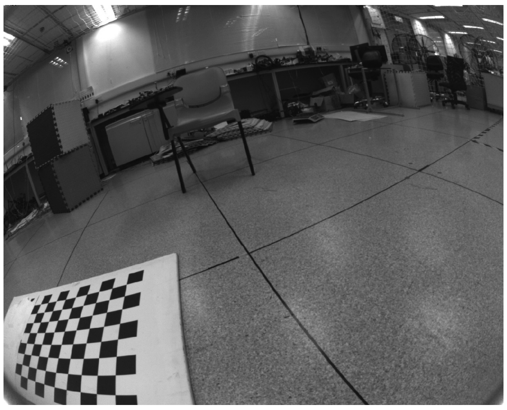
We make our implementation of Vignetting Model open-sourced[6].

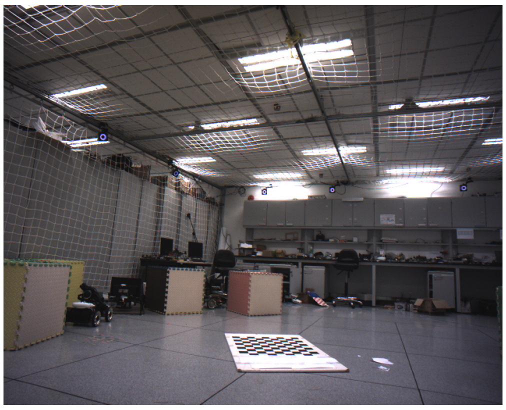
## 9.5.3 Extrinsic Calibration Results

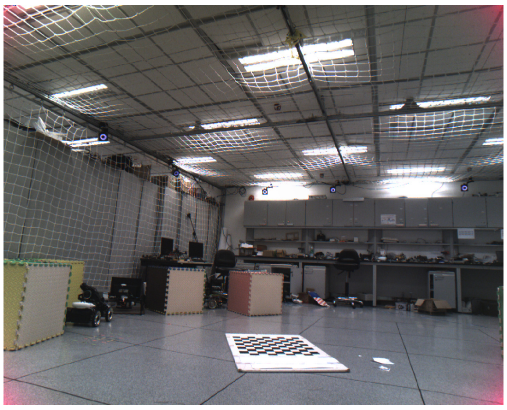
The extrinsic parameters are optimized by our proposed self-calibration approach, and achieves an accuracy that the standard block matching (BM) algorithm operates and can produce the disparity map, as Fig. 9.19 shows. In the scene shown in Fig. 9.19(a), the disparity map computed using initial stereo extrinsic parameters, by block matching (BM) algorithm is shown in Fig. 9.19(b) and the disparity map computed using optimized extrinsic parameters is shown in Fig. 9.19(c).

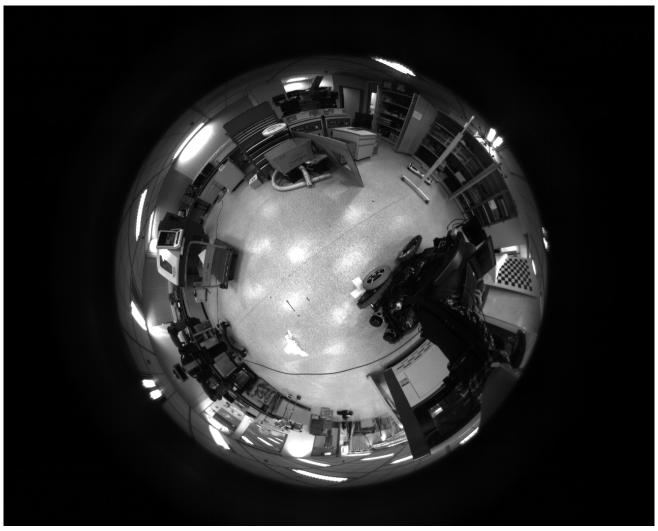
[6]https://github.com/gaowenliang/vignetting_calib


(a) Raw mono image


(b) Mono image with vignetting rectified


(c) Raw color image


(d) Color image with vignetting rectified


(e) Raw fisheye image

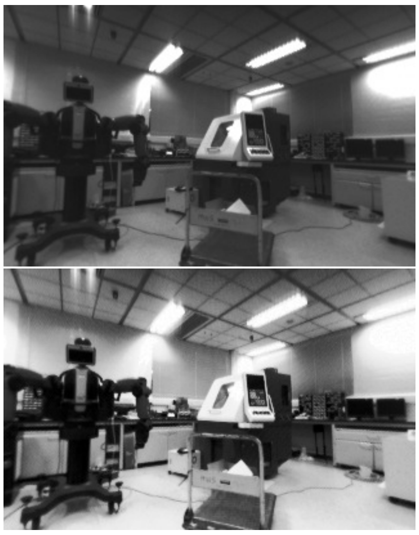

(f) ROI image.

Figure 9.17: Sample of vignetting removed image.

Since traditional stereo calibration methods are hard to apply on the proposed omnidirectional stereo system, and the ground truth is also unknown, we do not have the quantitative comparison. We include indoor static and dynamic scenes, as shown in Fig. 9.18. The matched features are preprocessed by RANSAC filtering to drop some outliers fea-

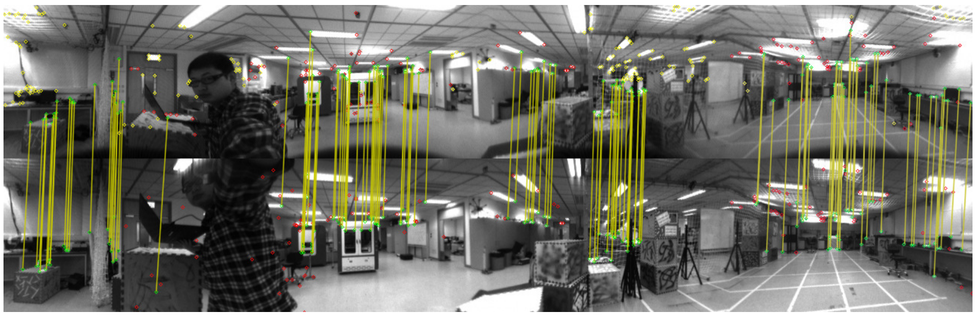
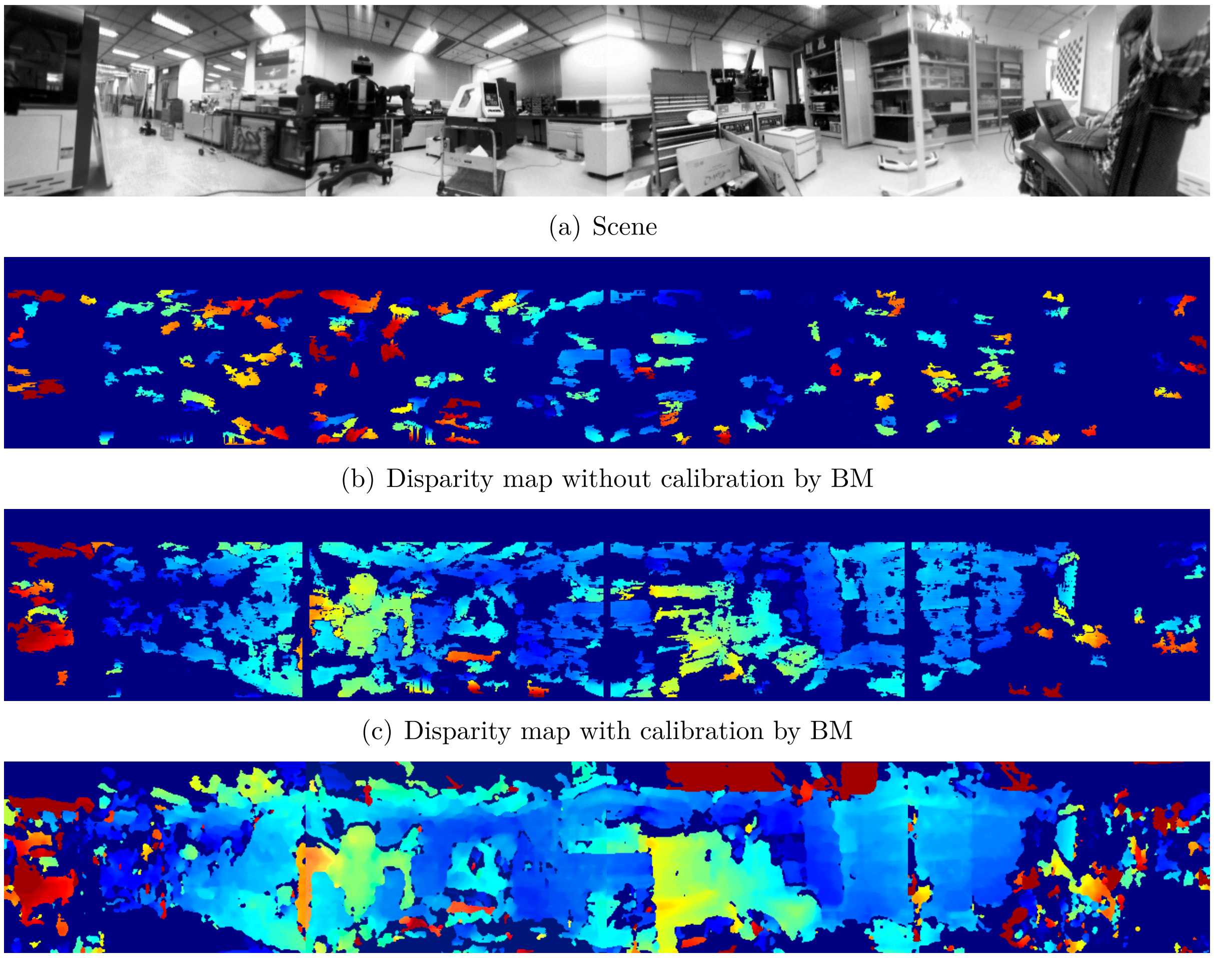


Figure 9.18: Omnidirectional features matching for self-calibration.

tures. We convert the relative rotations $\mathbf{R}$ to Euler angles for graphical presentation, as shown in Table 9.2.



(a) Scene

(b) Disparity map without calibration by BM

(c) Disparity map with calibration by BM

(d) Disparity map with calibration by SGBM

Figure 9.19: Disparity map comparison with extrinsic parameters calibration.

The qualitative results of dense stereo matching of the omnidirectional stereo system using the extrinsic parameters from our marker-less calibration method are shown in Fig. 9.19(d). Our approach significantly improves the stereo matching quality compared to the initially incorrect extrinsic parameters.

### 9.5.4 Screen Calibration Results

We test the screen calibration of two cameras, a perspective camera with limit filed-of-view and a fisheye camera with 235-degree field-of-view.

The calibration system is shown in Fig, 9.7. The calibration observation distribution of the perspective camera is shown in Fig. 9.6. And the calibration observation distribution of the fisheye camera is shown in Fig. 9.20.

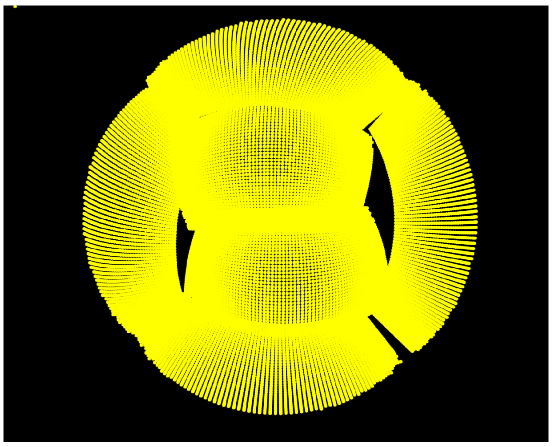
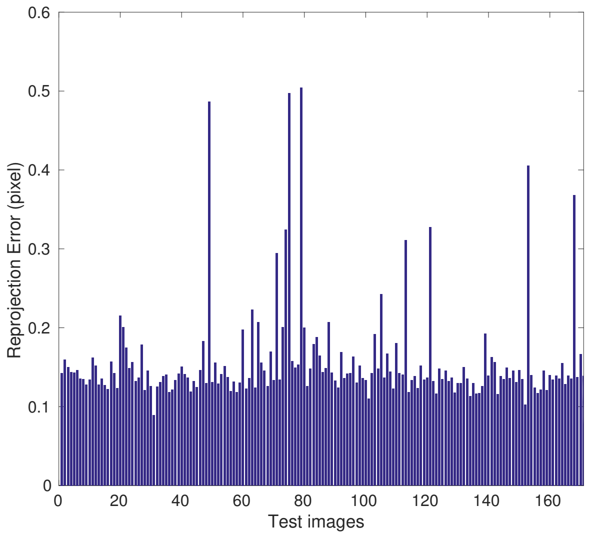


Figure 9.20: The points data distribution of screen calibration of fisheye camera.

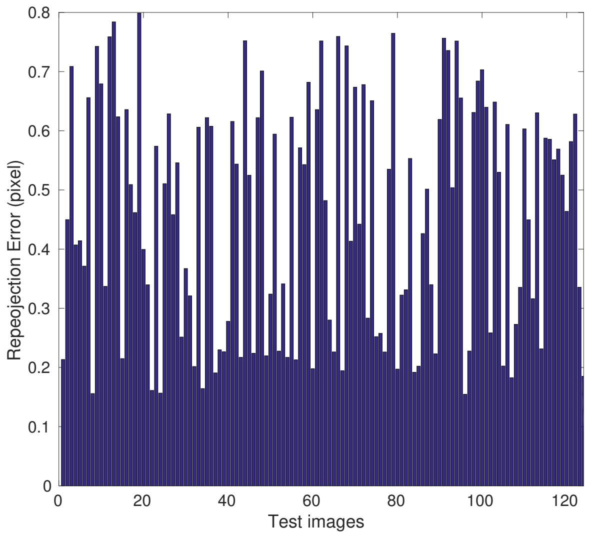
After calibration with the screen system, we test the calibration result. We collect some chessboard images, detect the chessboard points and calculate the reprojection error of the detected points via our screen calibration intrinsic parameters. The reprojection error is shown in Fig, 9.21. For perspective camera, the average reprojection error is 0.15 pixel by 171 chessboard images. For fisheye camera, the average reprojection error is 0.45 pixel by 124 chessboard images. The result of screen calibration is comparable with transitional chessboard calibration.

Table 9.2: Extrinsic Parameters of each Fisheye Camera

| Axis | x | y | z |
|---|---|---|---|
| Rotation (degree) | 177.428 | -0.751351 | -179.008 |
| Translation (m) | -0.0224503 | 0.0194018 | -0.283451 |



(a) Perspective camera

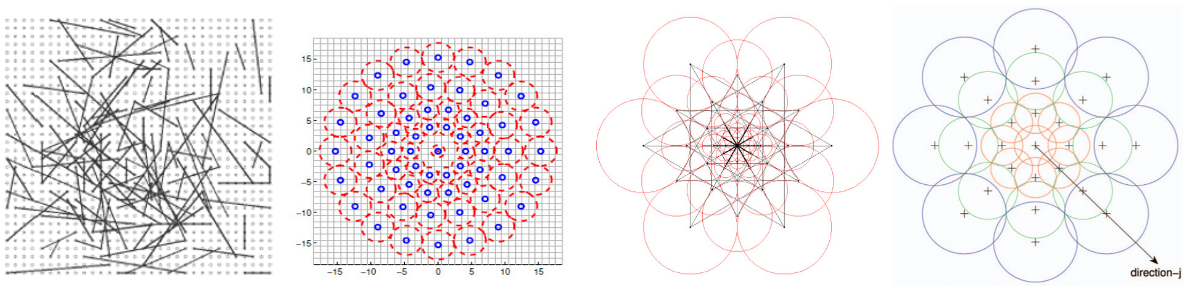


(b) Fisheye camera

Figure 9.21: The reprojection error of the chessboard.

# CHAPTER 10

# FSR:FAST SPHERICAL RETINA KEYPOINT



(a) BRIEF [8] (b) BRISK [40] (c) FREAK [1] (d) DAISY [71]

Figure 10.1: Patch of typical features.

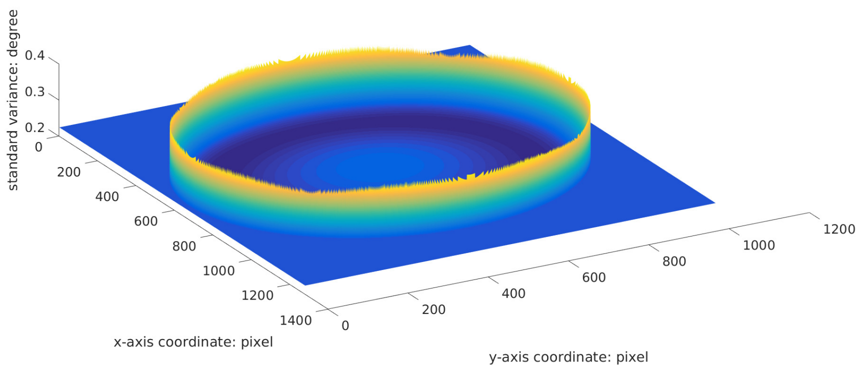
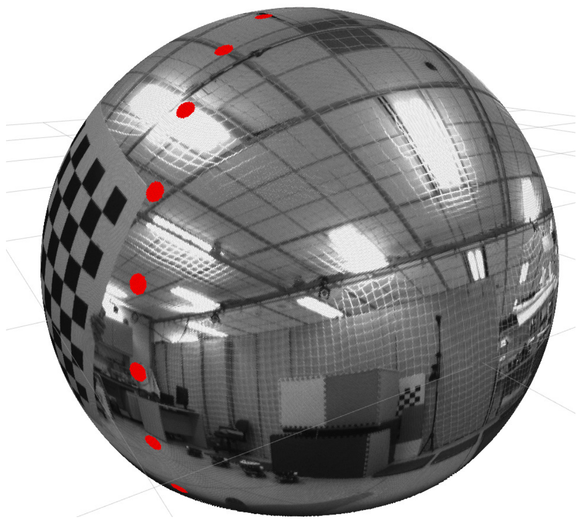


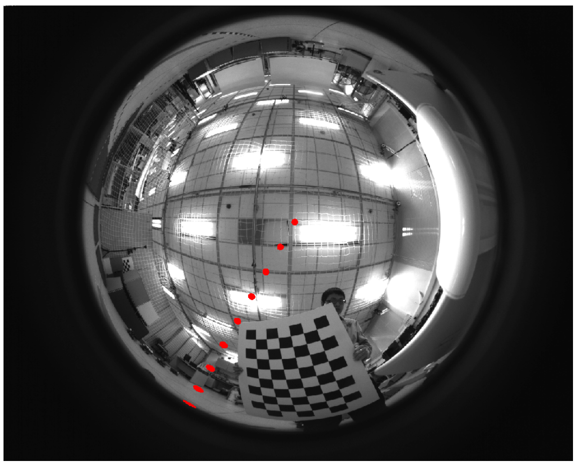
Figure 10.2: 3D graph of the resolution via pixels of the full image from the 240°-FOV camera, $°/pixel$.

traditional feature detectors and descriptors (such as SIFT [48] , SURF [4] , BRIEF [8] , ORB [64] , BRISK [40] , FREAK [1] , and DAISY [71]) are designed to work on the image from perspective cameras. They may not work on spherical images.

## 10.1 Sampling on Unit Sphere

For spherical images, the feature detect and the descriptor should sampling on the unit sphere domain.


(a) The circular patches on the sphere (red blobs)


(b) The circular patches are projected to the spherical images (red ellipses). The circular patches are distorted as ellipses;

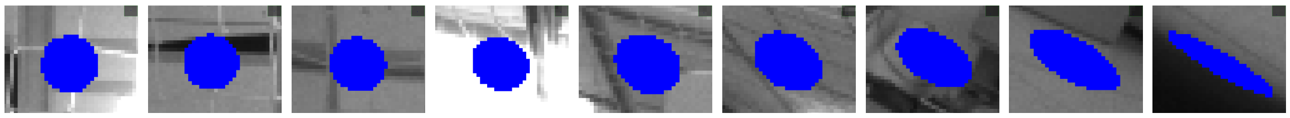

(c) The distorted circular patches at different FOV (0°, 30°, 60°, 90°, 120°, 150°, 180°, 210°, 240°)

Figure 10.3: The illustration of the distorted circle pattern in spherical images.

## 10.2 Feature Detection

### 10.2.1 Accelerated Segment Test

The center of a circular area is used to determine brighter and darker neighbouring pixels. Match in ROI images, on plane, virtual perspective pinhole cameras [51] and [21]. ASIFT: A new framework for fully affine invariant image comparison [54].

SUSAN [70] and FAST [63] use a Bresenham's circle as test pattern. The criteria for a pixel to be a corner according to the accelerated segment test (AST) is as follows: there must be at least S connected pixels on the circle which are brighter or darker than a threshold determined by the center pixel value. The AST applies a minimum difference threshold (t) when comparing the value of a pixel on the circular pattern with the brightness of the nucleus. A large t-value results in few but therefore only strong corners, while a small t-value yields also corners with smoother gradients.

In [63], Rosten uses a machine learning method, to find the best tree based on training data of the environment where FAST is applied. The decision tree learning used by the FAST algorithm builds a ternary tree with possible pixel states "darker", "brighter" and

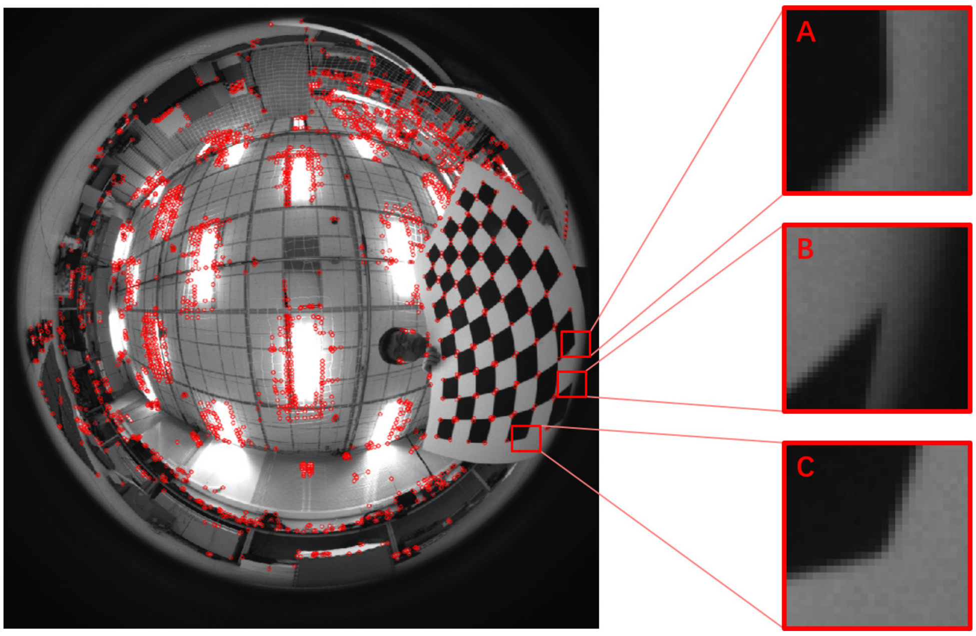
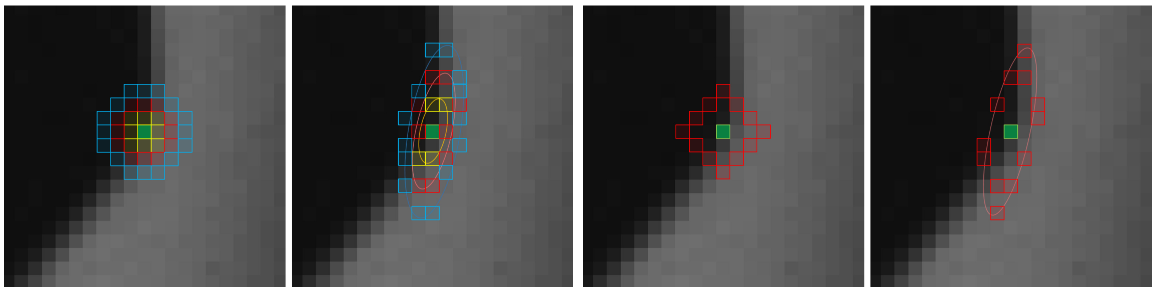


Figure 10.4: A negative-true case of the FAST feature detector ([63]).



(a) Different mask sizes for the AST: 8(yellow), 12 (red) and 16 (blue) pixels mask.

(b) A diamond shaped 12 pixels mask (red).

Figure 10.5: The illustration of the mask of the green nucleus point of AST. Left: raw mask; right: distorted mask.

"similar". Consequently, the configuration space increases by the addition of two more states: "not brighter" (b) and "not darker" (d). Using a similar notion as FAST [63], the state of a pixel relative to the nucleus n, denoted by n → x, is assigned as follows:

$$S_{n\to x} = \begin{cases} d, & \text{if } I_{n\to x} < I_n - t \quad (darker) \\ \bar{d}, & \text{if } I_{n\to x} \nless I_n - t \wedge S'_{n\to x} = u \quad (not\ darker) \\ s, & \text{if } I_{n\to x} \nless I_n - t \wedge S'_{n\to x} = \bar{b} \quad (similar) \\ s, & \text{if } I_{n\to x} \ngtr I_n + t \wedge S'_{n\to x} = \bar{d} \quad (similar) \\ \bar{b}, & \text{if } I_{n\to x} \ngtr I_n + t \wedge S'_{n\to x} = u \quad (not\ brighter) \\ b, & \text{if } I_{n\to x} > I_n + t \quad (brighter) \end{cases} \tag{10.1}$$

where $S'_{n\to x}$ is the preceding state, $I$ is the brightness of a pixel and $u$ means that the state is still unknown. This results in a binary tree representation, as opposed to a ternary tree, allowing a single evaluation at each node.

The challenge of the feature detection on ultra-wide filed-of-view camera and omnidirectional cameras:

1. Nonsensitive to noise due to the vignetting and unbalance of the radiometric response;
2. Rotation invariance and scale invariance due to the distorted patch;

Exist methods:

1. Sampling on the sphere, computation cost;
2. Fixed size patch, not suitable for full images;

### 10.2.2 Distort Patch

Pixel Patch Remap We introduce an pixel patch remap method for calculate the distorted path with as acceptable time. A three-step patch remap is used: 1) to back project each pixel of the standard patch to a unit sphere by $\pi_c^{-1}(\cdot)$, form a spherical path on the unit sphere; 2) to slide the spherical patch to required place on the unit sphere; 3) to project these vector of spherical patch to the image plane by $\pi_c(\cdot)$, form the distorted 2D patch (shown in Fig. 7.1(a)). The transformation between the standard patch to the required patch is:

$$\mathbf{u}^c = \pi_c(\mathbf{R}\pi_c^{-1}(\mathbf{u}_0^c)), \tag{10.2}$$

where $\pi_c(\cdot)$ and $\pi_c^{-1}(\cdot)$ is the projection function and the back projection function of the fisheye camera, with $\mathbf{u}^c$ being the pixel coordinates of the fisheye image. $\mathbf{u}_0^c$ is the standard patch on the pixel coordinates, and $\mathbf{R}$ formulate the related transformation of the spherical patch "slide" on the unit sphere.

Mask Size In [70] it is noted that the mask size does not influence the feature detection as long as there is no more than one feature within the mask.

In [50], the author discussed that the Lager mask size, they are therefore slower for two reasons: 1) the processing of a large pattern is of course computationally more expensive, and 2) they need to evaluate many features for non-maximum suppression. Smaller patterns better preserve the locality constraint of a corner. the features are too close, so that a part of them get lost after non-maximum suppression. If the source pattern is small, it is possible that the mapped points cannot form a pattern in pixel coordinate after distortion.

## 10.3 Feature Descriptor

Coarse-to-fine descriptor We construct our binary descriptor $F$ by thresholding the difference between pairs of receptive fields with their corresponding Gaussian kernel. In other words, F is a binary string formed by a sequence of one-bit Difference of Gaussians (DoG):

$$F = \sum_{0 \le a < N} 2^a T(P_a), \tag{10.3}$$

where $P_a$ is a pair of receptive fields, $N$ is the desired size of the descriptor, and

$$T(P_a) = \begin{cases} 1 & \text{if } (I(P_a^{r_1}) - I(P_a^{r_2})) > 0, \\ 0 & \text{otherwise,} \end{cases} \tag{10.4}$$

with $I(P_a^{r_1})$ is the smoothed intensity of the first receptive field of the pair $P_a$.

Orientation Let $G$ be the set of all the pairs used to compute the local gradients:

$$O = \frac{1}{a} \sum_{P_0 \in G} (I(P_o^{r_1}) - I(P_o^{r_2})) \frac{P_o^{r_1} - P_o^{r_2}}{\|P_o^{r_1} - P_o^{r_2}\|} \tag{10.5}$$

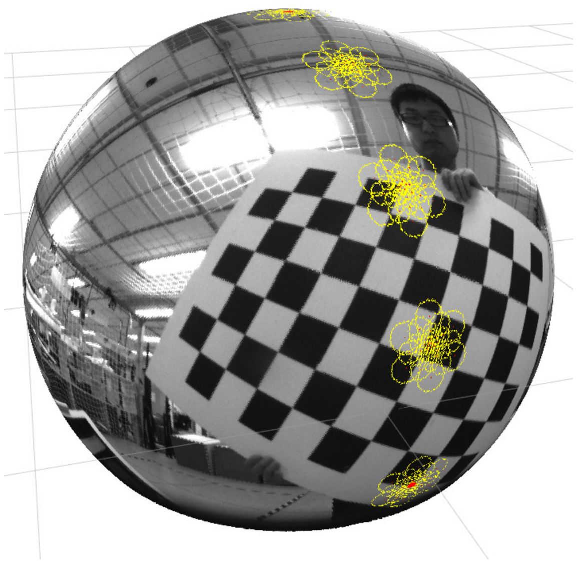
where $M$ is the number of pairs in $G$ and $P_o^{r_i}$ is the 2D vector of the spatial coordinates of the center of receptive field.

## 10.4 Implementation and Experiment

### 10.4.1 Feature Detector

In order to accelerate the detector, the distorted patch should not been calculate every time. We build a patch remap table while load the camera intrinsic parameters, and save the table so that the process do not wast computation and time with once calculation. The typical time cost of the process is shown in Table. 10.1. The implementation is based on the AGAST feature detector from OpenCV 3.0[1].

[1]https://opencv.org/


(a) The circular patches on the sphere (yellow circles)

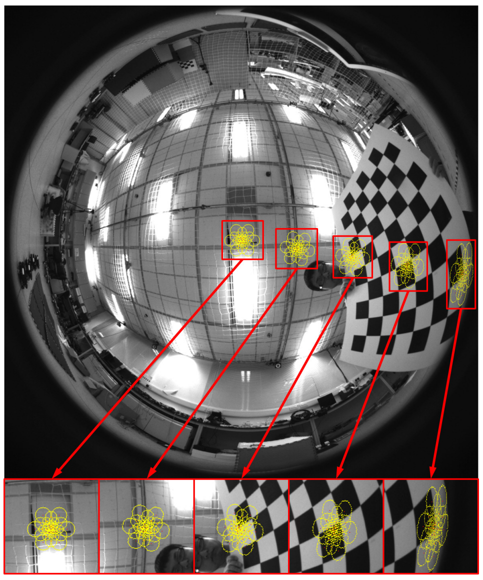

(b) The circular patches are projected to the spherical images (yellow ellipses) at different FOV (0°, 55°, 110°, 165°, 220°).

Figure 10.6: The circular patches are distorted as ellipses.

The typical result of the feature detection is shown in Fig. 10.8. With the consideration of the patch distortion and the camera vignetting, more features on the region of wide filed of view are detected.

The table size is $2 \times size \times row \times col$.

Table 10.1: Approximated Timing Statistics

| Module | Build Table | Read Table | Remove Vignetting | Detect |
|---|---|---|---|---|
| Time Cost(ms) | 8174.5 | 55.4 | 2.6 | 21.0 |

Table 10.2: Performance of Feature Detector

| Method | FAST | AGAST | SAGAST |
|---|---|---|---|
| Time Cost (ms) | 7.1 | 16.9 | 21.0 |
| Number of Feature | 2315 | 2377 | 3549 |

The approximated timing statistics of the feature detector is shown in Table 10.1. And the performance of the feature detector is shown in Table 10.2. The spherical feature detector is fast enough and satisfy real-time running, and the performance is good enough.

The feature detector implementation is open sourced, called SAGAST[2].

[2]https://github.com/gaowenliang/SAGAST

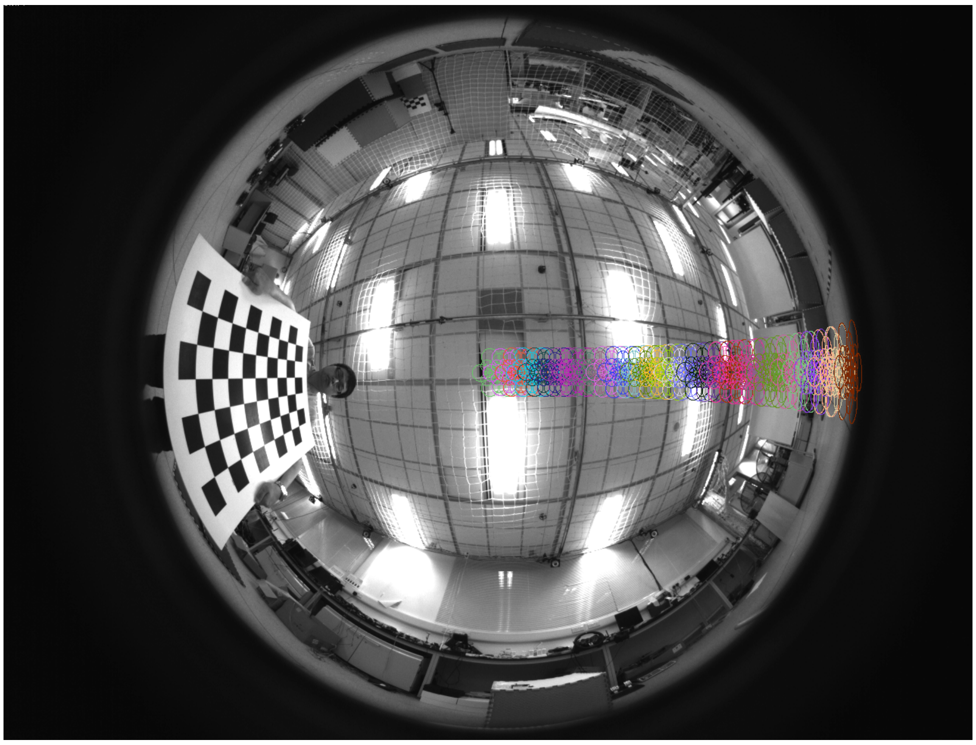


Figure 10.7: The pattern of feature descriptor on the $\phi = 0$ line.

### 10.4.2 Feature Descriptor

The approximated timing statistics of the feature descriptor is shown in Table 10.4. And the performance of the feature descriptor is shown in Table 10.3. The spherical feature is faster with the help of look-up-table (LUT).

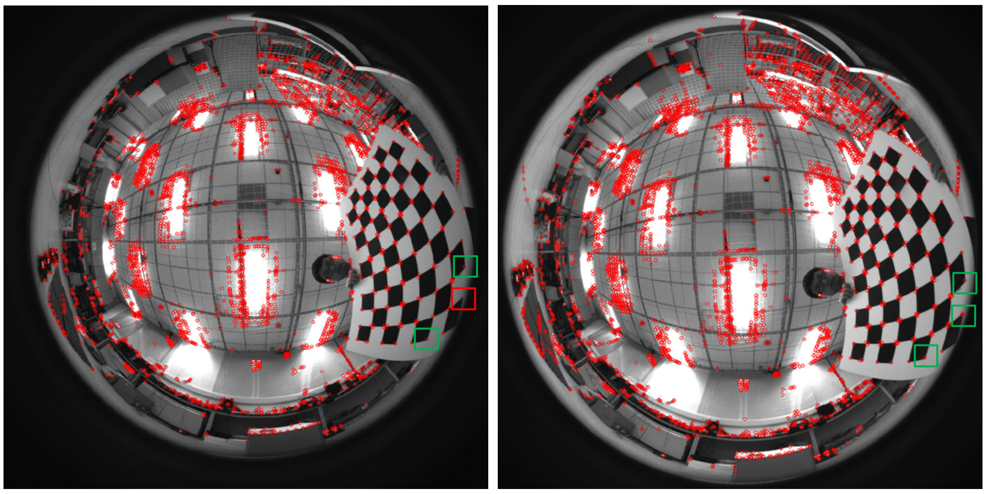
The feature match result between a spherical image and a pinhole image by BRISK, FREAK, ORB, and FSRK are shown in Fig 10.9. Our FSRK feature get more match and the match is better. And the FSRK feature is rotation invariance and distortion invariance.

The feature detector implementation is open sourced, called FSRK[3].

Table 10.3: Performance of Feature Descriptor

| Method | Match Number | Match Number (with RANSAC) |
|---|---|---|
| FREAK(FAST) | 156 | 101 |
| ORB | 555 | 92 |
| BRISK | 91 | 64 |
| FSRK | 406 | 217 |

[3]https://github.com/gaowenliang/fsrk

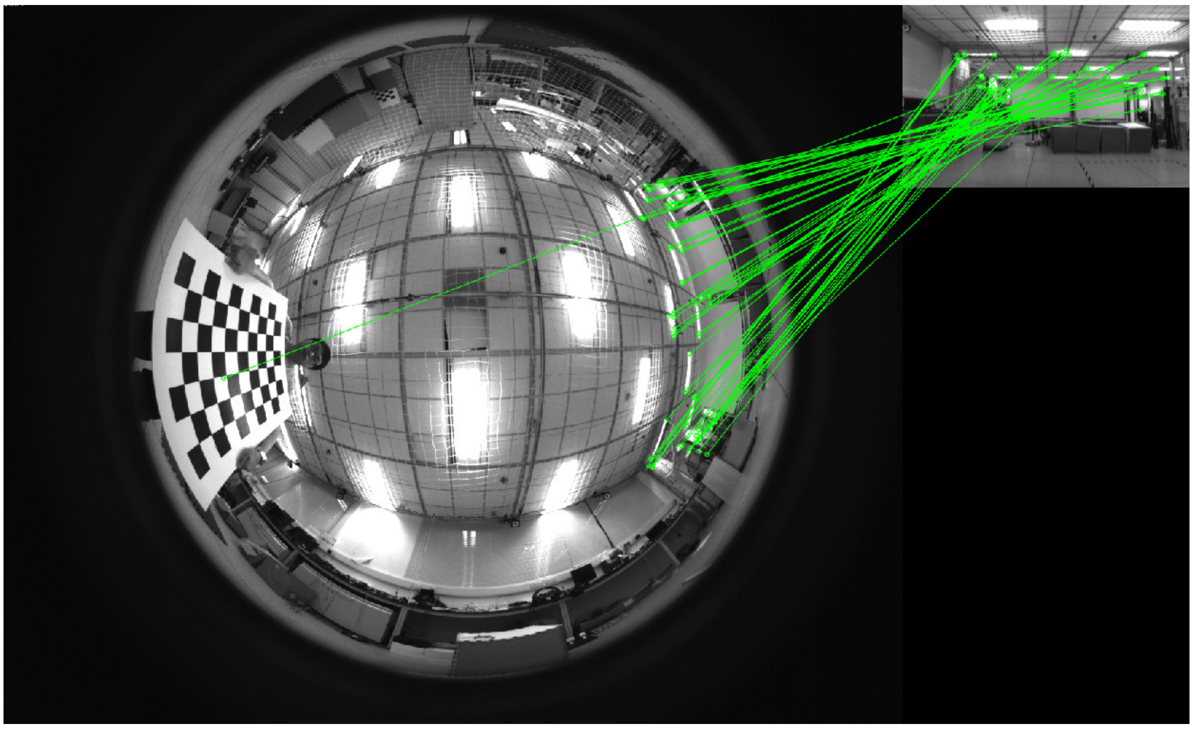


(a) Without vignetting removed (b) With vignetting removed

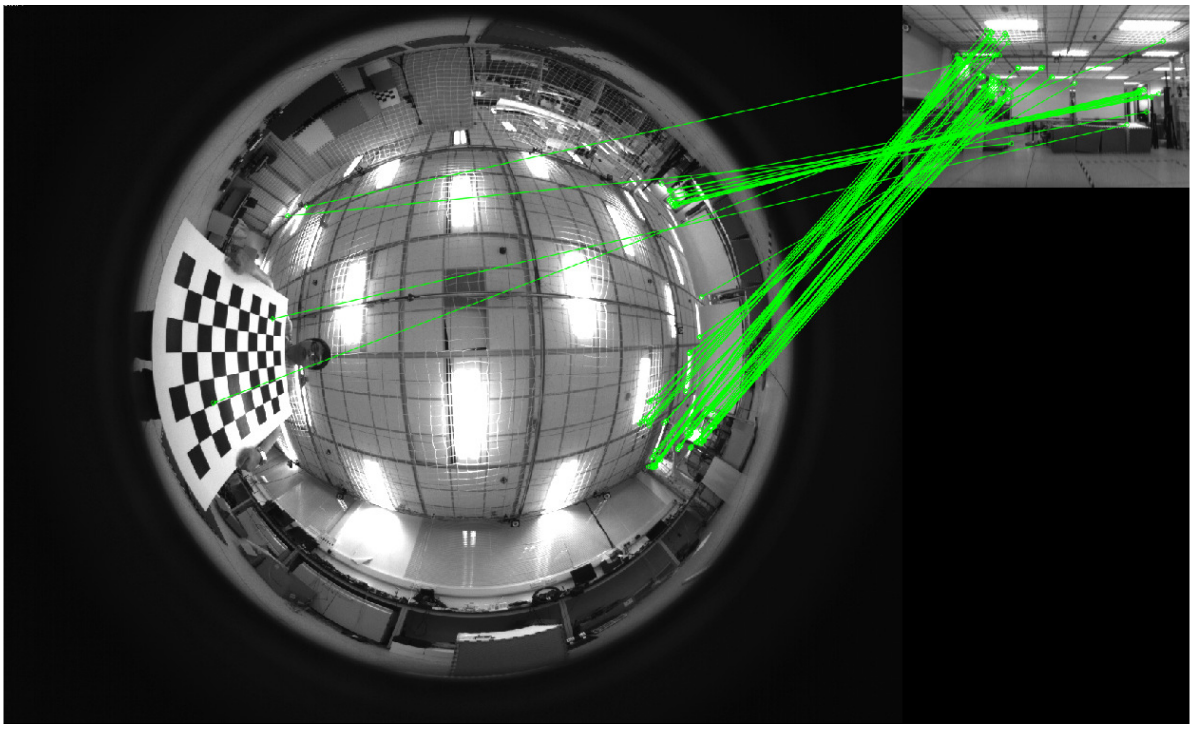
Figure 10.8: The result sample of the feature detector of camera with 240-degree field of view, same image with Fig. 10.4.

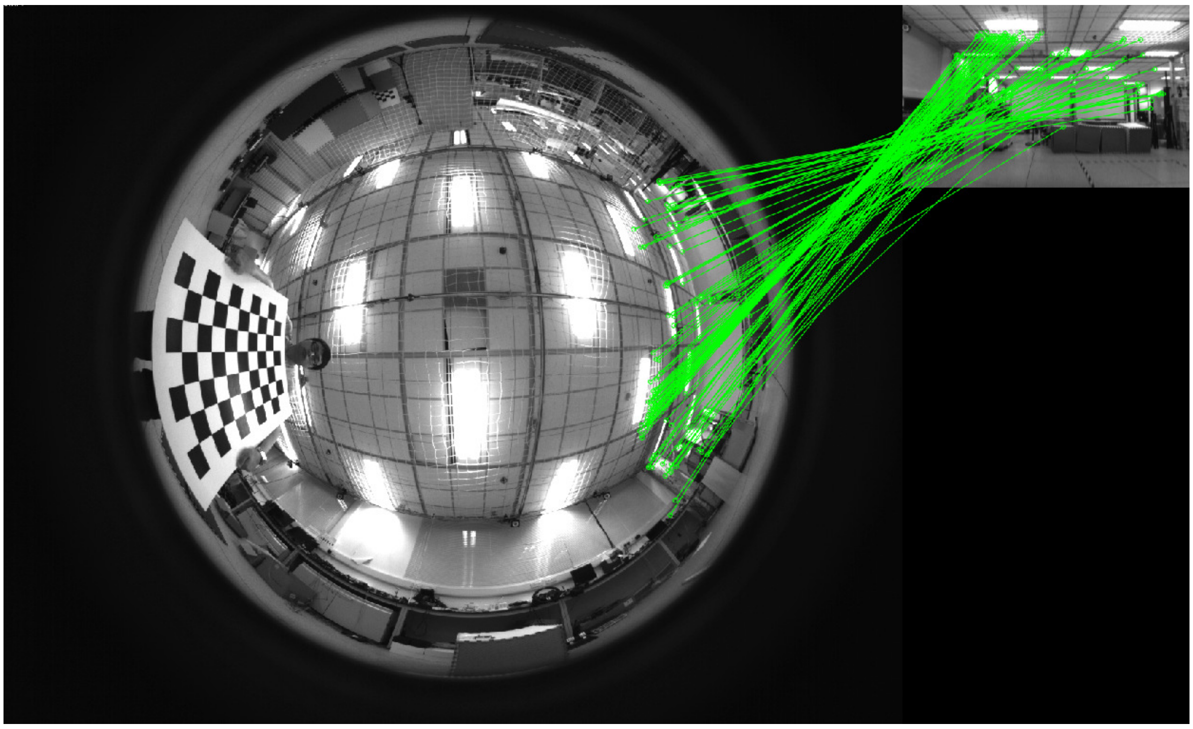
Table 10.4: Time Cost of Feature Descriptor

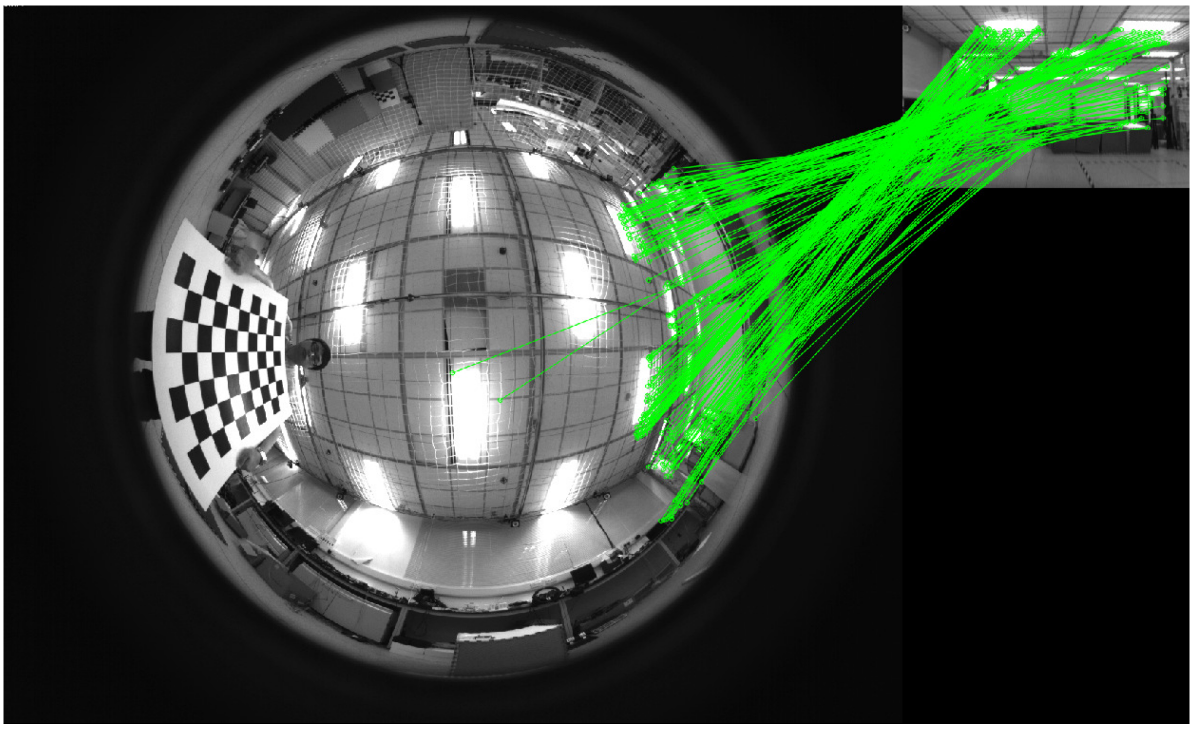
| Method | Time Cost ($ms$) | Time Cost per Feature ($\mu s$) |
|---|---|---|
| FREAK | 12.1 | 13.1 |
| FSRK (without LUT) | 3675.6 | 4457.3 |
| FSRK (with LUT) | 41.8 | 53.5 |



(a) The feature match result between a spherical image and a pinhole image by BRISK


(b) The feature match result between a spherical image and a pinhole image by ORB


(c) The feature match result between a spherical image and a pinhole image by FREAK



(d) The feature match result between a spherical image and a pinhole image by FSRK

Figure 10.9: The feature match result between a spherical image and a pinhole image.

# CHAPTER 11

# CONCLUSION

In this work, we present a real-time dual-fisheye visual-inertial dense mapping and autonomous navigation system. The whole system is implemented on a tight size and light weight quadrotor where all modules are processing onboard and in real time. By properly coordinating three major system modules: state estimation, dense mapping, and trajectory planning, we validate our system in both cluttered indoor and outdoor environments via multiple autonomous flight experiments. A tightly-coupled visual-inertial state estimator is developed for providing high-accuracy odometry, which is used for both feedback control and dense mapping. Our estimator supports self-initialization and is able to online estimate vehicle velocity, metric scale, and IMU biases. Without any prior knowledge of the environment, our dense mapping module extracts a 3D omnidirectional information. After semi-global optimization and post-processing, a dense depth image is calculated and fed into our uncertainty-aware TSDF fusion approach, from which a live dense 3D map is produced. Using this map, our planning module firstly generates an initial collision-free trajectory based on our sampling-based path searching method. Following the trend of rapid increases in mobile computing power, we believe our minimum sensing sensor setup suggests a feasible solution to fully autonomous miniaturized aerial robots.

# REFERENCES

[1] Alexandre Alahi, Raphael Ortiz, and Pierre Vandergheynst. Freak: Fast retina keypoint. In Computer vision and pattern recognition (CVPR), 2012 IEEE conference on, pages 510–517. Ieee, 2012.

[2] Sergey V Alexandrov, Johann Prankl, Michael Zillich, and Markus Vincze. Calibration and correction of vignetting effects with an application to 3d mapping. In Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on, pages 4217–4223. IEEE, 2016.

[3] Anup Basu and Sergio Licardie. Alternative models for fish-eye lenses. Pattern recognition letters, 16(4):433–441, 1995.

[4] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In European conference on computer vision, pages 404–417. Springer, 2006.

[5] Michael Bloesch, Sammy Omari, Marco Hutter, and Roland Siegwart. Robust visual inertial odometry using a direct ekf-based approach. In Proc. of the IEEE/RSJ Int. Conf. on Intell. Robots and Syst., pages 298–304. IEEE, 2015.

[6] Christian Brauer-Burchardt and Klaus Voss. A new algorithm to correct fish-eye- and strong wide-angle-lens-distortion from single images. In Proc. of the IEEE Intl. Conf. on Image Processing, volume 1, pages 225–228. IEEE, 2001.

[7] Duane C Brown. Decentering distortion of lenses. Photometric Engineering, 32(3):444–462, 1966.

[8] Michael Calonder, Vincent Lepetit, Christoph Strecha, and Pascal Fua. Brief: Binary robust independent elementary features. In European conference on computer vision, pages 778–792. Springer, 2010.

[9] David Caruso, Jakob Engel, and Daniel Cremers. Large-scale direct slam for omnidirectional cameras. In Proc. of the IEEE/RSJ Int. Conf. on Intell. Robots and Syst., pages 141–148. IEEE, 2015.

[10] Shreyansh Daftry, Sam Zeng, Arbaaz Khan, Debadeepta Dey, Narek Melik-Barkhudarov, J Andrew Bagnell, and Martial Hebert. Robust monocular flight in cluttered outdoor environments. arXiv preprint arXiv:1604.04779, 2016.

[11] Jeffrey Delmerico and Davide Scaramuzza. A benchmark comparison of monocular visual-inertial odometry algorithms for flying robots. In Proc. of the IEEE/RSJ Int. Conf. on Intell. Robots and Syst., volume 10, page 20, 2018.

[12] Frederic Devernay and Olivier Faugeras. Straight lines have to be straight. Machine vision and applications, 13(1):14–24, 2001.

[13] Wenchao Ding, Wenliang Gao, Kaixuan Wang, and Shaojie Shen. Trajectory replanning for quadrotors using kinodynamic search and elastic optimization. In Proc. of the IEEE Intl. Conf. on Robot. and Autom. IEEE, 2018.

[14] Wenchao. Ding, Wenliang. Gao, Kaixuan. Wang, and Shaojie. Shen. Trajectory replanning for quadrotors using kinodynamic search and elastic optimization. In Proc. of the IEEE Intl. Conf. on Robot. and Autom., Oct. 2018.

[15] Naser El-Sheimy, Haiying Hou, and Xiaoji Niu. Analysis and modeling of inertial sensors using allan variance. IEEE Transactions on instrumentation and measurement, 57(1):140–149, 2008.

[16] Jakob Engel, Thomas Schöps, and Daniel Cremers. Lsd-slam: Large-scale direct monocular slam. In Proc. of the Euro. Conf. on Computer Vision, pages 834–849. Springer, 2014.

[17] Matthias Faessler, Flavio Fontana, Christian Forster, Elias Mueggler, Matia Pizzoli, and Davide Scaramuzza. Autonomous, vision-based flight and live dense 3d mapping with a quadrotor micro aerial vehicle. J. Field Robot., 1, 2015.

[18] Gao Fei, WU William, Lin Yi, and Shen Shaojie. Online safe trajectory generation for quadrotors using fast marching method and bernstein basis polynomial. In Proc. of the IEEE Intl. Conf. on Robot. and Autom. IEEE, 2018.

[19] Christian Forster, Matia Pizzoli, and Davide Scaramuzza. Svo: Fast semi-direct monocular visual odometry. In Proc. of the IEEE Intl. Conf. on Robot. and Autom., pages 15–22. IEEE, 2014.

[20] Friedrich Fraundorfer, Lionel Heng, Dominik Honegger, Gim Hee Lee, Lorenz Meier, Petri Tanskanen, and Marc Pollefeys. Vision-based autonomous mapping and exploration using a quadrotor mav. In Proc. of the IEEE/RSJ Intl. Conf. on Intell. Robots and Syst., pages 4557–4564. IEEE, 2012.

[21] Wenliang Gao and Shaojie Shen. Dual-fisheye omnidirectional stereo. In Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on, pages 6715–6722. IEEE, 2017.

[22] Christopher Geyer and Kostas Daniilidis. A unifying theory for central panoramic systems and practical implications. In Proc. of the Euro. Conf. on Computer Vision, pages 445–461. Springer, 2000.

[23] Alessandro Giusti, Jérôme Guzzi, Dan C Cireşan, Fang-Lin He, Juan P Rodríguez, Flavio Fontana, Matthias Faessler, Christian Forster, Jürgen Schmidhuber, Gianni Di Caro, et al. A machine learning approach to visual perception of forest trails for mobile robots. IEEE Robotics and Automation Letters, 1(2):661–667, 2016.

[24] Joshua Gluckman, Shree K Nayar, and Keith J Thoresz. Real-time omnidirectional and panoramic stereo. In Proc. of Image Understanding Workshop, volume 1, pages 299–303. Citeseer, 1998.

[25] Pascal Gohl, Dominik Honegger, Sammy Omari, Markus Achtelik, Marc Pollefeys, and Roland Siegwart. Omnidirectional visual obstacle detection using embedded fpga. In Proc. of the IEEE/RSJ Intl. Conf. on Intell. Robots and Syst., pages 3938–3943. IEEE, 2015.

[26] Richard Hartley and Andrew Zisserman. Multiple view geometry in computer vision. Cambridge university press, 2003.

[27] Lionel Heng, Bo Li, and Marc Pollefeys. Camodocal: Automatic intrinsic and extrinsic calibration of a rig with multiple generic cameras and odometry. In Proc. of the IEEE/RSJ Intl. Conf. on Intell. Robots and Syst., pages 1793–1800. IEEE, 2013.

[28] J. A. Hesch, D. G. Kottas, S. L. Bowman, and S. I. Roumeliotis. Consistency analysis and improvement of vision-aided inertial navigation. IEEE Trans. Robot., 30(1):158–176, February 2014.

[29] Heiko Hirschmuller. Stereo processing by semiglobal matching and mutual information. IEEE Trans. Pattern Anal. Mach. Intell., 30(2):328–341, 2008.

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

[31] A. S. Huang, A. Bachrach, P. Henry, M. Krainin, D. Maturana, D. Fox, and N. Roy. Visual odometry and mapping for autonomous flight using an RGB-D camera. In Proc. of the Int. Sym. of Robot. Research, Flagstaff, AZ, August 2011.

[32] Ciarán Hughes, Patrick Denny, Edward Jones, and Martin Glavin. Accuracy of fish-eye lens models. Appl. Opt., 49(17):3338–3347, 2010.

[33] Ralph Jacobson, Sidney Ray, Geoffrey G Attridge, and Norman Axford. Manual of Photography. Taylor & Francis, 2000.

[34] Carlos Jaramillo, Roberto G. Valenti, Ling Guo, and Jizhong Xiao. Design and analysis of a single-camera omnistereo sensor for quadrotor micro aerial vehicles (mavs). Sensors, 16(2):7–15, 2016.

[35] Kenichi Kanatani. Calibration of ultrawide fisheye lens cameras by eigenvalue minimization. IEEE Trans. Pattern Anal. Mach. Intell., 35(4):813–822, 2013.

[36] Kenichi Kanatani, Yasuyuki Sugaya, and Yasushi Kanazawa. Ellipse fitting for computer vision: implementation and applications. Synthesis Lectures on Computer Vision, 6(1):1–141, 2016.

[37] Matthew Klingensmith, Ivan Dryanovski, Siddhartha Srinivasa, and Jizhong Xiao. Chisel: Real time large scale 3d reconstruction onboard a mobile device using spatially hashed signed distance fields. In Proc. of Robot.: Sci. and Syst., 2015.

[38] Igor Labutov, Carlos Jaramillo, and Jizhong Xiao. Generating near-spherical range panoramas by fusing optical flow and stereo from a single-camera folded catadioptric rig. Machine Vision and Applications, 24(1):133–144, 2013.

[39] S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, and P. Furgale. Keyframe-based visual–inertial odometry using nonlinear optimization. Int. J. Robot. Research, 34(3):314–334, March 2014.

[40] Stefan Leutenegger, Margarita Chli, and Roland Y Siegwart. Brisk: Binary robust invariant scalable keypoints. In Computer Vision (ICCV), 2011 IEEE International Conference on, pages 2548–2555. IEEE, 2011.

[41] Stefan Leutenegger, Simon Lynen, Michael Bosse, Roland Siegwart, and Paul Furgale. Keyframe-based visual–inertial odometry using nonlinear optimization. The International Journal of Robotics Research, 34(3):314–334, 2015.

[42] M. Li and A.I. Mourikis. High-precision, consistent EKF-based visual-inertial odometry. Int. J. Robot. Research, 32(6):690–711, May 2013.

[43] Weiming Li and You Fu Li. Single-camera panoramic stereo imaging system with a fisheye lens and a convex mirror. Optics Express, 19(7):5855–5867, 2011.

[44] Yi Lin, Fei Gao, Tong Qin, Wenliang Gao, Tianbo Liu, William Wu, Zhenfei Yang, and Shaojie Shen. Autonomous aerial navigation using monocular visual-inertial fusion. J. Field Robot., 35(1):23–51, 2018.

[45] Yonggen Ling and Shaojie Shen. High-precision online markerless stereo extrinsic calibration. In Proc. of the IEEE/RSJ Intl. Conf. on Intell. Robots and Syst., pages 1771–1778. IEEE, 2016.

[46] Yonggen Ling and Shaojie Shen. High-precision online markerless stereo extrinsic calibration. In Proc. of the IEEE/RSJ Int. Conf. on Intell. Robots and Syst., pages 1771–1778, Oct 2016.

[47] William E Lorensen and Harvey E Cline. Marching cubes: A high resolution 3d surface construction algorithm. In ACM siggraph computer graphics, volume 21, pages 163–169. ACM, 1987.

[48] David G Lowe. Distinctive image features from scale-invariant keypoints. International journal of computer vision, 60(2):91–110, 2004.

[49] B. D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In Proc. of the Intl. Joint Conf. on Artificial Intelligence, pages 24–28, Vancouver, Canada, August 1981.

[50] Elmar Mair, Gregory D. Hager, Darius Burschka, Michael Suppa, and Gerhard Hirzinger. Adaptive and generic corner detection based on the accelerated segment test. In Proceedings of the European Conference on Computer Vision (ECCV'10), September 2010.

[51] Thomas Mauthner, Friedrich Fraundorfer, and Horst Bischof. Region matching for omnidirectional images using virtual camera planes. In Proc. of Computer Vision Winter Workshop. Citeseer, 2006.

[52] Christopher Mei and Patrick Rives. Single view point omnidirectional camera calibration from planar grids. In Proc. of the IEEE Intl. Conf. on Robot. and Autom., pages 3945–3950. IEEE, 2007.

[53] Jeff Michels, Ashutosh Saxena, and Andrew Y Ng. High speed obstacle avoidance using monocular vision and reinforcement learning. In Proc. of the Intl. Conf. on Machine learning, pages 593–600. ACM, 2005.

[54] Jean-Michel Morel and Guoshen Yu. Asift: A new framework for fully affine invariant image comparison. SIAM journal on imaging sciences, 2(2):438–469, 2009.

[55] Anastasios I Mourikis and Stergios I Roumeliotis. A multi-state constraint kalman filter for vision-aided inertial navigation. In Proc. of the IEEE Int. Conf. on Robot. and Autom., pages 3565–3572. IEEE, 2007.

[56] Richard A Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J Davison, Pushmeet Kohi, Jamie Shotton, Steve Hodges, and Andrew Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In Proc. of the Intl. Sym. on Mixed and augmented reality, pages 127–136. IEEE, 2011.

[57] David Nistér. An efficient solution to the five-point relative pose problem. IEEE transactions on pattern analysis and machine intelligence, 26(6):756–770, 2004.

[58] Kyel Ok, W Nicholas Greene, and Nicholas Roy. Simultaneous tracking and rendering: Real-time monocular localization for mavs. In Proc. of the IEEE Intl. Conf. on Robot. and Autom., pages 4522–4529. IEEE, 2016.

[59] Helen Oleynikova, Zachary Taylor, Marius Fehr, Juan Nieto, and Roland Siegwart. Voxblox: Incremental 3d euclidean signed distance fields for on-board mav planning. arXiv preprint arXiv:1611.03631, 2016.

[60] Matia Pizzoli, Christian Forster, and Davide Scaramuzza. Remode: Probabilistic, monocular dense reconstruction in real time. In Proc. of the IEEE Intl. Conf. on Robot. and Autom., pages 2609–2616. IEEE, 2014.

[61] Tong Qin, Peiliang Li, and Shaojie Shen. Vins-mono: A robust and versatile monocular visual-inertial state estimator. arXiv preprint arXiv:1708.03852, 2017.

[62] Milad Ramezani, Kourosh Khoshelham, and Clive Fraser. Pose estimation by omnidirectional visual-inertial odometry. Robotics and Autonomous Systems, 2018.

[63] 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, 32(1):105–119, 2010.

[64] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: An efficient alternative to sift or surf. In Computer Vision (ICCV), 2011 IEEE international conference on, pages 2564–2571. IEEE, 2011.

[65] D. Scaramuzza, M.C. Achtelik, L. Doitsidis, F. Fraundorfer, E.B. Kosmatopoulos, A. Martinelli, M.W. Achtelik, M. Chli, S.A. Chatzichristofis, L. Kneip, D. Gurdan, L. Heng, G.H. Lee, S. Lynen, L. Meier, M. Pollefeys, A. Renzaglia, R. Siegwart, J.C. Stumpf, P. Tanskanen, C. Troiani, and S. Weiss. Vision-controlled micro flying robots: from system design to autonomous navigation and mapping in GPS-denied environments. IEEE Robot. Autom. Mag., 21(3), 2014.

[66] Davide Scaramuzza, Agostino Martinelli, and Roland Siegwart. A flexible technique for accurate omnidirectional camera calibration and structure from motion. In Proc. of the IEEE Intl. Conf. on Computer Vision Systems, pages 45–45. IEEE, 2006.

[67] Korbinian Schmid, Philipp Lutz, Teodor Tomić, Elmar Mair, and Heiko Hirschmüller. Autonomous vision-based micro air vehicle for indoor and outdoor navigation. J. Field Robot., 31(4):537–570, 2014.

[68] Eric L Schwartz. Computational anatomy and functional architecture of striate cortex: a spatial mapping approach to perceptual coding. Vision research, 20(8):645–669, 1980.

[69] S. Shen, N. Michael, and V. Kumar. Tightly-coupled monocular visual-inertial fusion for autonomous flight of rotorcraft MAVs. In Proc. of the IEEE Int. Conf. on Robot. and Autom., Seattle, WA, May 2015.

[70] Stephen M Smith and J Michael Brady. Susan—a new approach to low level image processing. International journal of computer vision, 23(1):45–78, 1997.

[71] Engin Tola, Vincent Lepetit, and Pascal Fua. Daisy: An efficient dense descriptor applied to wide-baseline stereo. IEEE transactions on pattern analysis and machine intelligence, 32(5):815–830, 2010.

[72] Bill Triggs, Philip F McLauchlan, Richard I Hartley, and Andrew W Fitzgibbon. Bundle adjustment—a modern synthesis. In International workshop on vision algorithms, pages 298–372. Springer, 1999.

[73] Zhenfei Yang and Shaojie Shen. Monocular visual–inertial state estimation with online initialization and camera–imu extrinsic calibration. IEEE Transactions on Automation Science and Engineering, 14(1):39–51, 2017.

[74] Jure Zbontar and Yann LeCun. Stereo matching by training a convolutional neural network to compare image patches. J. Mach. Learn. Res., 17(1-32):2, 2016.

[75] Radiant Zemax. Optical design program–user's manual, 2012.

[76] Ji Zhang and Sanjiv Singh. Loam: Lidar odometry and mapping in real-time. In Robotics: Science and Systems, volume 2, page 9, 2014.

[77] Fuqiang Zhou, Xinghua Chai, Xin Chen, and Ya Song. Omnidirectional stereo vision sensor based on single camera and catoptric system. Appl. Opt., 55(25):6813–6820, 2016.

[78] Guyue Zhou, Lu Fang, Ketan Tang, Honghui Zhang, Kai Wang, and Kang Yang. Guidance: A visual sensing platform for robotic applications. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, pages 9–14, 2015.