Visual-Inertial Simultaneous Localization and Mapping with Multiple Cameras

Pak Hong Chui

Munich 2020

Visual-Inertial Gleichzeitige Lokalisierung und Mappen mit mehreren Kameras

Pak Hong Chui

Munich 2020

Visual-Inertial Simultaneous Localization and Mapping with Multiple Cameras

Pak Hong Chui

Master thesis
an der Fakultät für Physik
der Ludwig–Maximilians–Universität
München

vorgelegt von
Pak Hong Chui
aus Hong Kong

Erster Supervisor: Dr. Vladyslav Usenko
Zweiter Supervisor: Dr. Torsten Enßlin

München, den 30. April 2020

Declaration:

I hereby declare that this thesis is my own work, and that I have not used any sources and aids other than those stated in the thesis.

München,

Author's signature

Contents

List of Figures

List of Tables

Abstract

Autonomous driving relies on accurate and efficient state estimation. Owing to the fact that sensors are becoming smaller, lighter and cheaper, the fusion of different kinds of sensors becomes more applicable. Nowadays, these techniques have been used on obstacle detection, path planning, autonomous driving, and even VR gaming.

Simultaneous localization and mapping, so-called SLAM, can be classified into direct method and feature-based method: the former is based on optimizing the photometric error and the latter is based on optimizing the reprojection error of features. Both methods have their own advantages. Direct method uses every pixel in images to estimate the state and thus more accurate, and the feature-based method builds features on some local sub-images and thus more robust to distortion and illumination.

In this thesis, a novel approach for state estimation by using visual inertial simultaneous localization and mapping (VI-SLAM) with multiple cameras is presented. It combines the advantages of feature-based method with optical flow, and is able to operate in real time. The major challenge is to find an efficient way to transmit the information from visual-inertial odometry to the mapper in order to create a consistent gravity aligned map. In this thesis, the corresponding method and the evaluation would be discussed in details.

Acknowledgements

Here, I would like to express my gratitude and appreciation to people who helped, motivated and supported me during my studying of my master. First, my first supervisor Dr. Vladyslav Usenko, who gave me an opportunity to finish my thesis under his supervision. Being a non-computer science major graduate student, I lacked the intensive training in algorithm and programming, however, he was still willing to teach me back and give me a lot of guidelines and comments on my research. I could not thank him enough for all of his patience and willingness to motivate and support me on my thesis.

My second supervisor Dr. Torsten Enßlin who brought me into the area of information theory and computer vision. He gave a great lecture "Information field theory" and seminar in "Information theory", driving me to think about the research area outside physics.

Prof. Dr. Gerhard Buchalla, Prof. Dr. Matthias Punk and Prof. Dr. Ilka Brunner for writing me a reference letter to apply the scholarship which was extremely important for me to maintain my study and living in Munich.

Last but not least, I would like to thank my family for their support. When I was struggling whether I should study a master degree abroad, they always encouraged me to be brave and say "now or never".

Chapter 1

Introduction

1.1 What is computer vision?

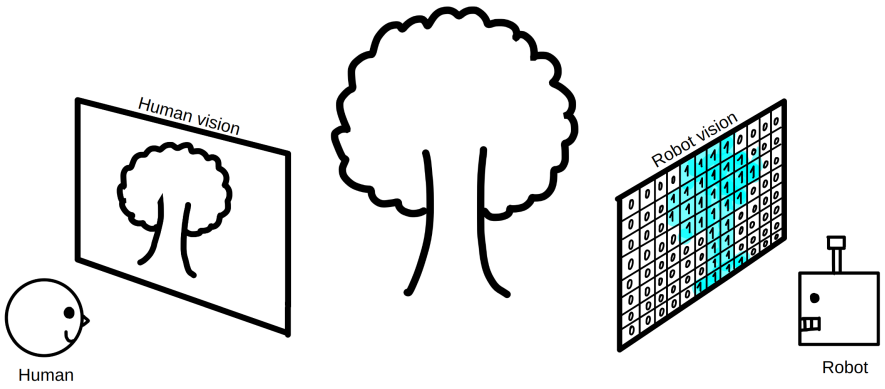


Figure 1.1: Human vision vs Computer vision

Computer vision is one of the important areas of computer science which aims to teach machine how to understand images and extract information from them. We hope that we can use cameras and computers to replace human eyes and brains to do recognition, recitification, relocalization, etc. By implementing algorithms (the knowledge or theory we deliver to computer), which can turn measurements to another form of information such as depth measurement to camera pose. The process of scene understanding is a hierarchy as follows.

Low-level vision: de-noising, bluring, de-bluring, edge detection, keypoints extraction. See fig 1.2

Mid-level vision: image segmentation, depth estimation, keypoint description, optical flow estimation. See fig 1.3.

High-level vision: classification of segments, object recognition, mapping and localization. See fig 1.4.

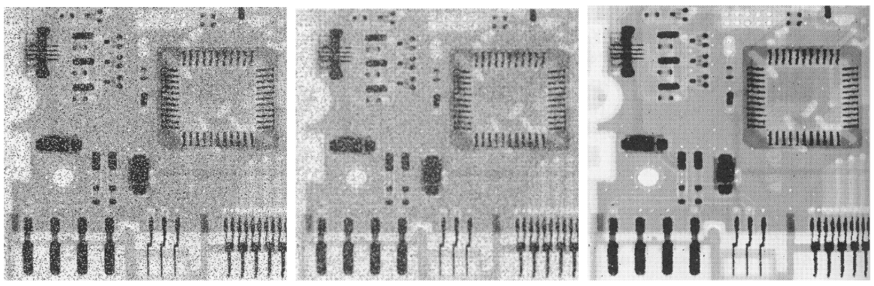


Figure 1.2: left: origin, middle: Gaussian filter, right: median filter (Original image courtesy of Mr.Joseph E. Pascente. Lixi, Inc)

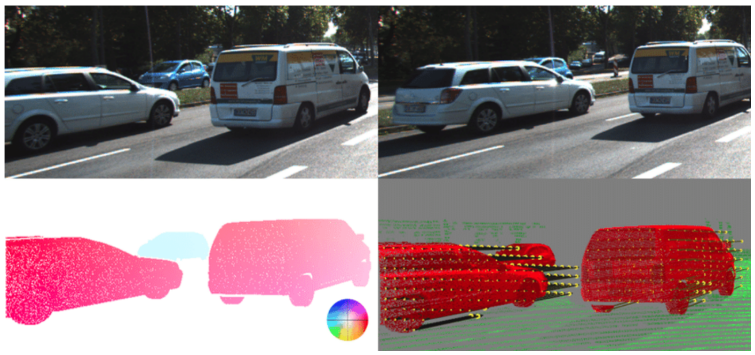


Figure 1.3: Example of Optical flow[26]

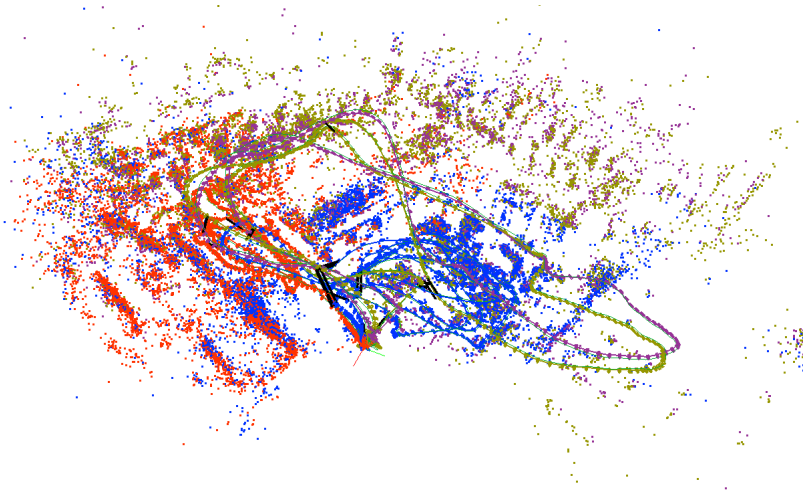


Figure 1.4: Example of SLAM

1.2 Physics and computer vision

Computer vision is also a science to understand the physical world through different kinds of sensors (camera, radar, lidar, Inertial measurement unit (IMU)), therefore the idea and the mathematics used in computer vision are very similar in the following aspects.

Loss function: In physics, equation of motion can be derived from loss function including Klein Gordan equation for spin-0 particle, Dirac equation for spin-$\frac{1}{2}$ particle, etc. In computer vision, we also construct loss function (error function) to find the best state to minimize the error function. The state in computer vision can be poses of camera, intrinsic parameters of camera, landmarks' position, etc.

Triangulation: In physics, we always use it in measuring the distance between the Earth and a star by parallex. The parallex is obtained by measuring the view of angle on one day and on a day six months later. The difference of view of angle yields the distance. It is similar in computer vision, we triangulate those features which exist in at least two images yielding 3d points. Those 3d points can be used to localize new images.

Group theory-Lie group and Lie algebra: In physics, we use this to examine a system if it has any symmetry. For example, If a Hamiltonian remains unchanged under SO(3)/U(1) transformation, then the angular momentum/charge is conserved. We also know that the eigen states are degenerated under the symmetry. In computer vision, Lie group and Lie algebra are mainly used in finding optimizer of an error function.

Other physics concepts: For instance, the concept of torque is used in calculating the orientation of an image patch in order to do feature matching correctly.

1.3 Motivation

One of the biggest challenges in robot navigation is how a robot navigates and localizes itself in an unknown environment. One simple solution is dead reckoning, based on the previous status and the current velocity to estimate the current status. Nowadays, people use Inertial Measurement Unit (IMU) to measure the acceleration which makes dead reckoning more accurate, however these techniques cannot avoid the error to accumulate to the next state estimation, and the error becomes huge over time.

People add different measures in the calculation to reduce the drift such as positions of significant landmarks. In ancient navigation, sailors would use different instruments to determine the position of islands to correct the estimation of velocity.

We can either use the landmarks with global coordinate or the landmarks that have observed in early times in order to remove the drift completely, just as sailors use lighthouses

or stars to locate themselves, or in a maze, people remember the place and then recognize that if they revisit it.

In this thesis, we will try to turn Basalt-vio to a slam system to remove drift by loop-closure and extend it to online multi-camera localization based on the maps created by those cameras.

1.4 Related work

Simultaneous Localization And Mapping (SLAM) is an algorithm to track the current position while building a map at the same time for later use[7][2]. During 1986-2004, probabilistic formulations for SLAM had been developed, for instance, Extended Kalman Filter[10] and Particle Filter[1]. During 2004-2016, the slam research was focused on the fundamental property of SLAM, such as observability, convergence and consistency. Today, 3d object detection using deep learning is becoming a hot topic.

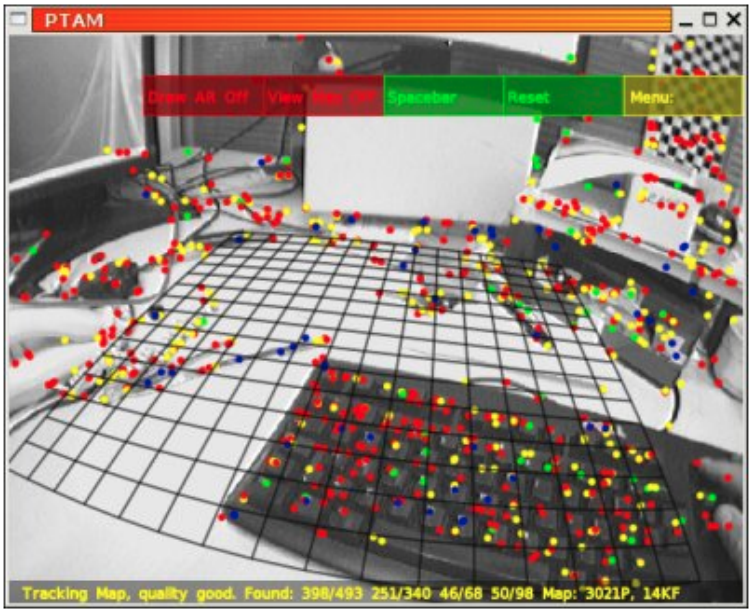


Figure 1.5: Demotration of PTAM[17]

MonoSLAM[9], the first slam system was developed by Davison et al in 2003. It is a filter-base slam. There is only one state vector to represent all positions of 3d points and all poses, and keep updating by Extended Kalman Filter. As we can see, the size of the state vector increases in proportion to the size of the map, thus MonoSLAM is not able to run in real time. After that, Parallel Tracking And Mapping (PTAM)[17] was developed by Georg Klein and David Murray in 2007. It is the first algorithm spliting the tracking and the mapping into two different threads on CPU. To track the camera pose, 3d mappoints

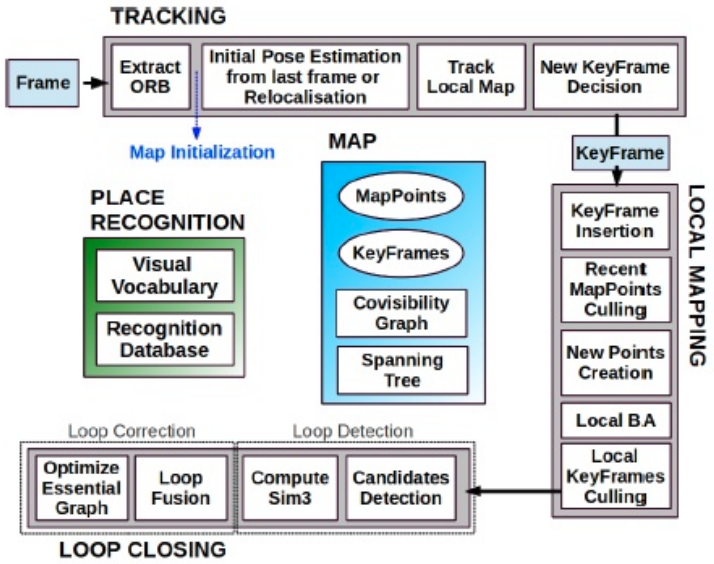


Figure 1.6: Structure of ORB-SLAM[20]

project on a 2d image to find corresponding pairs by matching the texture. To build or update the map, it would triangulate unmatched points on the reference frame. It also optimizes the mappoints by local bundle adjustment and optimizes all poses of keyframe and mappoints by global adjustment. In 2015, ORB-SLAM[20] has been developed by Raul Mur-Artalet al. Instead of two threads for tracking and mapping, it has three threads for tracking, mapping and loop-closing. Without loop-closing, the slam system treats every sence a new sence and thus it is deemed to drift over time. ORB-SLAM would detect loops, close the loop and optimize all the poses by pose-graph optimization (it is very fast comparing to BA, hence it can close the loop in real time). Furthermore, it is the first algorithm to use oriented binary feature, which is much faster and capable of handling a lot of keypoints in real time.

1.5 Outline

Chapter 2 will introduce the basic knowledge that we need for this thesis, including coordinate system, feature detection and matching, Lie group and Lie algebra, probabilistic theory and triangulation. In Chapter 3, we will discuss the contributions and provide qualitative and quantitative testing results. Last but not least, Chapter 4 will summarize this thesis.

Chapter 2

Fundamentals

2.1 Lie group and Lie algebra

A group is a set $\circ : G \times G \to G$ satisfying four properties,

1. **Closure**, $\forall a, b \in G,\ a \circ b \in G$
2. **Associativity**, $\forall a, b, c \in G,\ (a \circ b) \circ c = a \circ (b \circ c)$
3. **Identity element**, $\exists e \in G,\ s.t\ \forall x \in G,\ e \circ x = x \circ e = x$
4. **Inverse element**, $\forall a \in G, \exists b \in G\ s.t\ e \circ x = x \circ e = x$

Lie group is a group with continuous elements. A rigid body motion is composed of translation and rotation. These are continuous motion in space, so that they can be represented by lie group. Lie algebra is the generator of Lie group, which lives in the tangent space of Lie group[11][27]. The adjoint of Lie algebra is used to transform the Lie algebra from one tangent space to another. To define the adjoint of Lie algebra, suppose that G is a Lie Group and $\mathcal{L}(G)$ is the Lie algebra associated with G, for $R \in G$ and $\Phi \in \mathcal{L}(G)$, the adjoint of Lie algebra is showed as follows,

$$\mathrm{ad}_R \Phi = \ln(R \Phi^{\wedge} R^{-1})^{\vee} \tag{2.1}$$

where $^{\wedge}$ changes the Lie algebra to the correspondent element in Lie Group and $\vee$ changes the element from Lie Group to its correspondent element in Lie Algebra.

From the above definition, the following equation hold for any $\Phi \in \mathcal{R}^d$, where d is the degree of freedom of the Lie group.

$$R e^{\Phi} = e^{\mathrm{ad}_R \Phi} R \tag{2.2}$$

To perform efficient optimization with Gaussian-Newton or Levenberg-Marquardt algorithms, we need to define Jacobians that represent a linearization of the cost function.

Since the addition of two elements from Lie group is not in Lie group, the traditional definition of differentiation ($\frac{f(x+\Delta x)-f(x)}{\Delta x}$) is not suitable for Lie group. Here, we will use left-tangent space perturbation to define differentiation. We just need to be careful when we update the minimizer after each iteration of optimization.

2.1.1 Rotation

The special orthogonal group, representing rotation,

$$SO(3) = \{R \in \mathbb{R}^{3x3} | R^T R = RR^T = \mathbf{I}, \det R = 1\} \tag{2.3}$$

SO(3) is a non-commutive group, which means that $R_1, R_2 \in SO(3) \nRightarrow R_1 R_2 = R_2 R_1$. We can use exponential map to change Lie group to the corresponding Lie algebra and vice versa.

$$R = e^{\Phi} \tag{2.4}$$

where $R \in SO(3)$ and $\Phi \in so(3)$.

Φ is an anti-symmetric matrix, we can write it as a 3-vector ϕ. People use $\wedge$ and $\vee$ to interchange these two forms of Lie algebra (only $\Phi = \phi^\wedge$ satisfies the definition of Lie algebra).

$$\Phi = \phi^\wedge = \begin{pmatrix} 0 & -\phi_3 & \phi_2 \\ \phi_3 & 0 & -\phi_1 \\ -\phi_2 & \phi_1 & 0 \end{pmatrix} \Longleftrightarrow \phi = \Phi^\vee = \begin{pmatrix} -\Phi_{23} \\ \Phi_{13} \\ -\Phi_{12} \end{pmatrix} \tag{2.5}$$

By expanding the exponential, we get a close-form transformation from ϕ to R so called Rodrigues formula. Let $a = \frac{\phi}{|\phi|}$, we obtain

$$R = \cos\phi + (1 - \cos\phi) a a^T + \sin\phi \, a^\wedge \tag{2.6}$$

and its inverse

$$\ln(R) = \frac{\theta}{2\sin\theta}(R - R^T) \tag{2.7}$$

$$\theta = \arccos(\frac{\mathrm{tr}(R) - 1}{2}) \tag{2.8}$$

2.1.1.1 Adjoint

Adjoint of Lie algebra will be used when we calculate the derivative of relative rotation with respect to an absolute rotation. For $\Phi \in so(3)$ and $R \in SO(3)$, let $\Phi = wt$, we take derivative of eq.2.2 with respect to t and perform taylor expansion, we obtain

$$\mathrm{ad}_R = R \tag{2.9}$$

2.1.1.2 Jacobians

Let p_0 to be the original point, $p = Rp_0$ is the transformed point ,then

$$\frac{\partial p}{\partial R} \equiv \frac{\partial p}{\partial \phi_R} = \frac{\partial}{\partial w} e^{w^\wedge} R p_0 |_{w=0} = -p^\wedge \tag{2.10}$$

where $\phi_R = \ln(R)$ and $w \in \mathbb{R}^3$.

REMARKS: When there is a derivative w.r.t a Lie algebra, we will change the symbol of Lie algebra to its Lie group for convenience, for exmaple, $\frac{\partial p}{\partial \phi_R} \equiv \frac{\partial p}{\partial R}$.

To calculate the jacobian of relative rotation, let $R_2 = R_1^T R_0$, $R_3 = R_1 R_0^T$ and again using left tangent space perturbation to define differentiation ($e^\epsilon R_2 = R_1 e^{w^\wedge} R_0$) and $R_1 e^{w^\wedge} = e^{\mathrm{ad}_{R_1} w} R_1$

$$\frac{\partial R_2}{\partial R_0} = \frac{\partial \ln(R_1^T e^{w^\wedge} R_0 R_2^{-1})}{\partial w}|_{w=0} = \mathrm{ad}_{R_1^T} = R_1^T \tag{2.11}$$

$$\frac{\partial R_2}{\partial R_1} = \frac{\partial \ln(R_1^T e^{-w^\wedge} R_0 R_2^{-1})}{\partial w}|_{w=0} = -\mathrm{ad}_{R_1^T} = -R_1^T \tag{2.12}$$

$$\frac{\partial R_3}{\partial R_0} = \frac{\partial \ln(R_3 e^{-w^\wedge} R_3^T)}{\partial w}|_{w=0} = -\mathrm{ad}_{R_3} = -R_3 \tag{2.13}$$

$$\frac{\partial R_3}{\partial R_1} = \frac{\partial \ln(e^{w^\wedge} R_1 R_0^T R_3^{-1})}{\partial w}|_{w=0} = I_{3x3} \tag{2.14}$$

2.1.2 Rigid body transformation in 2D

The special Euclidean group represents the pose in 2-dimension. This will be used in optical flow algorithm.

$$\mathrm{SE}(2) = \{S = \begin{bmatrix} R & t \\ 0^T & 1 \end{bmatrix}, \in \mathbb{R}^{3\times 3} | R \in SO(2), t \in \mathbb{R}^2\} \tag{2.15}$$

SE(2) is a non-coummutive group. Each element is composed of 2d rotation and 2d translation. Again, we can use the exponential map to transform a Lie algebra element to its corresponding Lie group element and vice versa. In order to do so, similar to the SO(3) case, we first find the generators of the group by

$$G_i = \frac{\partial}{\partial \alpha_i} S \tag{2.16}$$

where $\alpha_i \in \{u_x, u_y, \theta\}$, $\mathbf{u}$ is the translation vector and the θ is the rotation angle.

$$G_1 = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad (2.17) \qquad G_2 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix} \quad (2.18) \qquad G_3 = \begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad (2.19)$$

So, we define $\delta = \begin{pmatrix} u_x & u_y & \theta \end{pmatrix}^T$, such that

$$\Delta = \delta^\wedge = \alpha_i G_i = \begin{pmatrix} \theta\sigma_x & \mathbf{u} \\ 0 & 0 \end{pmatrix} \quad (2.20)$$

where $\sigma_x = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}$.

By expanding the exponential, we get a close-form transformation from δ to S,

$$S = \begin{pmatrix} e^{\theta\sigma_x} & V\mathbf{u} \\ 0 & 1 \end{pmatrix} \quad (2.21)$$

where $V = \frac{1}{\theta}\begin{pmatrix} \sin\theta & -(1-\cos\theta) \\ 1-\cos\theta & \sin\theta \end{pmatrix}$ and its inverse

$$\ln(S) = \ln\begin{pmatrix} R & t \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} V^{-1}t \\ \theta \end{pmatrix} \quad (2.22)$$

$$\theta = \arctan(\frac{R_{21}}{R_{11}}) \quad (2.23)$$

2.1.2.1 Adjoint

Adjoint of se(2) is defined as follows,

$$Se^{\Delta} = e^{\mathrm{ad}_S \Delta} S \quad (2.24)$$

Let $\Delta = wt$, take derivative with respect to t and perform taylor expansion, we can get,

$$\mathrm{ad}_S = \begin{pmatrix} R & -\sigma_x \mathbf{t} \\ 0 & 1 \end{pmatrix} \quad (2.25)$$

2.1.2.2 Jacobians

Jacobians will be used to minimize the photometric error later in optical flow. Let p_0 to be the origin point and $p = Sp_0$ the transformed point, then

$$\frac{\partial p}{\partial S} = \frac{\partial}{\partial w} e^{w^\wedge} S p_0|_{w=0} = \begin{pmatrix} \mathbb{I}_{2\times 2} & \sigma_x \mathbf{t} \end{pmatrix} \quad (2.26)$$

2.1.3 Ridge body transformation in 3D

The special Euclidean group represents the pose in 3-dimension. This will be used in bundle adjustment.

$$SE(3) = \{T = \begin{bmatrix} R & t \\ 0^T & 1 \end{bmatrix}, \in \mathbb{R}^{4\times4} | R \in SO(3), t \in \mathbb{R}^3\} \tag{2.27}$$

SE(3) is also a non-coummutive group. Each element is composed of 3d rotation and 3d translation. Once again, we can use the exponential map to change Lie group to the corresponding Lie algebra and vice versa. In order to do so, just like the SO(3) and SE(2) case, we first find the generators of the group by

$$G_i = \frac{\partial}{\partial \alpha_i} S \tag{2.28}$$

where $\alpha_i \in \{u_x, u_y, u_z, w_1, w_2, w_3\}$.

$$G_1 = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \tag{2.29}$$

$$G_2 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \tag{2.30}$$

$$G_3 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix} \tag{2.31}$$

$$G_4 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \tag{2.32}$$

$$G_5 = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \tag{2.33}$$

$$G_6 = \begin{pmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \tag{2.34}$$

So, we define $\delta = [u_x, u_y, u_z, w_1, w_2, w_3]^T = [\mathbf{u}, \mathbf{w}]^T$, such that

$$\Delta = \delta^\wedge = \alpha_i G_i = \begin{pmatrix} \mathbf{w}^\wedge & \mathbf{u} \\ 0 & 0 \end{pmatrix} \tag{2.35}$$

By expanding the exponential, we get a close-form transformation from δ to S.

$$T = \begin{pmatrix} R & V\mathbf{u} \\ 0 & 1 \end{pmatrix} \tag{2.36}$$

where $\theta = |w|$, $V = \mathbf{I} + \frac{\sin\theta}{\theta} w^\wedge + \frac{1-\cos\theta}{\theta^2}(w^\wedge)^2$ is the Rodrigues formula and $V = \mathbf{I} + \frac{1-\cos\theta}{\theta^2} w^\wedge + \frac{1-\frac{\sin\theta}{\theta}}{\theta^2}(w^\wedge)^2$ and it inverse just

$$\ln(T) = \ln \begin{pmatrix} R & t \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} V^{-1}t \\ \ln R \end{pmatrix} \tag{2.37}$$

where $\ln R$ can be calculated by eq.2.7

2.1.3.1 Adjoint

Adjoint of se(3) is defined as follows,

$$Te^{\Delta} = e^{\mathrm{ad}_T \Delta} T \tag{2.38}$$

Let $\Delta = wx$, take derivative with respect to x and perform taylor expansion, we can get,

$$\mathrm{ad}_T = \begin{pmatrix} R & \mathbf{t}^{\wedge} R \\ 0 & R \end{pmatrix} \in \mathbb{R}_{6\times 6} \tag{2.39}$$

2.1.3.2 Jacobians

Jacobians will be used to optimize the relative poses (for example $T_{wt}^{-1} T_{wh}$) later in the section of optical flow. Let p_0 to be the origin point, $p_t = T_{th} p_h$ the transformed point,

$$\frac{\partial p}{\partial T} = \frac{\partial}{\partial w} e^{w^{\wedge}} T p_0 |_{w=0} = \begin{pmatrix} \mathbf{I}_{3\times 3} & -\mathbf{p}^{\wedge} \end{pmatrix} \tag{2.40}$$

For calculating the jacobian of relative pose, let $T_{th} = T_{wt}^{-1} T_{wh}$ and $T'_{th} = T_{tw} T_{hw}^{-1}$, note that even $T_{th} = T'_{th}$, their jacobians are not the same due to the fact that we use left tangent space perturbation to define derivative of poses, and again use the left tangent space perturbation to define differentiation, $Te^{w^{\wedge}} = e^{\mathrm{ad}_T w^{\wedge}} T$ and $T_{wt}^{-1} T_{wh} T_{th}^{-1} = \mathbf{I}_{4x4}$

$$\frac{\partial T_{th}}{\partial T_{wh}} = \frac{\partial \ln(T_{wt}^{-1} e^{w^{\wedge}} T_{wh} T_{th}^{-1})}{\partial w} |_{w=0} = \mathrm{ad}_{T_{wt}^{-1}} \tag{2.41}$$

$$\frac{\partial T_{th}}{\partial T_{wt}} = \frac{\partial \ln(T_{wt}^{-1} e^{-w^{\wedge}} T_{wh} T_{th}^{-1})}{\partial w} |_{w=0} = -\mathrm{ad}_{T_{wt}^{-1}} \tag{2.42}$$

$$\frac{\partial T'_{th}}{\partial T_{wh}} = \frac{\partial \ln(T_{tw} T_{hw}^{-1} e^{-w^{\wedge}} T'^{-1}_{th})}{\partial w} |_{w=0} = -\mathrm{ad}_{T_{th}} \tag{2.43}$$

$$\frac{\partial T'_{th}}{\partial T_{wt}} = \frac{\partial \ln(e^{w^{\wedge}} T_{tw} T_{hw}^{-1} T'^{-1}_{th})}{\partial w} |_{w=0} = \mathbf{I}_{6\times 6} \tag{2.44}$$

2.2 Inertial measurement unit (IMU)

Inertial measurement unit, also known as IMU is composed of a gyroscope and accelerometer. Gyroscope is used to measure the angular velocity with respect to its own IMU coordinate. Accelerometer is then used to measure the linear acceleration.

A gyroscope measurement can be split into three parts, the ground true angular velocity (w), bias (b_g) and zero Gaussian noise (η_g).

$$\tilde{w} = w + b_g + \eta_g \tag{2.45}$$

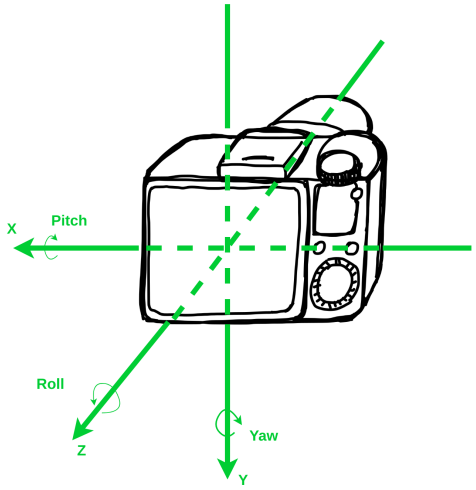


Figure 2.1: IMU coordinate

where $\eta_g = N(0, \sigma^2_{gyro})$.

An accelerometer measurement can be split into four parts, the acceleration of IMU in world coordinate ($^w a$), the gravity acceleration in world coordinate ($^w g$), bias ($^b b_a$) and (3) zero Gaussian noise ($^b \eta_a$).

$$^b\tilde{a} = R_{bw}(^w a - {}^w g) + {}^b b_a + {}^b \eta_a \tag{2.46}$$

where ^{b} and ^{w} denote quantities in IMU coordinate and world coordinate respectively, and $\eta_a = N(0, \sigma^2_{acc})$.

To transform the measurements of IMU into pose and velocity, we use the kinematic equations,

$$R_{wb}(t + \Delta t) = R_{wb}(t)\ e^{(w(t)\Delta t)^\wedge} \tag{2.47}$$

$$^w v(t + \Delta t) = {}^w v(t) + {}^w a(t)\Delta t \tag{2.48}$$

$$^w s(t + \Delta t) = {}^w s(t) + {}^w v(t)\Delta t + \frac{1}{2} {}^w a(t)\Delta t^2 \tag{2.49}$$

To use the measurements for predicting the next state, we insert eq. 2.45 and eq. 2.46 into kinematic equations,

$$R_{wb}(t + \Delta t) = R_{wb}(t)\ exp(((\tilde{w}(t) - b_g(t) - \eta_g(t))\Delta t)^\wedge) \tag{2.50}$$

$$^w v(t + \Delta t) = {}^w v(t) + {}^w g\Delta t + R_{wb}(t)(^b\tilde{a}(t) - {}^b b_a - {}^b \eta_a)\Delta t \tag{2.51}$$

$$^{w}s(t+\Delta t) = {}^{w}s(t) + {}^{w}v(t)\Delta t + \frac{1}{2}{}^{w}g\Delta t^2 + \frac{1}{2}R_{wb}(t)({}^{b}\tilde{a(t)} - {}^{b}b_a - {}^{b}\eta_a)\Delta t^2 \tag{2.52}$$

2.3 Camera models and Calibration

A camera is an electronic device to project a 3D world into a 2D plane, we will introduce several camera models.

2.3.1 Coordinate system

We need different kinds of coordinate in a bid to represent locations of landmarks, poses of cameras and keypoints in images. Camera models are the ways to transform coordinate between these coordinate systems.

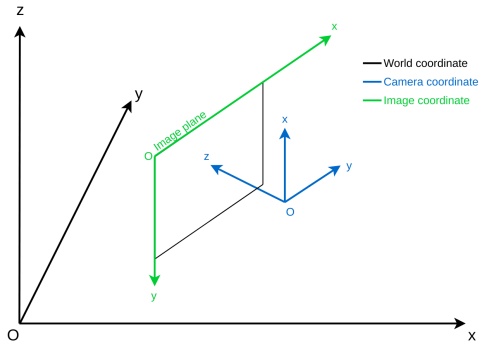


Figure 2.2: World, camera and pixel coordinate

2.3.1.1 World coordinate

The world coordinate can be fixed by given a prior map or using the first frame as reference frame if there is no prior coordinate system.

2.3.1.2 Camera coordinate

The origin of the camera coordinate is located at camera center and one of the axes is aligned to the principal axis. The transformation between world coordinate and camera coordinate is a rigid body transformation.

2.3.1.3 Image coordinate

The image coordinate is obtained by projecting corresponding camera coordinate onto an image plane.

2.3.2 Camera models

We will represent pixel coordinate as $\mathbf{u} = [u, v, 1]^T = [\tilde{u}, 1]^T \in \mathbb{P}^2$ and 3D points in camera coordinate as $\mathbf{X} = [X, Y, Z, 1]^T = [\tilde{X}, 1]^T \in \mathbb{P}^3$. Also, camera models are mapping from camera coordinate to pixel coordinate $\pi : \mathbb{P}^3 \to \mathbb{P}^2$, its inverse $\pi^{-1} : \mathbb{P}^2 \to \mathbb{P}^3$ unproject image coordinate to the bearing vector.

There are several famous camera models, which are based on different theories and have different numbers of degree of freedom, for example, pinhole camera model (4 DoF), field-of-view camera model (5 DoF), unified camera model(5 DoF), extended unified camera model (6 DoF)[16], double sphere camera model (6 DoF)[32] and Kannala-Brandt camera model (6 or 8 DoF)[15]. In this section, we will have a brief introduction to the pinhole camera model and double sphere model, the former is the simplest camera model and the latter will be used in this thesis.

2.3.2.1 Pinhole camera model

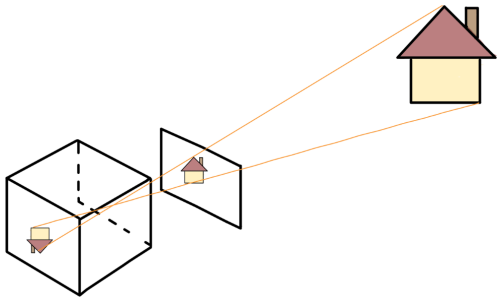


Figure 2.3: Pinhole camera model

Pinhole camera model is a linear transformation between image coordinate and pixel coordinate, which has four degree of freedom $i = [f_x, f_y, c_x, c_y]$ with the projection function below,

$$\mathbf{u} = \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \pi(\tilde{X}, i) \circ \mathbf{X} = \begin{bmatrix} f_x & 0 & c_x & 0 \\ 0 & f_y & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \tag{2.53}$$

2.3.2.2 Double sphere camera model

Double sphere camera model is a non-linear transformation between camera coordinate and pixel coordinate. It has a close-form inverse and six degree of freedom $i = [f_x, f_y, c_x, c_y, \xi, \alpha]$, while the extra two degrees of freedom are used to deal with the image distortion created by the camera, for example, fish eye camera. The projection function is defined as,

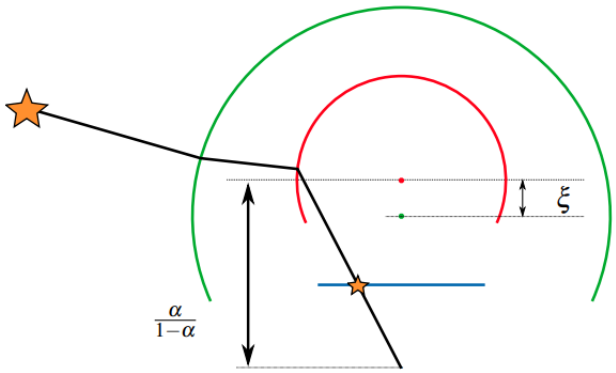


Figure 2.4: Double sphere camera model[33]

$$\mathbf{u} = \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \pi(\tilde{X}, i) \circ \mathbf{X} = \begin{pmatrix} f_x \frac{X}{\alpha d_2 + (1-\alpha)(\xi d_1 + Z)} + c_x \\ f_y \frac{Y}{\alpha d_2 + (1-\alpha)(\xi d_1 + Z)} + c_y \\ 1 \end{pmatrix} \tag{2.54}$$

$$d_1 = \sqrt{X^2 + Y^2 + Z^2} \tag{2.55}$$

$$d_2 = \sqrt{X^2 + Y^2 + (\xi d_1 + Z)^2} \tag{2.56}$$

2.4 Features detection

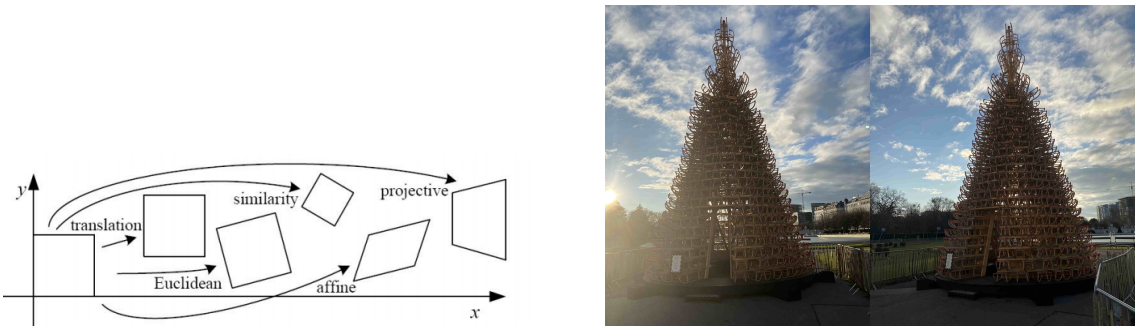
Features are special subset of the image which contains more information, so they are easily described and robust to view of angle, for example, corners, lines and blobs etc. There are two types of feature detection, indirect and direct method.

2.4.1 Indirect method

An image is composed of a matrix of brightness or colours (red, blue and green). One of the common ways to compare two images is to find several representative points in both images, and turn the brightness around those representative points into another mathematical representation which can be compared to each other easily. A good mathematical representation for a feature should fulfill requirements below.

1. Distinctiveness: Different features should have distinct feature descriptor.

2. Invariance: The same feature can be found across the images and it should be regardless to the geometric (translation, rotation, affine, projective transformation) and photometric (brightness) changes. See Fig 2.5



(a) Geometric change[30] (b) Photometric change

Figure 2.5: Geometric and photometric changes

3. Robustness: The feature descriptors are robust to noise, blur, quantization and etc.

4. Efficiency: The number of features should be much smaller than the number of pixel in the image.

5. Locality: The features are created by the pixels nearby the feature corners, which make the feature robust to clutter and occlusions.

There are many features extraction algorithms developed, such as Scale-invariant feature transform (SIFT)[19], Speeded up robust features (SURF)[4], Oriented Features from Accelerated Segment Test (FAST)[23] and Binary Robust Independent Elementary Features (BRIEF)[8]. In the following, we will briefly introduce Oriented FAST corner detection and BRIEF descriptor which are used in our thesis.

2.4.1.1 Oriented FAST corner detection

The first step to construct features is finding "interesting points" in the image. The FAST corner detector[23] was originally developed by Edward Rosten and Tom Drummond in 2006. It has been widely used in many computer vision tasks because of its computational efficiency. It is faster than many other well-known feature extraction methods, such as (1) Difference of Gaussians (DoG) which requires to apply gaussian filter for several times to the image and (2) Harris corner detectors which needs to calculate the derivative of x- and y-direction for each pixel in order to calculate Harris score.

In the contrary, computing FAST corner score does not require to calculate any derivative but just compare the brightness with its neighbour. The steps are as follows:

1. Assume its brightness is $I_p \in [0, 255]$ for a selected pixel p.

2. Set a threshold T to distinguish the testing pixel being inside or outside the corner.

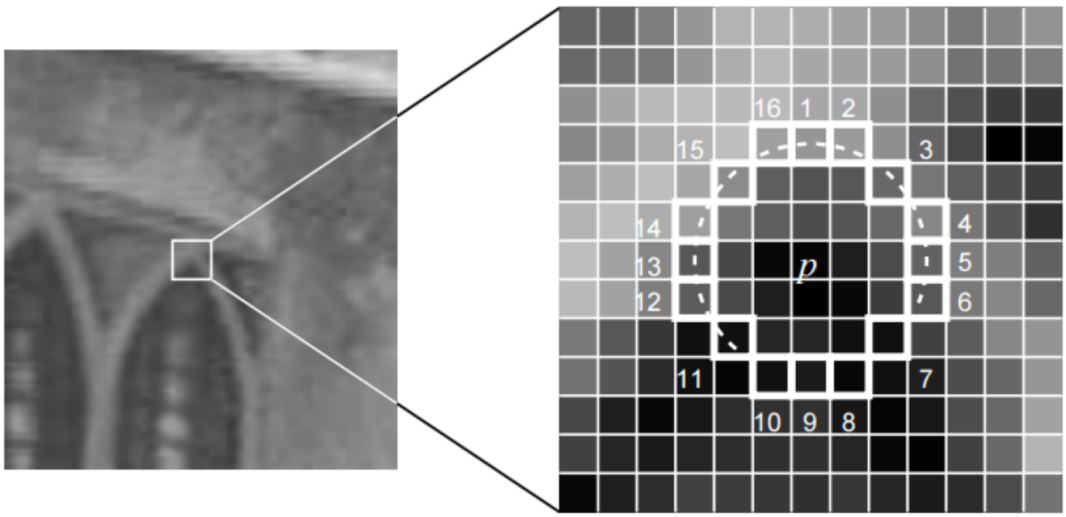


Figure 2.6: Fast corner detection[23]

3. Draw a circle centered at pixel p with radius 3 pixels, there are 16 pixels on the circumstances.

4. If there are at least N contiguous pixels that are either all T brighter or T darker than I_p, than we consider pixel p a feature corner, we usually take N=12.

5. Apply the above steps to every pixel in the image.

To make the algorithm faster, we can apply an high-speed test for rejecting non-corner points. For N=12, we test only the pixel 1, 4, 9, 13. Pixel p can only be a corner if and only if at least 3 out of 4 sampling pixels that are all brighter or darker than I_p. Moreover, since the neighbour of a pixel should also have similar FAST corner score to some extent, it would lead to too many feature corners in a small region of the image. In order to tackle this problem, we apply non-maximal suppression to filter some feature corners based on a few criteria.

We calculate the orientation of a keypoint by calculation of an Intensity Centroid. There is an analogy between Intensity Centroid and calculating center of mass in physics. The step of Intensity Centroid for sub-image S, is as follows,

1. The torque in x-direction (τ_x) is $\sum_{x,y \in S} x \cdot I(x,y)$.

2. The torque in y-direction (τ_y) is $\sum_{x,y \in S} y \cdot I(x,y)$.

3. The total intensity (I) is $\sum_{x,y \in S} I(x,y)$.

4. The center of intensity (C_x,C_y) is $(\frac{\tau_x}{I}, \frac{\tau_y}{I})$.

5. The orientation of subset S is defined as $\arctan(\frac{C_x}{C_y}) = \arctan(\frac{\tau_y}{\tau_x})$.

Note that the steps above works are suitable for any 2d coordinate system.

2.4.1.2 BRIEF descriptor

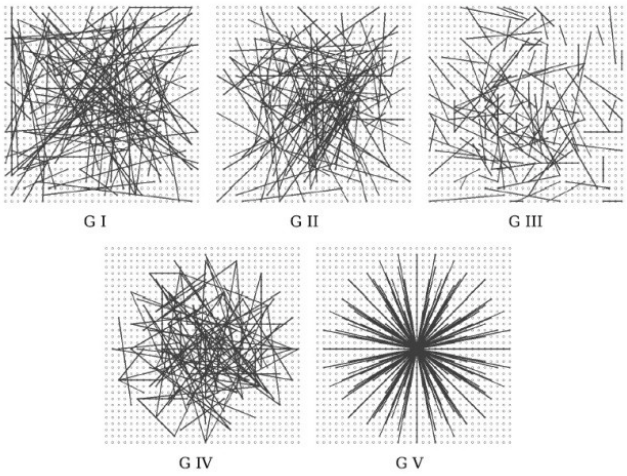


Figure 2.7: Five different approaches to choosing the pairs of points[8]

After extracting N "interesting" points, we need to describe those points. BRIEF is a binary feature descriptor[8], which is usually a 128 bit to 512 bit string. Since it is stored as a vector with elements only 0 or 1, it is easy to compute and do matching with other descriptors, which makes it extremely fast while having high repeatabilty at the same time.

To construct a binary feature vector for an image patch, we first select n pairs of points based on different kinds of sampling method and perform binary test (t), so the BRIEF is vector of the responses of the binary test. Assume p(i,n) be the n-th element in the i-th pair, the i-th element of binary feature vector (v[i]) is calculated by the following equation,

$$v[i] = \begin{cases} 1 & p(i,0) > p(i,1) \\ 0 & else \end{cases} \tag{2.57}$$

2.4.2 Direct method

Instead of constructing feature descriptor for each corner, the main idea of direct method is to use the intensity directly. Direct method has become a hot research topic due to several reasons.

1. Compared to indirect method which construct thousands features, direct method take all pixel in the image into account, and thus make sure that no information is missing.

2. Indirect method detects corners and creates descriptors. For a scene with no obvious texture, direct method can perform better.

2.5 Features matching

After creating features for each image in the sequence, we will find those features that also exist in other images. The idea of matching features is basically testing the similarity of two feature descriptors through their inner product, If it is smaller than the threshold, then it is considered as a match. However, it is expensive to perform bruce force matching to all features in those two images, hence we filter some candidates that do not match the geometric constraint. Here are two examples.

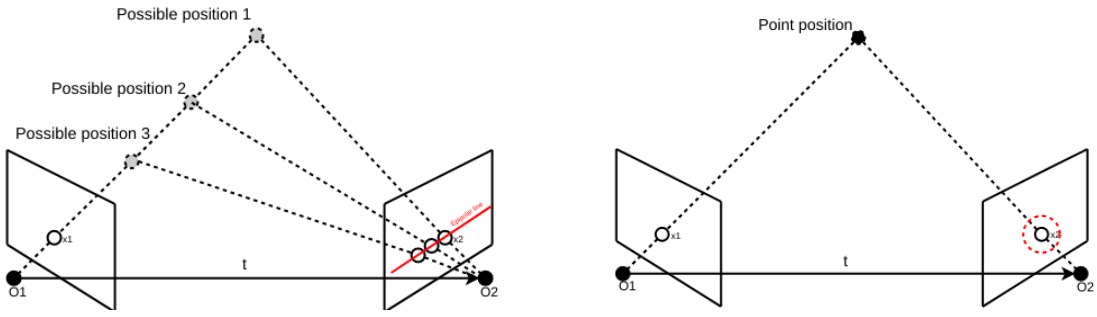
2.5.1 Epipolar constraint

Given the keypoint position in one frame and the relative pose between two frames, we can draw an epipolar line on the second frame and the corresponding feature must lie on the line, see Fig 2.8a

Assume x_1 (in homogeneous coordinates) be the keypoint position in image coordinate of the first frame, X be the corresponding 3d position, R be the rotation from 1st camera coordinate to 2nd camera coordinate, -T is the displacement from 2nd camera center to 1st camera center. Therefore, we have,

$$\lambda_1 x_1 = X \tag{2.58}$$

$$\lambda_2 x_2 = RX + T = \lambda_1 R x_1 + T \tag{2.59}$$



(a) Epipolar constraint (b) Reprojection constraint

Figure 2.8: Visualization of geometric constraint

To remove addition by multiplying with $\hat{T}$ where $\hat{T} = \begin{pmatrix} 0 & -T_z & T_y \\ T_z & 0 & -T_x \\ -T_y & T_x & 0 \end{pmatrix}$,

$$\lambda_2 \hat{T} x_2 = \lambda_1 \hat{T} R x_1 \tag{2.60}$$

Then we project the above equation to x_2, since λ_1 and λ_2 are not equal to 0,

$$x_2^T \hat{T} R x_1 = 0 \tag{2.61}$$

Owing to noise, pixel quantization and etc, it is very likely that there are no x_1 and x_2 satisfying eq 2.61. We would set a threshold ϵ, which means that the set of x_2 does not need to lie on the epipolar line but just nearby it. Therefore, the epipolar constraint should be rewritten as,

$$|x_2^T \hat{T} R x_1| < \epsilon \tag{2.62}$$

2.5.2 Reprojection constraint

Given the 3d position of the feauture and the absolute pose of the second frame, we can shrink the search area into a small circle, see Fig 2.8

2.6 Probability theory in visual navigation

There are many good methods to calculate close-form solution of poses given the 3D position of feature points and the corresponding points in the image, for example, PnP-ransac, 5-points and 8-points algorithm up to scale. However, every measurement contains noise, and there are problems like that

1. the algorithm is noise sensitive
2. no information about how accurate the solution is
3. how to update the solution while there are more observations made

Instead of calculating a close-form solution, which is the best solution the algorithm can give us, we calculate the probabilistic representation of the estimated solution, which is a probabilistic distribution over the space of all possible solution. With probabilistic representation, the uncertainty and degeneracy of the solution can be written in a mathematical way.

2.6.1 Gaussian distribution

The Gaussian distribution is widely used because

1. it is governed by only two parameters (mean and variance) which can be easily obtained from data,

2. the mathematics is simple and well defined

The 1D Gaussian distribution and multivariate Gaussian are

$$\mathcal{G}(x-\mu,\sigma_x^2) = \frac{1}{\sqrt{2\pi\sigma_x^2}}\exp(-\frac{(x-\mu)^2}{2\sigma_x^2}), \tag{2.63}$$

$$\mathcal{G}(x-\mu,\Sigma) = \frac{1}{\sqrt{|2\pi\Sigma|}}\exp(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)). \tag{2.64}$$

2.6.2 Maximum likelihood (MLE) and Maximum a posteriori (MAP)

If we want to find the probabilistic distribution of a "state variable" x, we can do some observation z. We can consider that z is a transformation of x that we can measure,

$$z = f(x) + \epsilon, \tag{2.65}$$

where f is the response function and $\epsilon \sim \mathcal{G}(x-\mu,\sigma_x^2)$ is stochastic a variable used to describe noise.

Likelihood is the probability distribution of measurement (z) given the true value f(x),

$$\mathcal{P}(z|x) = \int d\epsilon\ \mathcal{P}(z,\epsilon|x) = \int d\epsilon\ \mathcal{P}(z|\epsilon,x)\mathcal{P}(\epsilon|x) = \mathcal{G}(z-f(x),\sigma_x^2) \tag{2.66}$$

The posterior is the probability distribution of true value f(x) given the measurement (z). By using Bayes rule, we obtain the posterior by combining likelihood and prior of x as follows:

$$\mathcal{P}(x|z) = \frac{\mathcal{P}(z|x)\mathcal{P}(x)}{\int dy\ \mathcal{P}(z|y)\mathcal{P}(y)}. \tag{2.67}$$

There are many ways to define the "best guess" of a state (x) given measurement (z). One of the popular ways is to find x^* such that it maximizes the a posterior, this is called Maximum a posteriori (MAP),

$$x^*_{MAP} = \underset{x}{\operatorname{argmax}}\ P(x|z) = \underset{x}{\operatorname{argmax}}\ P(z|x)P(x) \tag{2.68}$$

Note that if we have a flat prior on x, the solution of MAP is equal to MLE.

2.6.3 The Hamitonian (Energy) to minimize

Let state $\mathbf{x} = \{x_1, x_2, ..., x_N, y_1, y_2, ..., y_M\}$, where the state contains N camera poses and M mappoints, and $\mathbf{u} = \{u_1, u_2, ..., u_{t=N}\}$ is the control data which carry information about the changes of state, for example u_t contains the change of state from time t-1 to time t, and $\mathbf{z} = \{z_{i,j} | i \in \{1, 2, ..., N\}, j \in \{1, 2, ..., M\}\}$ contains all the measurements, $z_{i,j}$ can understand as the measurement of mappoint j with camera pose i. With all these variables, we can represent the probability distribution of a state as

$$x_t = f(x_{t-1}, u_t) + \epsilon_f \tag{2.69}$$

$$z_{t,j} = h(x_t, y_j) + \epsilon_h \tag{2.70}$$

$$P(\boldsymbol{x}|\boldsymbol{z}, \boldsymbol{u}) \tag{2.71}$$

where $\epsilon_f \sim \mathcal{G}(0, F)$ and $\epsilon_h \sim \mathcal{G}(0, H)$, F and H are the covariances of the Gaussian distributions.

The information E is defined as the negative ln to the posterior. The result is as follows,

$$r_i = x_i - f(x_{i-1}, u_i) \tag{2.72}$$

$$r_{i,j} = z_{i,j} - h(x_i, y_j) \tag{2.73}$$

$$E = \frac{1}{2} \sum_i r_i^T F_i^{-1} r_i + \frac{1}{2} \sum_{i,j} r_{i,j}^T H_{i,j}^{-1} r_{i,j} \tag{2.74}$$

The first term is error term from measurement of speed and the second term is the reprojection error, flat prior has been used here.

2.6.4 Optimization scheme

In order to find the MAP estimator x^* by minimizing eq.2.74, we can only solve it with numerical method. We update the state (δx) at each iteration until the Hamitonian converges to a local minimum (Since the residual is usually non-linear, and it is not guaranteed that the local minimum is the global minimum).

2.6.4.1 Gradient descent

Gradient descent is a first order iteration method. By taylor-expanding the response function $\| \mathbf{f}(\mathbf{x} + \delta\mathbf{x}) \|^2$ up to first order,

$$\| \mathbf{f}(\mathbf{x} + \delta\mathbf{x}) \|^2 \approx \| \mathbf{f}(\mathbf{x}) \|^2 + \mathbf{J}(\mathbf{x})^T \delta\mathbf{x} \tag{2.75}$$

We can update the state along with the direction of negative gradient $-\mathbf{J}(\mathbf{x})$ with step size λ according to

$$\mathbf{x} \to \mathbf{x} - \mathbf{J}(\mathbf{x})\lambda. \tag{2.76}$$

2.6.4.2 Newton method

When using Gradient descent, we need to determine the step size λ at each iteration. In order to set the λ automatically, we taylor-expand the response function f(x) around x up to second order,

$$\| \mathbf{f}(\mathbf{x}+\delta\mathbf{x}) \|^2 \approx \| \mathbf{f}(\mathbf{x}) \|^2 + \mathbf{J}(\mathbf{x})^T \delta\mathbf{x} + \frac{1}{2}\, \delta\mathbf{x}^T \mathbf{H}\, \delta\mathbf{x}. \tag{2.77}$$

We differentiate the above equation with respect to $\delta\mathbf{x}$, let LHS to be zero and rearrange the $\delta\mathbf{x}$ as subject, to obtain

$$\mathbf{x} \rightarrow \mathbf{x} - \mathbf{H}^{-1}\mathbf{J}(\mathbf{x}) \tag{2.78}$$

2.6.4.3 Gaussian Newton method

Calculating second order derivative is expensive. Instead of expanding $\| \mathbf{f}(\mathbf{x}+\delta\mathbf{x}) \|^2$, we expand $\mathbf{f}(\mathbf{x}+\delta\mathbf{x})$

$$\| \mathbf{f}(\mathbf{x}+\delta\mathbf{x}) \|^2 \approx \| \mathbf{f}(\mathbf{x}) + \mathbf{J}(\mathbf{x})\delta\mathbf{x} \|^2 = \| \mathbf{f}(\mathbf{x}) \|^2 + 2\mathbf{f}(\mathbf{x})^T \mathbf{J}(\mathbf{x})\delta\mathbf{x} + \delta\mathbf{x}^T \mathbf{J}(\mathbf{x})^T \mathbf{J}(\mathbf{x})\delta\mathbf{x} \tag{2.79}$$

We differentiate above equation with respect to $\delta\mathbf{x}$, let LHS to be zero and rearrange the $\delta\mathbf{x}$ as subject, we obtain

$$\mathbf{x} \rightarrow \mathbf{x} - (\mathbf{J}(\mathbf{x})^T \mathbf{J}(\mathbf{x}))^{-1} \mathbf{J}(\mathbf{x})^T \mathbf{f}(\mathbf{x}) \tag{2.80}$$

2.6.4.4 Levenberg-Marquardt method

Levenberg-Marquardt method is a hybrid method of Gradient descent and Gaussian Newton method. The idea of this method is that if Gaussian Newton method cannot optimize the energy in the right way, it would slightly turn the model into Gradient descent by increasing λ

$$\mathbf{x} \rightarrow \mathbf{x} - (\mathbf{J}(\mathbf{x})^T \mathbf{J}(\mathbf{x}) + \lambda \mathrm{diag}(\mathbf{J}(\mathbf{x})^T \mathbf{J}(\mathbf{x})))^{-1} \mathbf{J}(\mathbf{x})^T \mathbf{f}(\mathbf{x}) \tag{2.81}$$

2.7 Keypoints triangulation

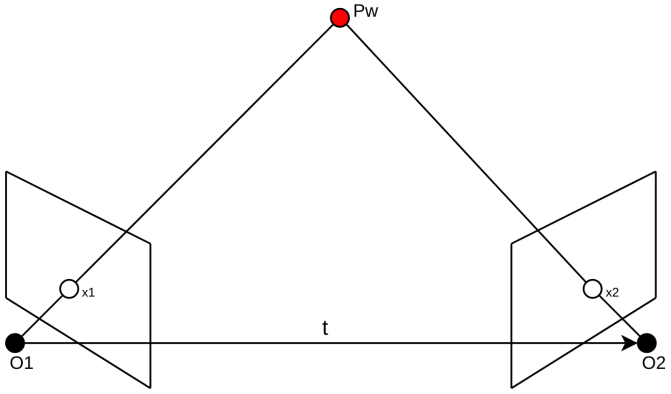


Figure 2.9: Obtain depth through triangulation

Keypoints on images are represented by 2d-homogeneous coordinate. To obtain the depth information of keypoints, we find correspondences between images by indirect feature matching or optical flow and etc.

Given two corresponding points $\mathbf{x}_1$, $\mathbf{x}_2$ on two different images and together with the relative pose, we can triangulate keypoints as follows. Let $\mathbf{p}_{c1} = s_1\mathbf{x}_1$ and $\mathbf{p}_{c2} = s_2\mathbf{x}_2$ to be the 3d position of the same mappoint with different camera coordinates, where s_1 and s_2 are the scale for homogeneous coordinate. So that $\mathbf{x}_1$ and $\mathbf{x}_2$ are related by the following equation,

$$s_1\mathbf{x}_1 = p_{c1} = R(s_2\mathbf{x}_2) + \mathbf{t} \tag{2.82}$$

Multiply both sides with $\mathbf{x}_1^\wedge$ ($a^\wedge b = a \times b$) to find s_2,

$$s_2 = -\frac{|\mathbf{x}_1^\wedge \mathbf{t}|}{|\mathbf{x}_1^\wedge R\mathbf{x}_2|} \tag{2.83}$$

Note that due to the noise, the preimages of the correspondence may not intersect, and this method no longer works. We have to solve it by least square method (used in our work).

Chapter 3

Approach

3.1 3D to 2D Sparse Optical flow

The estimation of an optical flow is based on the pattern of brightness. The advantage of using optical flow is that we don't need to calculate descriptor. However, since the concept of an optical flow is based on the assumption of constant brightness, the error function of the difference of brightness for a sub-image is highly non-convex and sensitive to reflection. Therefore, optical flow can only works well under the condition of small baseline and good initial guess of the optimizer are provided. To improve the accuracy of features tracking by optical flow, we use the reprojection of the 3d mappoints on the next frame as initial guess, and use Gauss-Newton method to optimize the keypoint position.

3.1.1 Purposed structure

3.1.1.1 Building new mappoints

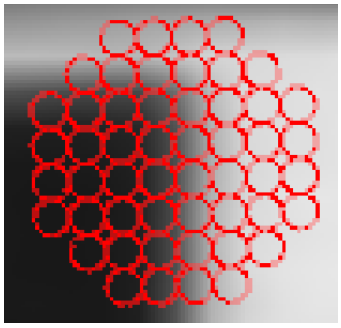


Figure 3.1: Patch of a keypoint[33].

To create new mappoints (in our case, we create new mappoints when inserting a new keyframe), we use FAST corner detector to detect a sparse set of keypoints, then we find the corresponding points on the right frame by using optical flow. Specific patch pattern

has been used to describe keypoints, such as $\Omega = \{x_1, ..., x_N\}$, where x_i is the offset from the keypoint corner and the patch (Ω) is composed of N points, see fig 3.1.

3.1.1.2 Mappoints tracking

We find the keypoint correspondence on another image by using nonlinear optimization. We set up an error function and find a 2d pose $T_i \in SE(2)$ for keypoint i such that it minimizes the following error function[33],

$$E(\xi) = \sum_{p_i \in \Omega} \parallel \text{res}_{i,\Omega}(\xi) \parallel^2 = \sum_{p_i \in \Omega} \parallel \frac{I_{t+1}(e^{\xi^\wedge} p_i)}{\overline{I_{t+1,\Omega}}(\xi)} - \frac{I_t(p_i)}{\overline{I_{t,\Omega}}(\mathbf{0})} \parallel^2 \tag{3.1}$$

where $\overline{I_{t,\Omega}}(\mathbf{0}) \equiv \frac{1}{|\Omega|} \sum_{p_i \in \Omega} I_t(p_i)$ and $\overline{I_{t+1,\Omega}}(\xi) \equiv \frac{1}{|\Omega|} \sum_{p_i \in \Omega} I_{t+1}(e^{\xi^\wedge} p_i)$

To calculate $\delta\xi$ for each iteration, we will use Gauss-Newton method to do so. First, we calculate the jacobian

$$J_i(\xi) = \frac{\partial \text{res}_{i,\Omega}(\xi)}{\partial \xi} = \frac{\partial}{\partial \xi}(\frac{I_{t+1}(exp^{\xi^\wedge} p_i)}{\overline{I_{t+1,\Omega}}(\xi)}) = \frac{\partial}{\partial p'}(\frac{I_{t+1}(p')}{\overline{I_{t+1,i}}(\xi)}) \frac{\partial p'}{\partial \xi}|_{p'=exp^{\xi^\wedge} p_i} \tag{3.2}$$

By using chain rule, we have

$$J_i(\xi) = \frac{\partial}{\partial p'}(\frac{I_{t+1}(p')}{\overline{I_{t+1,i}}(\xi)}) \frac{\partial p'}{\partial \xi}|_{p'=e^{\xi^\wedge}(p_i+x)} \tag{3.3}$$

$\frac{\partial p'}{\partial \xi}|_{p'=e^{\xi^\wedge} p_i}$ can be calculated by eq.2.26 and the updated ξ_{new} is as follows,

$$\xi_{\text{new}} = \xi_{\text{old}} - (\sum_i J_i(\xi_{\text{old}})(J_i(\xi_{\text{old}}))^T)^{-1}(\sum_i J_i(\xi_{\text{old}})^T \text{res}_{i,\Omega}(\xi_{\text{old}})) \tag{3.4}$$

3.1.1.3 Increase efficiency and robustness

To speed up the algorithm, we calculate the jacobian around the point p on I_t so that we don't need to update the jacobian after each iteration. To make this trick work, we need to make sure that the baseline of two images is small enough. Initial position is required for iterative optimization. In Basalt, they use the position of the last frame as the initial position, we have changed to use the reprojection of the keypoint and no correspondence has been found if the optimized position is T more pixels away from the reprojection (T=5 in our case). To filter more outliers, we remove mappoints which do not have enough observations after N frames pass.

3.2 Basalt-SLAM

For Basalt-VIO, a local map of the frames in sliding window is built for local bundle adjustment to increase the stability and accurary. However, all the VIO algorithms are

deemed to drift due to lack of the ability to recognize the place where it has been. In this task, we will build a global map based on ORB-features and rectify the VIO path while loop has been detected.

3.2.1 Purposed structure

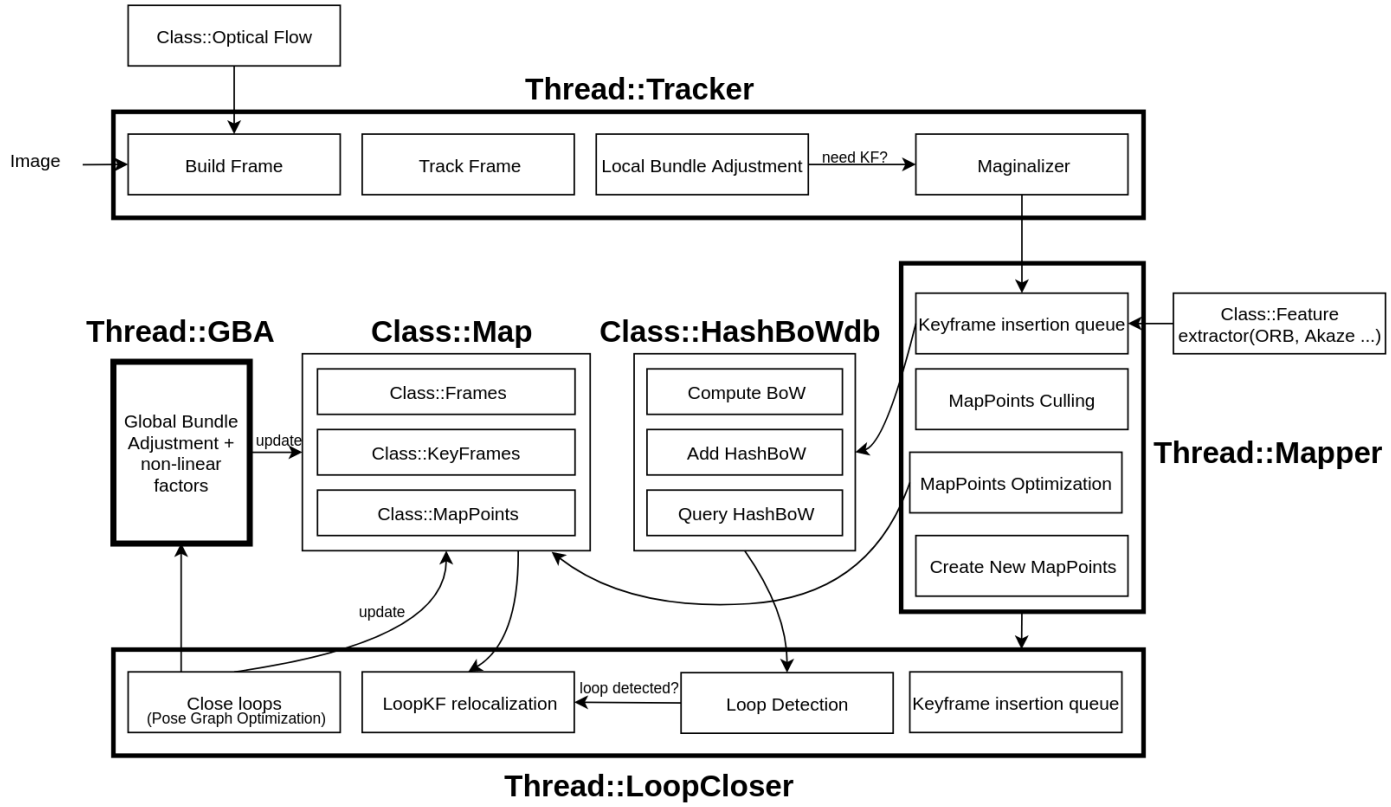


Figure 3.2: Purposed structure of Basalt-SLAM

Basalt-VIO only keeps keyframes in the sliding window. It would marginalize a keyframe out[33] when a new keyframe is created. Some information of the marginalized keyframe are saved, including

1. the pose of the marginalized keyframe,
2. the relative poses to all the frames in the sliding window,
3. the information matrices of the relative poses,
4. the information matrices of the orientation.

The structure is similar to ORB-slam[20], showed in Fig 3.2. The slam system incorperates three threads namely tracking, mapping, and loop-closing, which launch in parallel. We have built several classes to make the structure of the system clearer.

3.2.1.1 Data structure

Camera contains the calibrated interanl parameters. It also contains functions such as projection and unprojection which are used to perform transformation between camera coordinate and pixel coordinate.

Frame is a basic data unit. It contains frame id, camera id (Left or right), timestamp, frame state (frame pose + velocity + gyroscope bias + accelerator bias), image, features, observed map points. Furthermore, it also defines many functions, such as "get features in selected area", "mappoint is visible" and etc.

KeyFrame is similar to Frame except having mMapperID. This is used to represent which sequence the keyframe belongs to.

MapPoint is also a basic data unit. It contains mappoint id, location, frame observation, keyframe observation, mean viewing direction, descriptor of last observation (for tracking) and the best (mean) descriptor (for local mappoint projection). Apart from that, it defines many functions, such as "add observation", "erase observation", "set bad flag" and etc.

Map manages all the mappoints, adding or erasing new frames, keyframes and mappoints.

3.2.1.2 Linearization and optimization

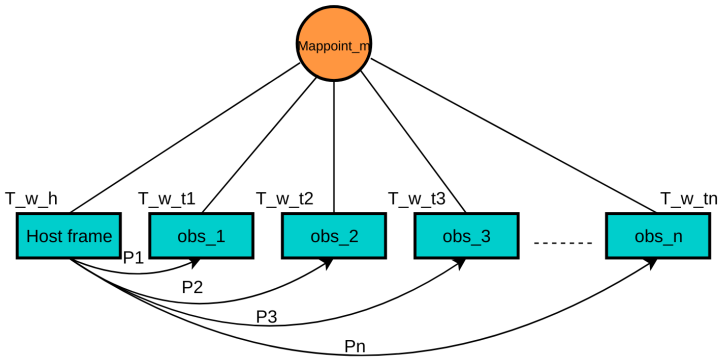


Figure 3.3: Factor graph of a mappoint and its observer.

We can optimize the error function after finding the linearization **H** and **b**. We have stored the observation of mappoints in the structure showed in Fig3.3, where $P_i = T_{ih}$. To calculate H_{abs}, it is easier to calculate the **H** and **b** w.r.t the relative pose for each mappoint m, then transform them into absolute pose.

$$H^m_{\text{rel}} = \begin{pmatrix} H^m_{P_1P_1} & & & & H^m_{P_1m} \\ & H^m_{P_2P_2} & & & H^m_{P_2m} \\ & & \ddots & & \vdots \\ & & & H^m_{P_nP_n} & H^m_{P_nm} \\ H^m_{mP_1} & H^m_{mP_2} & \cdots & H^m_{mP_nm} & H^m_{mm} \end{pmatrix} \tag{3.5}$$

where each matrix element can be obtained by using different M-estimator and different optimization scheme, see Sec2.6.4.

Since the number of mappoint is much larger than the number of pose, the H_{abs} will be very large. However, it is easy to show that it is a sparse matrix. We can reduce the matrix size by using marginalization. The marginalized $\widetilde{H^m_{\text{rel}}}$ and $\widetilde{b^m_{\text{rel}}}$ for mappoint m can be calculated by Gaussian elimination and is showed as follows,

$$\widetilde{H^m_{\text{rel}P_iP_j}} = H^m_{\text{rel}P_iP_j} - H^m_{P_im}\, {H^m_{mm}}^{-1}\, H^m_{mP_j} \tag{3.6}$$

$$\widetilde{b^m_{\text{rel}P_i}} = b^m_{\text{rel}P_i} - H^m_{P_im}\, {H^m_{mm}}^{-1}\, b^m_m \tag{3.7}$$

We can calculate $\widetilde{H_{\text{abs}}}$ and $\widetilde{b_{\text{abs}}}$ with chain rule,

$$\begin{aligned} [\widetilde{H_{\text{abs}}}]_{ij} &= \sum_m \frac{\partial \text{res}(m)}{\partial T_i}^T \frac{\partial \text{res}(m)}{\partial T_j} \\ &= \sum_m (\sum_{l \in \text{rel}(m)} \frac{\partial \text{res}(m)}{\partial P_l} \frac{\partial P_l}{\partial T_i})^T (\sum_{k \in \text{rel}(m)} \frac{\partial \text{res}(m)}{\partial P_k} \frac{\partial P_k}{\partial T_j}) \\ &= \sum_m \sum_{l \in \text{rel}(m), k \in \text{rel}(m)} \frac{\partial P_l}{\partial T_i}^T \frac{\partial \text{res}(m)}{\partial P_l}^T \frac{\partial \text{res}(m)}{\partial P_k} \frac{\partial P_k}{\partial T_j} \\ &= \sum_m \sum_{l \in \text{rel}(m), k \in \text{rel}(m)} \frac{\partial P_l}{\partial T_i}^T \widetilde{H^m_{\text{rel}P_lP_k}} \frac{\partial P_k}{\partial T_j} \end{aligned} \tag{3.8}$$

$$
\begin{aligned}
\widetilde{[b_{\text{abs}}]}_i &= \sum_m \frac{\partial \text{res}(m)}{\partial T_i}^T \text{res}(m) \\
&= \sum_m (\sum_{l \in \text{rel}(m)} \frac{\partial \text{res}(m)}{\partial P_l} \frac{\partial P_l}{\partial T_i})^T \text{res}(m) \\
&= \sum_m \sum_{l \in \text{rel}(m)} \frac{\partial P_l}{\partial T_i}^T \frac{\partial \text{res}(m)}{\partial P_l}^T \text{res}(m) \\
&= \sum_m \sum_{l \in \text{rel}(m)} \frac{\partial P_l}{\partial T_i}^T \widetilde{b^m_{\text{rel}P_l}}
\end{aligned}
\tag{3.9}
$$

where $\text{res}(m) = \sum_{t \in all\_frames} r_{mt}$, r_{mt} is defined as eq3.13, the $\frac{\partial P_l}{\partial T_i}$ can be calculated by using eq2.41, and rel(m) is the set of relative pose of mappoint m from host frame to target frame, see Fig3.3.

The incremental update of poses ($\delta\xi$) is calculated as follows,

$$
\delta\xi = \widetilde{H_{\text{abs}}}^{-1} \widetilde{b_{\text{abs}}} \tag{3.10}
$$

and the incremental update of position of mappoint m (δp_m) is calculated as follows,

$$
\begin{aligned}
\delta p_m &= H^{m-1}_{mm}(b^m_m - \sum_i \frac{\partial \text{res}(m)}{\partial p_m}^T \frac{\partial \text{res}(m)}{\partial \xi_i} \delta\xi_i) \\
&= H^{m-1}_{mm}(b^m_m - \sum_i \frac{\partial \text{res}(m)}{\partial p_m}^T \sum_{j \in \text{rel}(m)} \frac{\partial \text{res}(m)}{\partial P_j} \frac{\partial P_j}{\partial \xi_i} \delta\xi_i) \\
&= H^{m-1}_{mm}(b^m_m - \sum_i \sum_{j \in \text{rel}(m)} H^m_{mP_j} \frac{\partial P_j}{\partial \xi_i} \delta\xi_i)
\end{aligned}
\tag{3.11}
$$

3.2.1.3 Tracking thread

Tracking thread is used to localize the latest frame and insert marginalized keyframe into mapper. First, it performs the optical flow from the previous image. By using IMU measurement as the initial guess of the pose, we perform local bundle adjustment within the sliding window. To decide if inserting a new keyframe, we check whether more than 30 % of all the mappoints can not be observed by the current frame and more than 6 frames have been passed after the latest keyframe. Once we decide to take a new keyframe, we triangulate all the features which have no corresponding mappoint. Before triangulating a new mappoint, we check whether the feature pairs satisfying the epipolar constraint to filter some wrong matching pairs. To keep the system capable of running in real time, we maintain the size of the sliding window as a constant. To do so, we marginalize a keyframe in the window out based on the following criteria similar to DSO[12],

1. We always keep the last keyframe.
2. Keyframes with less than 5% of the features have the corresponding mappoint.
3. If still no keyframe is considered as bad keyframe, we marginalize the one which maximizes the "distance score" s(I_i), defined as follows,

$$s(I_i) = \sqrt{d(i,1)} \sum_{j \in [2,n] lbrace i\}} (d(i,j) + \epsilon)^{-1}, i \neq 1 \tag{3.12}$$

where d(i,j) is the Eulidean distance between keyframes I_i and I_j, and ϵ is a small constant which is used to prevent the score being too larger, when some keyframes in sliding window are very close together.

The score function is designated to keep more keyframes being close to the current keyframe and at the same time prevent keyframes are very close together. While there are keyframes being marginalized, it will send to the mapper for building a global map by ORB features.

3.2.1.4 Mapping thread

Mapping thread processes new keyframes and optimizes the mappoints which can be observed in the current keyframe (we do not update the poses in the map). When mapper receives a new keyframe, it would first build orb keypoints and match them with the previous keyframe. We then project map points created by covisible keyframes of the last keyframe (covisible keyframes are the keyframes sharing at least 15 features of the target frame) to build connection to the keyframes nearby.

3.2.1.4.1 Feature extraction

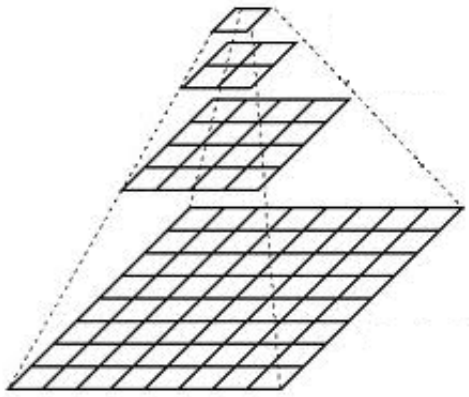
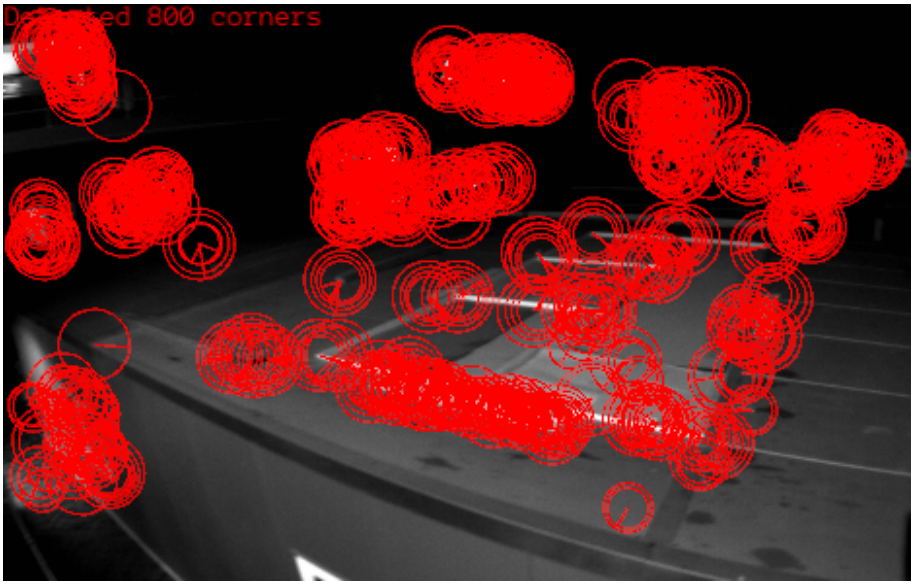
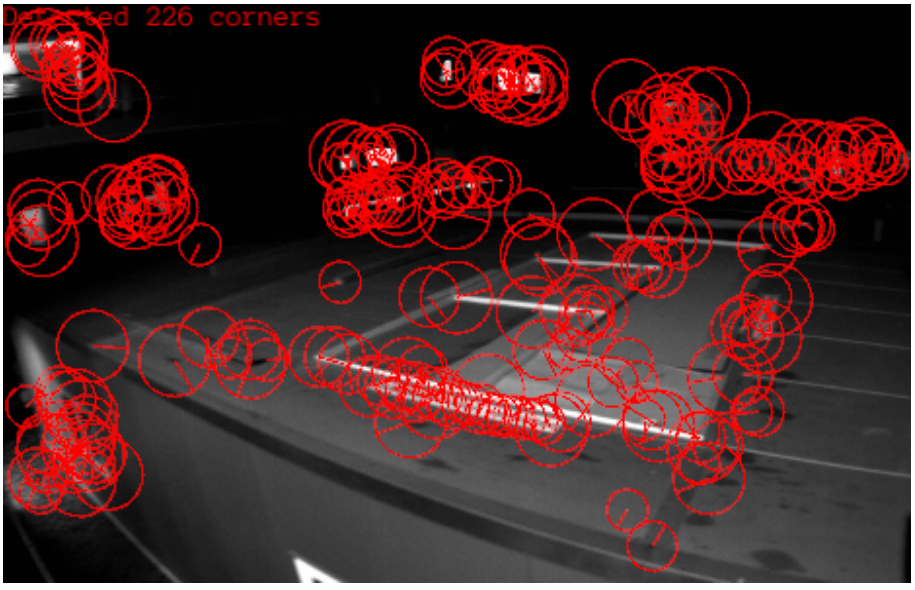


Figure 3.4: Image pyramid[21]



(a) Scale-aware feature without non-maximal subpression



(b) Scale-aware feature with non-maximal subpression

Figure 3.5: With/without non-maximal subpression

In order to create features in different scales, for each new keyframe, we construct the image pyramid, see Fig 3.4, and then extract orb features from it. Since we create features in different scale, some corners may generate more than one feature (features in different scale), this may lead to some features with higher weight than the others (features are not evenly distributed), and the pose estimations may contain bias. We have implemented non-maximal subpression to solve this problem. We keep the feature with highest score and discard others in a small region, see Fig 3.5.

3.2.1.4.2 Feature tracking

To track features, we reproject local mappoints on the current keyframe. We also build new mappoints by triangulating unmatched keypoints on the current frame and the connected frames of the previous frame.

3.2.1.4.3 Mappoints Optimization

Since the pose of the newly created keyframe is determined by Basalt, it is already very accurate locally, we then only optimize the positions of mappoints which can be observed by the newly created keyframe. Comparing to local bundle adjustment (a few thousands mappoints and dozen of poses), we optimize at most few hundreds mappoints which are much faster than local bundle adjustment, while keeping high accuracy. The optimization is done by minimizing the reprojection error. When a point i which is hosted in frame h, is detected in target frame t at pixel coordinate $z_{i,t}^{pix}$, the residual is defined as[33],

$$\mathrm{res}_{i,t} = z_{i,t}^{\mathrm{pix}} - \pi_t(T_{wt}^{-1} P_i^w) \tag{3.13}$$

where π_t is the projection function of frame t and P_i^w is the world coordinate of point i.

Then we can construct an error function by using different M-estimator and solve it by using different optimization schemes, including Newton-Gaussian method and LM method. Finally, we minimize the error function with respect to P_i^w.

3.2.1.4.4 Recent map points culling

All newly created map points are just considered as temporary mappoints. In order to be retained in the global map, it has to be observed during the first n keyframes (n=3 in our case) after creation. This test is used to make sure that the mappoints are trackable and created correctly.

3.2.1.4.5 Create more map points

After projecting local mappoints (hosted by covisiable frames of previous frame) on current frame, new edge between current and older frames have been created. After that, we triangulate more mappoints between the current frame and its covisiable frames (share at least 15 mappoints).

3.2.1.4.6 Pass to Loop-closer

After updating the global map, we pass the keyframe to loop-closer to detect if there are loops between the map and the new keyframe.

3.2.1.5 Loop-closing thread

Loop-closing thread is used to search for loops with newly created keyframe. If a loop is found, compute the relative pose between the new keyframe and the loop keyframe by PnP-ransac, then project mappoints owned by the loop keyframe and its covisibility graph, and vice versa, to build loop edges. After that merge the deprecated mappoints. The next step is to update all poses in the map with loop information.

3.2.1.5.1 Bag of Words (BoW)

In order to detect loop, we should compare the keypoints in the current keyframe to all the keyframes in the global map. However, feature matching is expensive, especially we need to perform ransac to remove outliers. Instead of comparing the current keyframe to every keyframe, we compute the bag of words vector first to obtain loop candidate keyframes.

As mentioned before, each binary feature descriptor is a vector composed of 0 and 1. In our case, each binary feature describtor is 256-bit. We use a hash function to hash it into 32-bit vector (called word). The idea is, if some descriptors are similar, they should hash to the same word. The i-th letter in a word is defined as,

$$\text{hash\_word}[i] = \text{descriptors}[P(i)] \tag{3.14}$$

where P is a permutation function, for example $P(i) = j \in \{0, ..., 255\}$

The bag of word vector for one image is defined as,

$$\text{BoW} = \{(\text{word}_i, \frac{n_i}{N_f}) | i \in \text{index of all words}\} \tag{3.15}$$

where n_i is number of $word_i$ appeared in the image and N_f is the number of keypoint of frame f.
To compute the similarity of two keyframes, let Ω to be a set of word index that both two BoW vectors contain. The similarity score can be visualized in Fig 3.6 and is defined as,

$$\text{score}(f_1, f_2) = \sum_{i \in \Omega} \min(\frac{n_{f_1,i}}{N_{f_1}}, \frac{n_{f_2,i}}{N_{f_2}}) \tag{3.16}$$

The range of the score is

$$0 \leq \text{score}(f_1, f_2) \leq 1 \tag{3.17}$$

	Common word id							
	0	2	3	4	5	6	9	
BoW1	3	1	6	4	1	6	1	
BoW2	2	3	2	4	2	9	1	
score	(2+1+2+4+1+6+1)/256 = 0.07							

Figure 3.6: Concept visualization of calculating BoW similarity.

$score(f_1, f_2) = 1$ means two keyframes having exactly the same word with the same weight. In the contrary, $score(f_1, f_2) = 0$ means two keyframes sharing no word. Then, loop candidate keyframes for current keyframe can be selected as follows,

$$\text{candidates} = \{\text{kf} \mid score(\text{curr kf}, \text{kf}) \geq T_{score} \wedge \sim \text{kf} \in \text{covisiable kf of curr kf}\} \quad (3.18)$$

where T_{score} is the threshold.

Since T_{score} can vary for different keyframes, one way to define T_{score} for frame i is,

$$T_{score}(kf_i) = min(\{score(\text{kf}_i, \text{kf}) | \text{kf} \in \text{covisiable kf of kf}_i\}) \quad (3.19)$$

We discard all those query results whose score is lower than T_{score}. This action can significantly reduce the number of loop candidate keyframes.

3.2.1.5.2 Loop detection

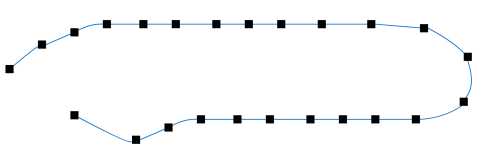
To detect loops, we take the following steps together with an example Fig 3.7

1. Query the hashbow database for loop kf candidates, keeping the candidates with loop score larger than min_score. The min_score is the smallest inner product of BoW between the current keyframe and its connected keyframes. See Fig 3.7b.

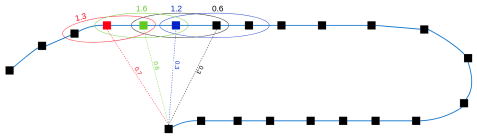
2. Form consistent groups for current keyframe. A consistent group consists of loop kf candidates and its connected keyframes. Calculate the accummulated score for each consistent group (sum over all scores in the group). In order to limit the number of loop kf candidates, we keep only loop kf candidates with the score larger than 0.75* best accumulated score. See Fig 3.7b.

3. Keep the consistent group if (1) the loop score of the loop kf candidates is the largest in the group and (2) consistent group intersects at least one of the previous consistent group. Consistent score is used to trace the age of one consistent group. See Fig 3.7c, Fig 3.7d.

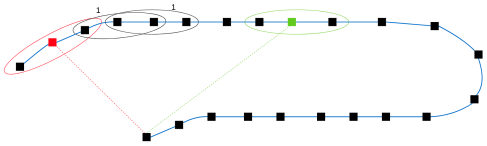
4. Loop kf candidates will be considered as loop kf if its consistent score is larger than or equal to 2 (sometimes is 3).



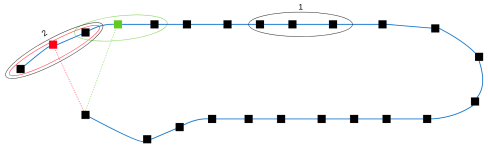
(a) Without loop detection.



(b) At time t, four query results with loop score 0.7 (red), 0.6 (green), 0.3 (blue) and 0.3 (black) respectively. The best accumulated score is 1.3, so the keyframes in red, green and blue are good loop kf candidates



(c) At time t+1, two consistent groups with consistant score 1, two new loop keyframe candidates (red, green).

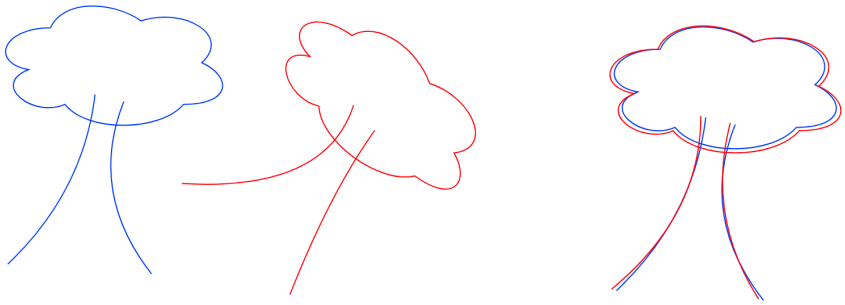


(d) At time t+2, two consistent groups with consistent score 1 and consistant score 2 respectively, two new loop keyframe candidates (red and green).

Figure 3.7: Steps of loop detection.

3.2.1.5.3 Relocalization

After getting a set of keyframe candidates for loop-closing (we will call them loop keyframe candidates), we then calculate their relative pose. There are two different methods to do so, (1) 3D-3D and (2) 3D-2D.



(a) Before (b) After

Figure 3.8: Concept visualization of ICP

(1) 3D-3D, use Iterative Closest Point (ICP) method[22] to align two local maps (one is constructed by current frame and its covisiable keyframe, another other is constructed by kf candidate and its covisiable keyframe),see Fig 3.8. This method is usually used in

lidar slam. However, in our case, the maps are not dense, so the result of this method is not guaranteed to align two maps correctly.

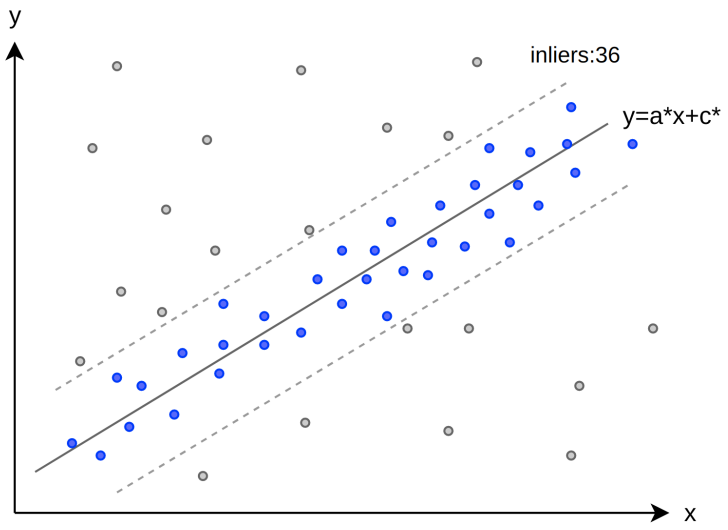


Figure 3.9: Concept visualization of RANSAC

(2) 3D-2D, use Perspective-n-Point (PnP) method[34] to estimate the pose of a camera given a set of 3D points in world coordinate and their corresponding bearing vector. However, PnP method is sensitive to outliers. Thus, we use RANSAC to make the solution more robust. RANSAC is an iterative method to estimate parameters that the samples contain outliers. It would keep sampling n points for PnP to estimate the pose until it passes the test (for example, more than 20 inliers). It can also filter outliers at the same time, see Fig 3.9. Note that RANSAC is a non-deterministic algorithm which only provides us a solution which passes the test, so we still need to minimize the reprojection error.

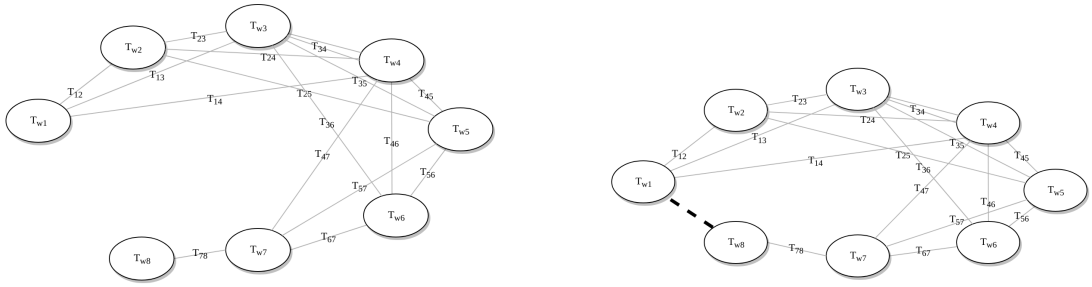
In our thesis, we will use the second method to find the relative poses between current keyframe and loop keyframe candidates.

3.2.1.5.4 Pose graph optimization

A global bundle adjustment is the most accurate way to update the map, however, it is slower when the map is getting larger and it is no longer able to be a real-time algorithm. Instead of performing global bundle adjustment, we perform pose graph optimization which only takes the relative pose edge (from basalt nonlinear factor) and loop edge into account. The residual is defined as,

$$res_{h,t} = ln(T_{wt}^{-1} T_{wh} T_{ht}^{edge}) \tag{3.20}$$

During the optimization, in order to fix the gauge freedom, we need to fix the pose of first keyframe. In order to merge the loop, we also need to fix the nodes (connected by the loop



(a) without loop edge (b) with loop edge

Figure 3.10: Pose graph with/without loop edge.

edge) or the relative pose between them to get rid from the local minimum of the error function. We have used ceres to optimize the error function.

3.2.1.5.5 Global Bundle Adjustment + Non-linear factor

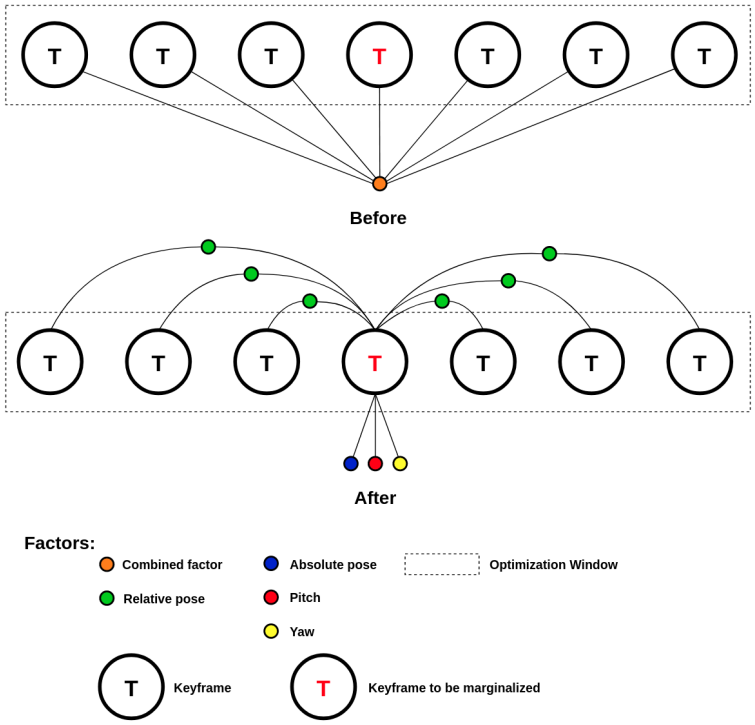


Figure 3.11: Visualization of non-linear factor extraction

In tracker thread, we marginalize keyframe which is outside the optimization window. We can recover the measurements and the corresponding uncertainties from the information matrix and save it for later use[33]. In theory, we can extract the uncertainty of absolute position, roll, pitch and yaw from the information matrix. However for VIO, absolute

position and yaw can not be determined but pitch, due to the gravity, is always constant and downward. Instead of extracting the absolute position and its covarience, we recover the measurement of relative pose of marginalized keyframe to the rest of the keyframe in the optimization window.

The information matrix of extracted factor can be divided into two ways and both methods obtain the same closed-form solution. The first way is to minimize the Kullback-Leibler divergence (KLD) between the recovered distribution and the original distribution[33]. The second way is to linearize the target residual function to obtain the Jacobian (J_{target}) first and propagate the uncertainty of poses to target function by

$$\Sigma_{\text{target}} = J_{\text{target}} \Sigma_{\text{poses}} J_{\text{target}}^T \tag{3.21}$$

Then, the information matrix (H_{target}) is, is $\Sigma_{\text{target}}^{-1}$

$$H_{\text{target}} = \Sigma_{\text{target}}^{-1} \tag{3.22}$$

To recover relative pose factor and pitch factor between marginalized keyframe and all other keyframes in the optimization window, we define the residual functions for relative pose and roll pitch,

$$\mathbf{r}_{\text{rel}}(s, z_{\text{rel}}^{i,j}, i, j) = ln(z_{\text{rel}} T_j^{-1} T_i) \tag{3.23}$$

$$\mathbf{r}_{\text{rp}}(s, z_{\text{rp}}^{i}, i) = ln(z_{\text{rp}} R_i^T (0,0,1)^T) \tag{3.24}$$

Then, the information matrix for $\mathbf{r}_{\text{rel}}(s, z_{\text{rel}}^{i,j}, i, j)$ and $\mathbf{r}_{\text{rp}}(s, z_{\text{rp}}^{i}, i)$ can be in the following,

$$H_{\text{rel}}(i,j) = (\frac{\partial \mathbf{r}_{\text{rel}}^{i,j}}{\partial s} \Sigma_{\text{poses}} \frac{\partial \mathbf{r}_{\text{rel}}^{i,j}}{\partial s}^T)^{-1} \tag{3.25}$$

$$H_{\text{rp}}(i) = (\frac{\partial \mathbf{r}_{\text{rp}}^{i}}{\partial s} \Sigma_{\text{poses}} \frac{\partial \mathbf{r}_{\text{rp}}^{i}}{\partial s}^T)^{-1} \tag{3.26}$$

The $\frac{\partial \mathbf{r}_{\text{rel}}^{i,j}}{\partial s}$ can be calculated in a way similar to deriving eq 2.41 and eq 2.42, see appendixA.1 and appendixA.2 for the derivation.

$$\frac{\partial \mathbf{r}_{\text{rel}}^{i,j}}{\partial s} = \begin{pmatrix} 0_{6\times6} & \dots & 0_{6\times6} & \frac{\partial \mathbf{r}_{\text{rel}}^{i,j}}{\partial s^i} & 0_{6\times6} & \dots & 0_{6\times6} & \frac{\partial \mathbf{r}_{\text{rel}}^{i,j}}{\partial s^j} & 0_{6\times6} & \dots & 0_{6\times6} \end{pmatrix} \tag{3.27}$$

$$\frac{\partial \mathbf{r}_{\text{rel}}^{i,j}}{\partial s^i} = J_{r,\text{decoupled}}^{-1} \begin{pmatrix} R_i^T & 0 \\ 0 & R_i^T \end{pmatrix} \tag{3.28}$$

$$\frac{\partial \mathbf{r}_{\text{rel}}^{i,j}}{\partial s^j} = J_{r,\text{decoupled}}^{-1} \frac{\partial s_i}{\partial s_j} \begin{pmatrix} R_i^T & 0 \\ 0 & R_i^T \end{pmatrix} \tag{3.29}$$

The $\frac{\partial \mathbf{r}_{\text{rp}}^{i}}{\partial s}$ can be calculated by eq 2.40, see appendixB.1 for the derivation,

$$\frac{\partial \mathbf{r}^i_{\text{rp}}}{\partial s} = \begin{pmatrix} 0_{6\times6} & \dots & 0_{6\times6} & \frac{\partial \mathbf{r}^i_{\text{rp}}}{\partial s^i} & 0_{6\times6} & \dots & 0_{6\times6} \end{pmatrix} \tag{3.30}$$

$$\frac{\partial \mathbf{r}^i_{\text{rp}}}{\partial s^i} = \begin{pmatrix} -R_{01} & R_{00} & 0 \\ -R_{11} & R_{10} & 0 \end{pmatrix} \tag{3.31}$$

Since the optical flow features built in the tracker thread and the ORB features extracted in Mapper thread are statistically independent, this allows us to reproject error function ($E_{\text{reproj}}(s)$) as defined in eq 3.13, together with the error term ($E_{\text{nfr}}(s)$) from the recovered non-linear factors yields the objective function,

$$E_{\text{total}}(s) = E_{\text{reproj}}(s) + E_{\text{nfr}}(s) \tag{3.32}$$

$$E_{\text{reproj}}(s) = \sum_{i\in\mathcal{P}, t\in obs(i)} (r^{it}_{\text{reproj}})^T \Sigma_{it} r^{it}_{\text{reproj}} \tag{3.33}$$

$$E_{\text{nfr}}(s) = \sum_{(i,j)\in\mathcal{R}} (\mathbf{r}^{i,j}_{\text{rel}})^T H^{i,j}_{\text{rel}} \mathbf{r}^{i,j}_{\text{rel}} + \sum_{i\in\mathcal{K}} (\mathbf{r}^i_{\text{rp}})^T H^i_{\text{rp}} \mathbf{r}^i_{\text{rp}} \tag{3.34}$$

$\mathcal{P}$ is a set of all mappoints in map, $\mathcal{R}$ is a set of all pairs of relative pose factor and $\mathcal{K}$ is a set of all pitchs factor.

The global bundle adjustment together with non-linear factor runs in an extra thread (not in Tracker, Mapper and LoopCloser). Since Mapping was active during the global bundle adjustment, which means that there might be new keyframes not included in the global bundle adjustment so these new keyframes are not consistent with the updated map. We correct these new keyframes by multiplying the change of the pose of the last keyframe in the global bundle adjustment.

3.3 Basalt-SLAM for Multi-cameras

If cameras are in the same place, they will share the same scene. In this work, we detect loops in maps built by different maps and merge them if loops are detected, and all cameras share the same global map so they update the same map and at the same time use the same map to do localization.

3.3.1 Purposed structure

The purposed structure is showed in Fig 3.12.

1. Each camera has its own tracker, mapper, and loopcloser, but shares the same global map. Each keyframe has its own sequence id in order to distinguish where it comes from.

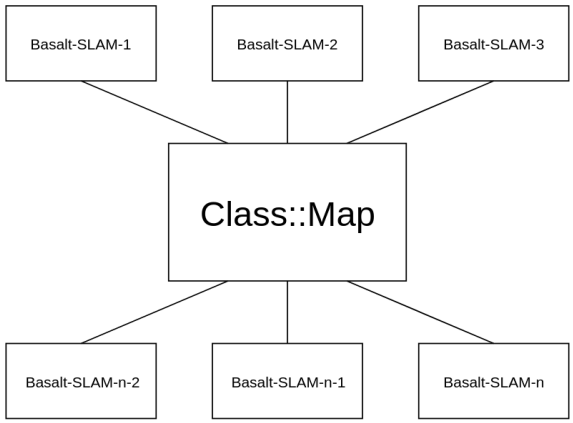


Figure 3.12: Purposed structure of multi camera slam

2. Every newly created keyframe detects loop keyframe candidates by Hashbow. This prevents to perform expensive feature matching to every keyframes in th global map.

3. If loops are detected, compute the Euclidean transformation $[R|t]$ for the new keyframe.

4. Pause all sequences' mapper.

5. Wait until all mapper have been paused, then perform pose graph optimization and set the poses of new keyframe, loop keyframe and its first frame as constant during the pose graph optimization.

6. Run global bundle adjustment in parallel.

7. Resume all sequences' mapper.

Chapter 4

Evaluation

To evaluate our new method, we perform experiments on two stereo + IMU datasets, (1) the well-known EuRoC dataset[6] and the TUM VI Benchmark[25]. All experiments have been conducted on an Intel Core i7-8850H running at 2.6GHz with 32GB RAM.

We compare our new approach to Basalt-VIO[33] and Basalt-SLAM[33]. We will use absolute trajectory error (ATE)[29]. Let P_i to be the estimated pose at time step i, Q_j to be the ground truth trajectory at time step j, then pose error (F_i) for time step i is computed as follows,

$$F_i(S) \equiv Q_i^{-1} S P_i \tag{4.1}$$

The root mean square error (RMSE) for the trajactory is,

$$\text{RMSE}(F_{1:n}(S)) \equiv (\frac{1}{n} \sum_{i=1}^{n} \| \text{ trans}(F_i(S)) \|^2)^{\frac{1}{2}} \tag{4.2}$$

where trans(P) returns the translation of P.

The ATE of the trajectory is defined as,

$$\text{ATE} = \min_S(\text{RMSE}(F_{1:n}(S))) \tag{4.3}$$

EuRoC dataset was recorded by hex-rotor helicoptor with a front-down looking stereo camera with 200Hz and synchronized IMU measurements with 200Hz. The datasets took place in small working space (called V1 and V2) and in large machine hall (called MH). All EuRoC datasets at least come with 3D position ground truth.

The Machine hall dataset contains five sequences with different difficulties. MH_01 and MH_02 are "easy", these sequences are bright and with good texture. MH_03 is "medium", it contains fast motion. MH_04 and MH_05 are "difficult", these sequences contain not only fast motion, but also the scenes being relatively dark. Each small working space dataset

(V1 and V2) contains 3 sequences. V1_01 and V2_01 are "easy", these sequences contains only slow motion and the scenes are bright. V1_02 and V2_02 are "medium", these sequences were recorded with a lot fast motions. V1_03 and V2_03 are "difficult", these sequences contain fast motion and motion blur. V2_02 is excluded in our evaluation, it is because there are more than 400 frames can not be traced due to strong motion blur.

4.0.1 Proposed VIO and Basalt-SLAM

4.0.1.1 Accuracy

Sequence	MH_01	MH_02	MH_03	MH_04	MH_05	V1_01	V1_02	V1_03	V2_01	V2_02
Basalt VIO,stereo[33]	**0.09**	**0.05**	0.09	**0.11**	0.11	**0.04**	0.06	0.07	**0.04**	0.06
Proposed VIO,stereo	**0.09**	0.06	**0.08**	**0.11**	**0.08**	**0.04**	**0.04**	**0.05**	0.05	**0.04**
Basalt VI Mapping,stereo, KF[33]	0.08	0.06	**0.05**	0.11	**0.09**	**0.04**	**0.03**	**0.03**	**0.03**	**0.02**
Proposed SLAM (PGO),stereo, KF	0.08	**0.05**	**0.05**	**0.09**	**0.09**	0.05	0.04	0.05	0.05	0.05
Proposed SLAM (GBA),stereo, KF	**0.07**	**0.05**	**0.05**	**0.09**	**0.09**	**0.04**	**0.03**	0.05	0.04	0.04

Table 4.1: RMS ATE of the estimated trajectory in meters on the EuRoC dataset.

The resulting ATE is listed in Table 4.1. We use exactly the same parameters for both MH, V1 and V2 datasets in the evaluation.

Our proposed VIO vs Basalt-VIO. For the MH dataset, the accuracies of MH_01, MH_02, MH_03 and MH_04 are similar comparing to Basalt-VIO. However, MH_05 has been improved significantly ($\sim 25\%$), it is because there are many reflections in these sequences leading to false tracking on optical flow keypoints, the new approach can make sure that the tracked keypoints fulfill the geometric constraint. For the V1 and V2 datasets, the accuracies of V1_01 and V2_01 are similar comparing to Basalt-VIO. For V1_02, V1_03 and V2_02 have been improved considerably ($\sim 30\%$), it is because these sequences contain strong motion blur, the new method can provide good initial position to track the feature in the next frame. See Table 4.1 and Fig 4.1 for more details and visualization.

Our proposed SLAM vs Basalt-VIO. For the MH dataset, the accuracy is similar for MH_02 and has been improved significantly for MH_01, MH_03, MH_04 and MH_05 for 18% - 45%. For the V1 and V2 datasets, the accuracies of V1_01 and V2_01 are similar comparing to Basalt-VIO. For V1_02, V1_03 and V2_02 have been improved significantly ($\sim 40\%$). See Table 4.1 and Fig 4.2 for more details and visualization.

Our proposed SLAM vs Basalt-Mapping. For the MH dataset, MH_01, MH_02 and MH_04 have been improved a little ($\sim 20\%$), and remain unchanged for MH_03 and MH_05. For the V1 and V2 datasets, V1_01 and V1_02 provide similiar results, however, the accuracies of V1_03, V2_01 and V2_02 have decreased around $\sim 50\%$. This results suggest that our orb feature tracking in the mapper thread does not perform as good as the feature matching in Basalt-Mapping. V1 and V2 datasets are recorded in a small working space, so the triangulated mappoints are closed to the camera, small error on the

position of mappoints could lead to a large error on reprojection, which leads to failure on the feature tracking. The above situation would not appear on Basalt-mapping since it matches features by comparing all image pairs.

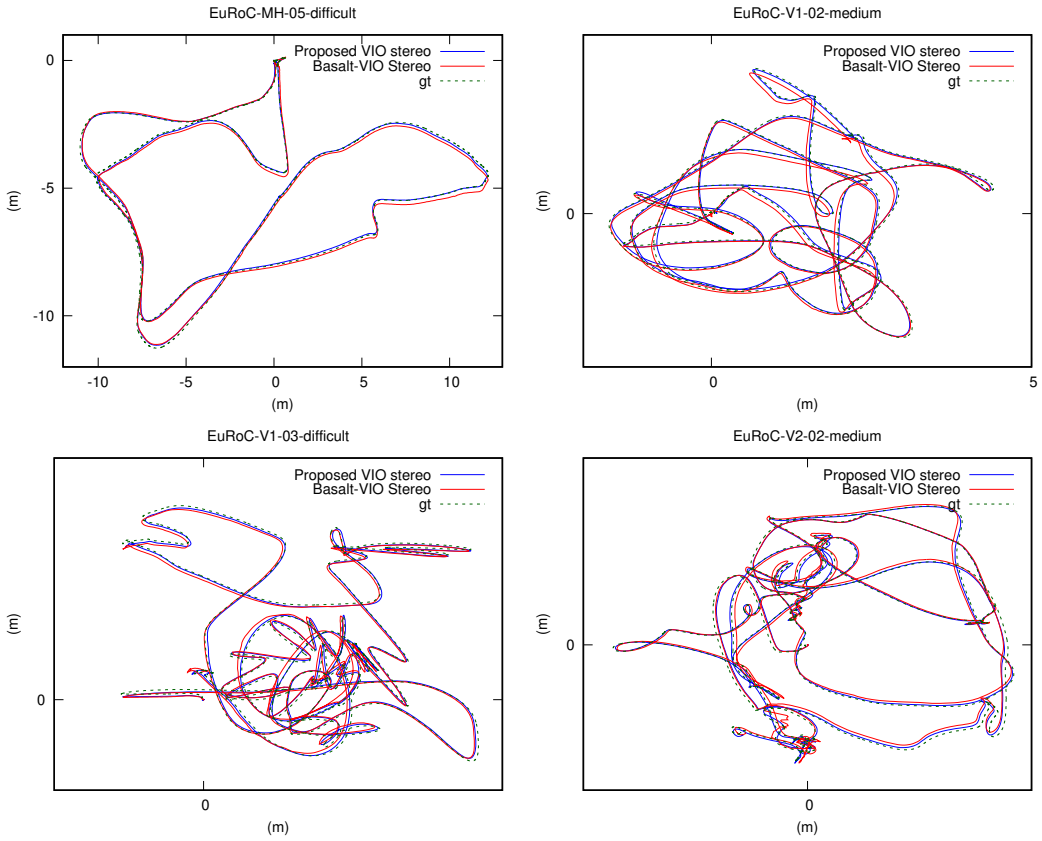


Figure 4.1: Comparison of the results from Basalt-VIO stereo to proposed VIO stereo (new method)

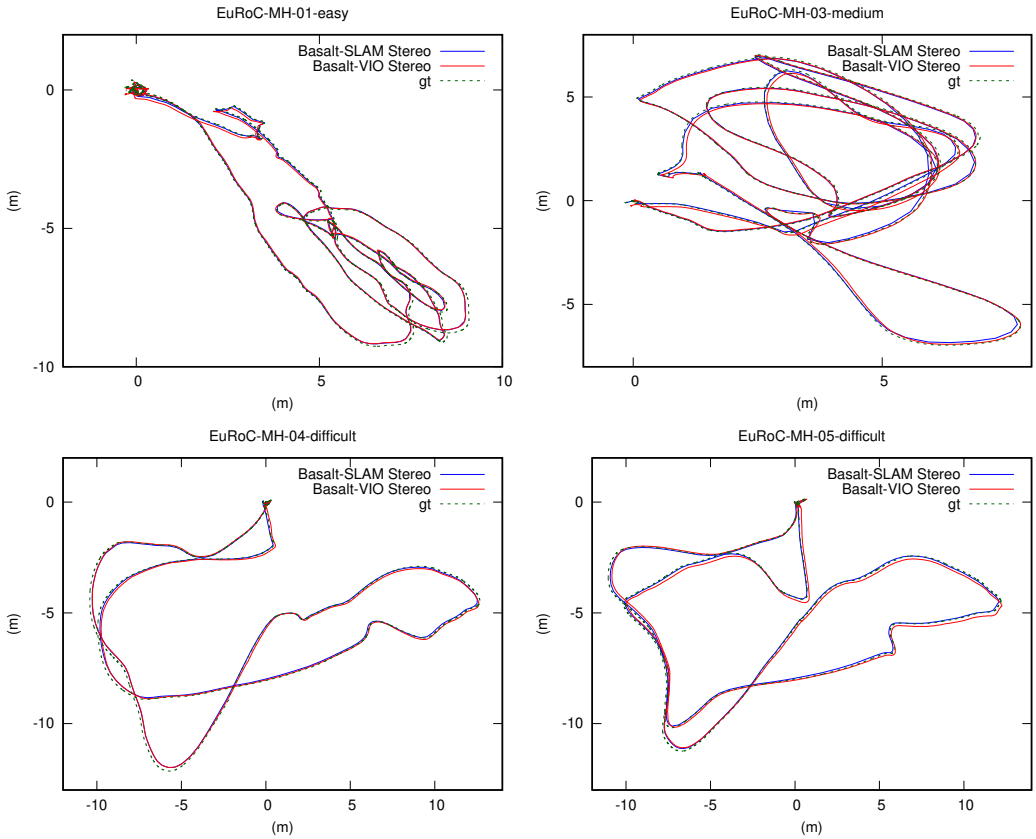
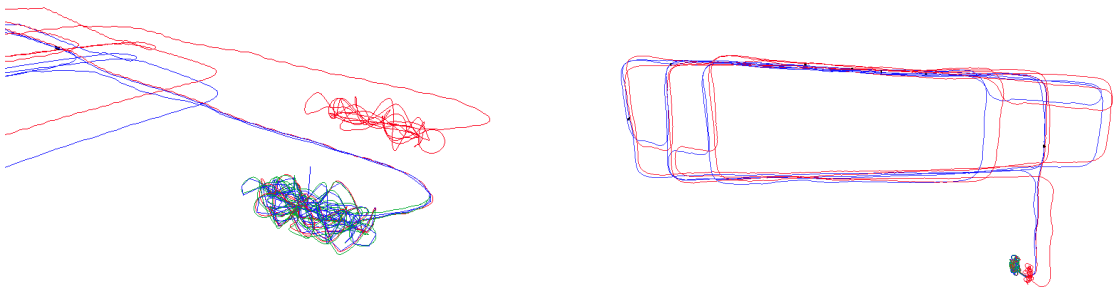


Figure 4.2: Comparison of the results from Basalt-VIO stereo to Basalt-SLAM stereo (new method)



(a) view 1 (b) view 2

Figure 4.3: TUMVI dataset result

4.0.1.2 Runtime

	MH_01	MH_02	MH_03	MH_04	MH_05
Basalt-VIO (Tracker thread)	20.2	19.6	21.5	19.8	19.9
Mapper (Mapper thread)	122.8	89.6	81.6	70.1	69.1
-Feature tracking	65.9 (54%)	48.2 (54%)	44.4 (54%)	35.3 (50%)	35.8 (52%)
-Mappoints creation	34.9 (28%)	24.8 (28%)	22.2 (27%)	18.7 (27%)	19.5 (28%)
-Mappoints optimization	23.1 (19%)	16.5 (18%)	15.1 (19%)	16.1 (23%)	13.7 (20%)
Loop Closer(LoopCloser thread)	179.0	166.4	142.9	189.8	214.8
-Loop detection	15.4 (1%)	21.4 (2%)	19.1 (2%)	24.8 (2%)	9.1 (1%)
-Compute E3	161.1 (9%)	142.2 (13%)	120.3 (13%)	161.1 (13%)	203.6 (4%)
-Correct loop	179.0 (90%)	166.4 (85%)	142.9 (84%)	189.8 (85%)	214.8 (95%)

Table 4.2: Average runtime in mini-second on EuRoC MH dataset

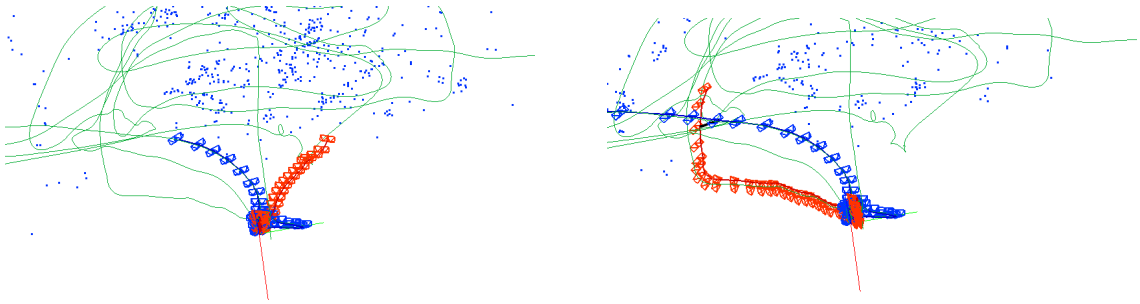
	MH_01	MH_02	MH_03	MH_04	MH_05
Tracker rate	49.5	51.1	46.5	52.1	50.4
-Keyframe rate	5.9	6.2	5.6	6.5	6.0
Mapper rate	8.1	11.2	12.6	14.3	14.5
Loop Closer rate	5.6	6.0	7.0	5.3	4.7

Table 4.3: Running rate in Hz on EuRoC MH dataset

To test the real-time capabilities, we measure the runtime in ms and running rate of different functions in different threading. The result is showed in Table 4.2 and Table 4.3. The tracker runs in a separated thread with around 50Hz, so the tracker can run in real-time. For mapper, it handles new keyframe with frequency around 12Hz, and the keyframe generation rate is around 6Hz, which means mapper can also run in real-time. For loop-closer, it runs in around 6Hz, and it is not likely that there are more than 6 completely new loops being detected in every second, which also means that the loop-closer is able to run in real-time.

One thing worths to mention is that the runtime of optimization in mapper is only around 20ms and it is around 500ms in orb-slam. It is because we use the result of Basalt-VIO to estimate the pose and we only optimize mappoints in the mapper instead of local bundle adjustment.

4.0.2 Basalt-SLAM for Multi-cameras



(a) Before loop detected (b) After loop detected

Figure 4.4: Visualization of loop merging.

When a loop is detected, it would close the loop and perform pose graph optimization. See Fig 4.4. The blue line denotes camera 1 and the red line denotes camera 2. The green line is the ground true of camera 1 and camera 2.

Quantitative and qualitative result

To evaluate multi Basalt-SLAM with EuRoC dataset, there are 10 combinations for 2-camera case and 3-camera case and 5 combinations for 4-camera case. We won't test 5-camera case due to limited computational resources. The results are listed in Table 4.4.

# of camera	MH_01	MH_02	MH_03	MH_04	MH_05
1	0.070	0.05	0.05	0.09	0.09
2	MH_01_02	MH_01_03	MH_01_04	MH_01_05	MH_02_03
2	0.069	0.099	0.096	0.091	0.067
2	MH_02_04	MH_02_05	MH_03_04	MH_03_05	MH_04_05
2	0.089	0.086	0.080	0.072	0.106
3	MH_01_02_03	MH_01_02_04	MH_01_02_05	MH_01_03_04	MH_01_03_05
3	0.072	0.079	0.089	0.081	0.088
3	MH_01_04_05	MH_02_03_04	MH_02_03_05	MH_02_04_05	MH_03_04_05
3	0.104	0.073	0.076	0.108	0.084
4	MH_01_02_03_04	MH_01_02_03_05	MH_01_02_04_05	MH_01_03_04_05	MH_02_03_04_05
4	0.080	0.098	0.098	0.094	0.091

Table 4.4: RMS ATE for multi Basalt-SLAM

All testing sequences set can detect and close the loops. Without global bundle adjustment (only pose graph optimization), the accuracy of case of 2-camera and 3-camera has decreased 20% comparing to 1-camera case, the case of 4-camera has decreased 30%. See Table 4.5. See Fig 4.5, Fig 4.6, Fig 4.7 and Fig 4.8 for more visualization on multi Basalt-SLAM.

# of camera	mean of RMS ATE (m)
1	0.070
2	0.085
3	0.085
4	0.092

Table 4.5: Mean of RMS ATE vs number of camera.

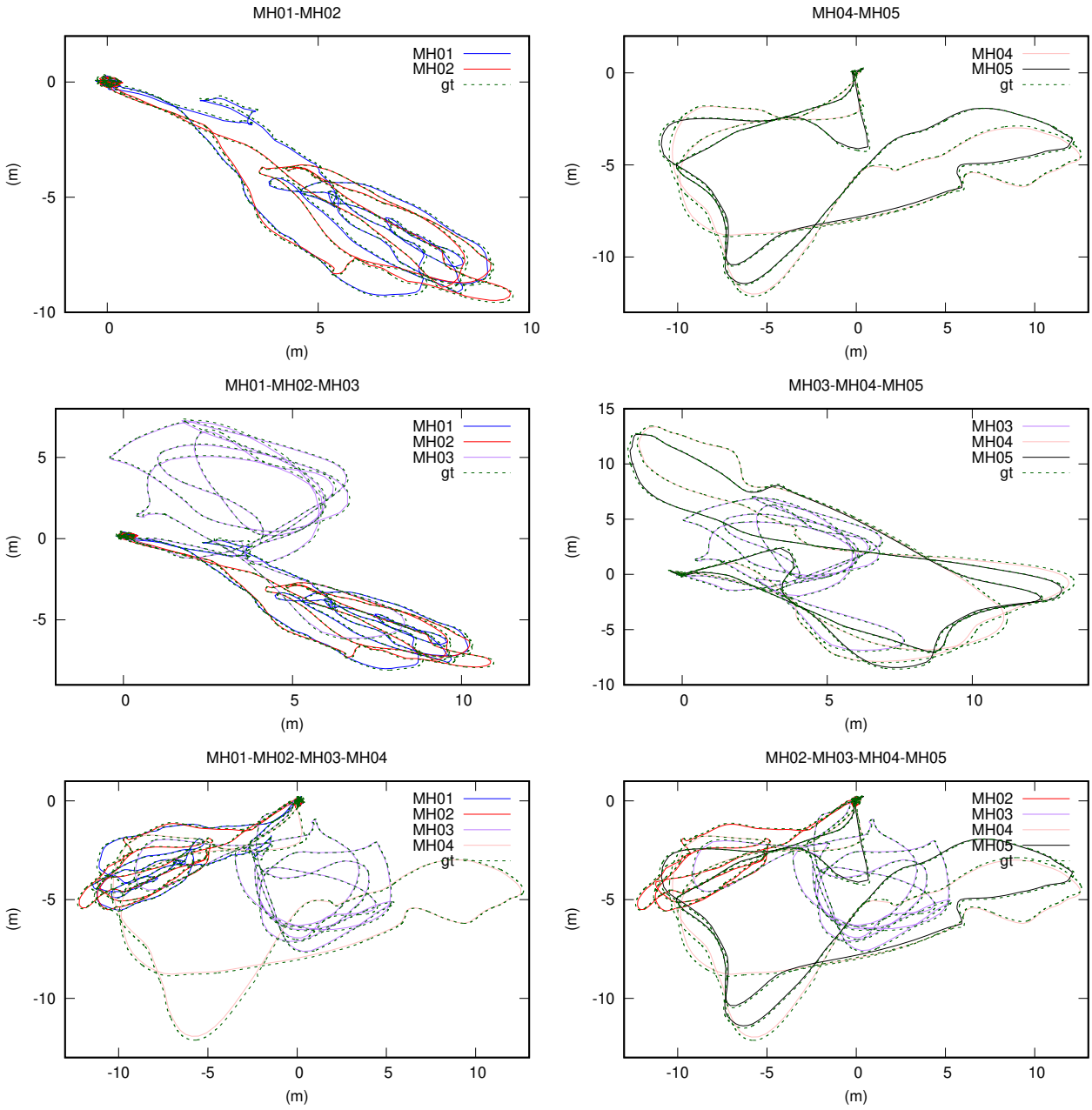
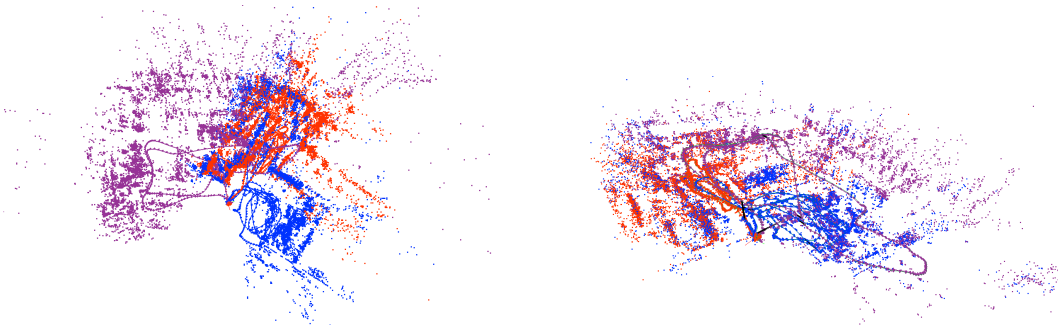
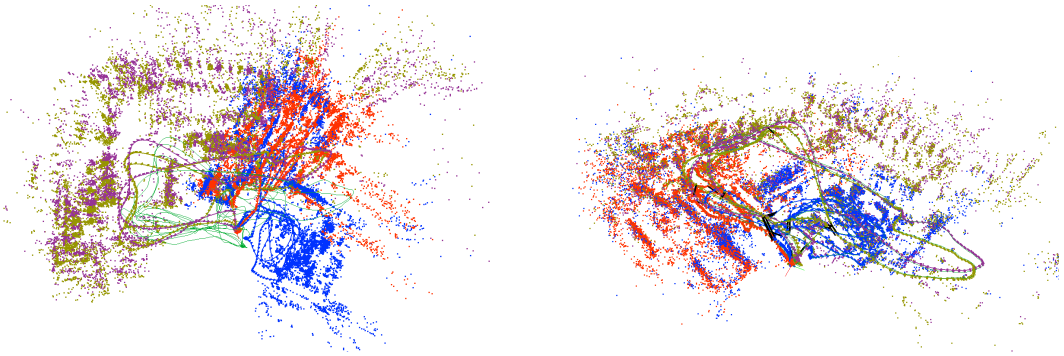


Figure 4.5: Visualization on multi Basalt-SLAM (birdeye)



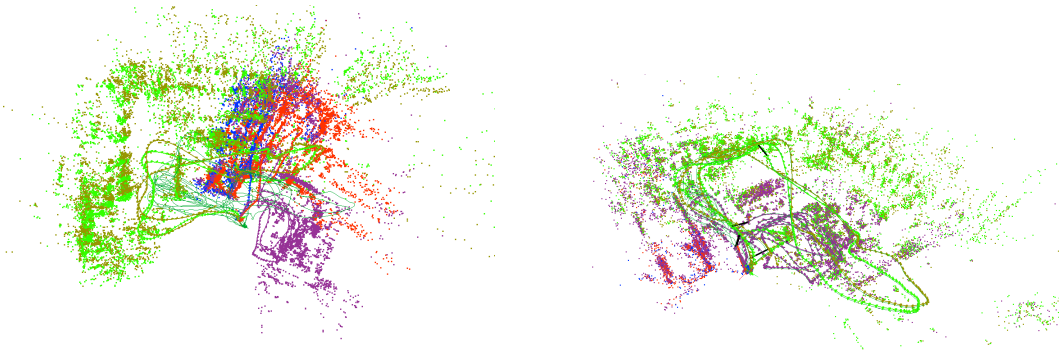
(a) Before loop detected (b) After loop detected

Figure 4.6: MH02-MH03-MH05



(a) Before loop detected (b) After loop detected

Figure 4.7: MH02-MH03-MH04-MH05



(a) Before loop detected (b) After loop detected

Figure 4.8: MH01-MH02-MH03-MH04-MH05

Chapter 5

Conclusion

This thesis presents a visual-inertial SLAM approach for multi-camera systems. This new method combines the strengths of highly accurate visual-inertial odometry with loop detection.

Unlike other existing approaches, we estimate the mean and the information matrix of the set of non-linear factors, which best represents the marginalized keyframe from visual-inertial odometry. Then, these non-linear factors serve as a prior in the mapper thread for building a new keyframe. The mappoints, which are observed in the newly created keyframe, will be optimized. Pose-graph optmization is performed when a loop is detected. In our proposed method, there is no big difference between the result with and without global bundle adjustment. Moreover, our method also supports more than one device at the same time, as a result, all devices can be used and updated on the same map simultaneously.

We evaluate the performance of our method regarding the accuracy and the runtime. Compared to Basalt-Mapping[33], our method shows better trajectory estimation on the EuRoC Machine Hall dataset. The runtime analysis shows that the tracker thread among three threads is the bottle neck of the system. Compared to other slam systems such as orb-slam, our method can reduce the computational cost of frame tracking and optimization by decreasing the number of variables and thus it is suitable for large-scale mapping and loop detection.

Finally, this method can be used to provide a good initial guess of the pose for other systems, including direct slam or extremely accurate mapping. Since it is able to recognize the relative poses between different devices, it can be applied to various areas such as multi-player VR gaming and path-planner of multi-robot cleaner. One of the research directions in the future is to add more sensors into the system. We can apply this system to RGB-D cameras to build a dense reconstruction and thus it can be used for 3D object detection and obstacle detection in autonomous driving. To make the system more robust to motion blur, we can use an event camera to measure the local brightness change with

higher frame rate.

Appendix A

The Jacobian of relative pose error

Express the full left increment by decoupled right increment

$$
\begin{aligned}
\delta\xi_l &= log(T * expd(\delta\xi_r^d) * T^{-1}) \\
&= log(\begin{pmatrix} Re^{\delta\phi} & t + R\delta p \\ 0 & 1 \end{pmatrix} \begin{pmatrix} R^T & -R^T t \\ 0 & 1 \end{pmatrix}) \\
&= log(\begin{pmatrix} Re^{\delta\phi}R^T & -Re^{\delta\phi}R^T t + t + R\delta p \\ 0 & 1 \end{pmatrix}) \\
&= log(\begin{pmatrix} e^{ad_R\delta\phi}RR^T & -e^{ad_R\delta\phi}RR^T t + t + R\delta p \\ 0 & 1 \end{pmatrix}) \\
&= log(\begin{pmatrix} e^{R\delta\phi} & -e^{R\delta\phi}t + t + R\delta p \\ 0 & 1 \end{pmatrix}) \\
&= log(\begin{pmatrix} e^{R\delta\phi} & \hat{t}R\delta\phi + R\delta p \\ 0 & 1 \end{pmatrix}) \\
&= \begin{pmatrix} \hat{t}R\delta\phi + R\delta p \\ R\delta\phi \end{pmatrix} \\
&= \begin{pmatrix} 1 & \hat{t} \\ 0 & 1 \end{pmatrix} \begin{pmatrix} R & 0 \\ 0 & R \end{pmatrix} \begin{pmatrix} \delta p \\ \delta\phi \end{pmatrix} \\
&= \begin{pmatrix} 1 & \hat{t} \\ 0 & 1 \end{pmatrix} \begin{pmatrix} R & 0 \\ 0 & R \end{pmatrix} \delta\xi_r^d
\end{aligned}
\tag{A.1}
$$

and express the full left increment by basalt left increment

$$
\begin{aligned}
\delta\xi_l &= T \oplus \xi_l^{ba} \ominus T \\
&= log(\begin{pmatrix} e^{\delta\phi}R & t+\delta v \\ 0 & 1 \end{pmatrix} \begin{pmatrix} R^T & -R^T t \\ 0 & 1 \end{pmatrix}) \\
&= log(\begin{pmatrix} e^{\delta\phi} & -e^{\delta\phi}t + t + \delta v \\ 0 & 1 \end{pmatrix}) \\
&= log(\begin{pmatrix} e^{\delta\phi} & \hat{t}\delta\phi + \delta v \\ 0 & 1 \end{pmatrix}) \\
&= \begin{pmatrix} \hat{t}\delta\phi + \delta v \\ \delta\phi \end{pmatrix} \\
&= \begin{pmatrix} 1 & \hat{t} \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \delta v \\ \delta\phi \end{pmatrix} \\
&= \begin{pmatrix} 1 & \hat{t} \\ 0 & 1 \end{pmatrix} \delta\xi_l^{ba}
\end{aligned}
\tag{A.2}
$$

Compare A.1 and A.2, we obtain

$$
\delta\xi_l = \begin{pmatrix} 1 & \hat{t} \\ 0 & 1 \end{pmatrix} \delta\xi_l^{ba} = \begin{pmatrix} 1 & \hat{t} \\ 0 & 1 \end{pmatrix} \begin{pmatrix} R & 0 \\ 0 & R \end{pmatrix} \delta\xi_r^d \tag{A.3}
$$

$$
\delta\xi_l^{ba} = \begin{pmatrix} R & 0 \\ 0 & R \end{pmatrix} \delta\xi_r^d \tag{A.4}
$$

To calculate $\frac{\partial f(T)}{\partial \xi_l^{ba}}$,

$$
\begin{aligned}
\frac{\partial f(T)}{\partial \xi_l^{ba}} &= \frac{\partial f(Texpd(\xi_r^d))}{\partial \xi_r^d} \frac{\partial \xi_r^d}{\partial \xi_l^{ba}} \\
&= \frac{\partial f(Texpd(\xi_r^d))}{\partial \xi_r^d} \begin{pmatrix} R^T & 0 \\ 0 & R^T \end{pmatrix}
\end{aligned}
\tag{A.5}
$$

A.1 Calculate $\frac{\partial res}{\partial T_i}$

$$
\begin{aligned}
\frac{\partial logd(z_{rel}T_j^{-1}T_i expd(\xi_{r,i}^d))}{\partial \xi_{l,i}^{ba}} &= \frac{\partial logd(z_{rel}T_j^{-1}T_i expd(\xi_{r,i}^d))}{\partial \xi_{r,i}^d} \begin{pmatrix} R_i^T & 0 \\ 0 & R_i^T \end{pmatrix} \\
&= J_l(z_{rel}T_j^{-1}T_i)^{-1} \begin{pmatrix} R_i^T & 0 \\ 0 & R_i^T \end{pmatrix}
\end{aligned}
\tag{A.6}
$$

A.2 Calculate $\frac{\partial res}{\partial T_j}$

$$
\begin{aligned}
\frac{\partial logd(z_{rel}(T_j expd(\xi^d_{r,j}))^{-1}T_i)}{\partial \xi^{ba}_{l,i}} &= \frac{\partial logd(z_{rel}(T_j expd(\xi^d_{r,j}))^{-1}T_i)}{\partial \xi^d_{r,j}} \begin{pmatrix} R_j^T & 0 \\ 0 & R_j^T \end{pmatrix} \\
&= \frac{\partial logd(z_{rel} expd(-\xi^d_{r,j}) T_j^{-1} T_i)}{\partial \xi^d_{r,j}} \begin{pmatrix} R_j^T & 0 \\ 0 & R_j^T \end{pmatrix} \\
&= \frac{\partial logd(z_{rel} T_j^{-1} T_i expd(-ad_{T_{ij}} \xi^d_{r,j}))}{\partial \xi^d_{r,j}} \begin{pmatrix} R_j^T & 0 \\ 0 & R_j^T \end{pmatrix} \\
&= -J_l(z_{rel} T_j^{-1} T_i)^{-1} ad_{T_{ij}} \begin{pmatrix} R_j^T & 0 \\ 0 & R_j^T \end{pmatrix} \\
&= -J_l(z_{rel} T_j^{-1} T_i)^{-1} \begin{pmatrix} R_{ij} & \hat{t_{ij}} R_{ij} \\ 0 & R_{ij} \end{pmatrix} \begin{pmatrix} R_j^T & 0 \\ 0 & R_j^T \end{pmatrix} \\
&= -J_l(z_{rel} T_j^{-1} T_i)^{-1} \begin{pmatrix} R_i^T & \hat{t_{ij}} R_i^T \\ 0 & R_i^T \end{pmatrix}
\end{aligned}
\tag{A.7}
$$

Appendix B

The Jacobian of roll pitch error

B.1 Calculate $\frac{\partial \mathbf{r}^i_{rp}}{\partial s^i}$

Similar approach as AppendixA.1,

$$
\begin{aligned}
\frac{\partial \mathbf{r}^i_{rp}}{\partial s^i} &= \lfloor \frac{\partial}{\partial w}(z_{rp} R_i^{-1} e^{-w}(0,0,1)^T) \mid_{w=0} \rfloor_{x,y} \\
&= \lfloor \frac{\partial}{\partial w} \left(-R_i \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & -w \\ 0 & w & 1 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ -1 \end{pmatrix} \quad -R_i \begin{pmatrix} 1 & 0 & w \\ 0 & 0 & 0 \\ -w & 0 & 1 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ -1 \end{pmatrix} \quad -R_i \begin{pmatrix} 1 & -w & 0 \\ w & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ -1 \end{pmatrix} \right) \mid_{w=0} \rfloor_{x,y} \\
&= \lfloor \frac{\partial}{\partial w} \left(R_i \begin{pmatrix} 0 \\ -w \\ -1 \end{pmatrix} \quad R_i \begin{pmatrix} w \\ 0 \\ 1 \end{pmatrix} \quad \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \right) \mid_{w=0} \rfloor_{x,y} \\
&= \begin{pmatrix} -R_{01} & R_{00} & 0 \\ -R_{11} & R_{10} & 0 \end{pmatrix}
\end{aligned}
\tag{B.1}
$$

Bibliography

[1] M. Sanjeev Arulampalam, Simon Maskell, and Neil Gordon. A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking. *IEEE TRANSACTIONS ON SIGNAL PROCESSING*, 50:174–188, 2002.

[2] P. K. Atrey, M. A. Hossain, Abdulmotaleb El Saddik, and Mohan S. Kankanhalli. Multimodal fusion for multimedia analysis: a survey, 2010.

[3] Timothy D. Barfoot. *State Estimation for Robotics*. Cambridge University Press, USA, 1st edition, 2017.

[4] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. *ECCV*.

[5] Michael Bloesch, Michael Burri, Sammy Omari, Marco Hutter, and Roland Siegwart. Iekf-based visual-inertial odometry using direct photometric feedback. 2017.

[6] Michael Burri, Janosch Nikolic, Pascal Gohl, Thomas Schneider, Joern Rehder, Sammy Omari, Markus Achtelik, and Roland Siegwart. The euroc micro aerial vehicle datasets. *The International Journal of Robotics Research*, 35, 01 2016.

[7] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard. Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *IEEE Transactions on Robotics*, 32(6):1309–1332, 2016.

[8] Michael Calonder, Vincent Lepetit, and Pascal Fua. Brief: Binary robust independent elementary features, 2010.

[9] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse. Monoslam: Real-time single camera slam. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(06):1052–1067, jun 2007.

[10] Concept Derivation, , and Maria Isabel Ribeiro. Kalman and extended kalman filters:.

[11] E. Eade. Lie groups for computer vision. 2014.

[12] Jakob Engel, Vladlen Koltun, and Daniel Cremers. Direct sparse odometry. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PP, 07 2016.

[13] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, USA, 2 edition, 2003.

[14] Christoph Hertzberg, Ren Wagner, Udo Frese, and Lutz Schrder. Integrating generic sensor fusion algorithms with sound state representations through encapsulation of manifolds. *Information Fusion - INFFUS*, 14, 07 2011.

[15] Juho Kannala and Sami Brandt. A generic camera model and calibration method for conventional, wide-angle, and fish-eye lenses. *IEEE transactions on pattern analysis and machine intelligence*, 28:1335–40, 09 2006.

[16] B. Khomutenko, G. Garcia, and P. Martinet. An enhanced unified camera model. *IEEE Robotics and Automation Letters*, 1(1):137–144, 2016.

[17] G. Klein and D. Murray. Parallel tracking and mapping for small ar workspaces. In *2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, pages 225–234, 2007.

[18] Mingyang Li and Anastasios I. Mourikis. High-precision, consistent ekf-based visual-inertial odometry. *The International Journal of Robotics Research*, 32(6):690–711, 2013.

[19] David G. Lowe. Distinctive image features from scale-invariant keypoints, 2003.

[20] R. Mur-Artal, J. M. M. Montiel, and J. D. Tards. Orb-slam: A versatile and accurate monocular slam system. *IEEE Transactions on Robotics*, 31(5):1147–1163, 2015.

[21] Opencv. Image pyramids. 2020.

[22] Jana Prochzkov and Dalibor Martiek. Notes on iterative closest point algorithm. 04 2018.

[23] Edward Rosten and Tom Drummond. Fusing points and lines for high performance tracking. In *IN INTERNATIONAL CONFERENCE ON COMPUTER VISION*, pages 1508–1515. Springer, 2005.

[24] Thomas Schneider, Marcin Dymczyk, Marius Fehr, Kevin Egger, Simon Lynen, Igor Gilitschenski, and Roland Siegwart. Maplab: An open framework for research in visual-inertial mapping and localization. *IEEE Robotics and Automation Letters*, PP:1–1, 01 2018.

[25] D. Schubert, T. Goll, N. Demmel, V. Usenko, J. Stckler, and D. Cremers. The tum vi benchmark for evaluating visual-inertial odometry. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1680–1687, 2018.

[26] Ren Schuster, Christian Bailer, Oliver Wasenmller, and Didier Stricker. *Combining Stereo Disparity and Optical Flow for Basic Scene Flow*, pages 90–101. 01 2018.

[27] Joan Sol, Jeremie Deray, and Dinesh Atchuthan. A micro lie theory for state estimation in robotics. 12 2018.

[28] Hauke Strasdat, J. Montiel, and Andrew Davison. Scale drift-aware large scale monocular slam. volume 2, 06 2010.

[29] Jrgen Sturm, Nikolas Engelhard, Felix Endres, Wolfram Burgard, and Daniel Cremers. A benchmark for the evaluation of rgb-d slam systems. pages 573–580, 10 2012.

[30] Richard Szeliski. *Computer Vision: Algorithms and Applications*, volume 5. 01 2011.

[31] Nikolas Trawny and Stergios Roumeliotis. Indirect kalman filter for 3d attitude estimation a tutorial for quaternion algebra multiple autonomous robotic systems laboratory technical report number 2005-002 , rev . 57. *Engineering*, 01 2005.

[32] V. Usenko, N. Demmel, and D. Cremers. The double sphere camera model. In *2018 International Conference on 3D Vision (3DV)*, pages 552–560, 2018.

[33] V. Usenko, N. Demmel, D. Schubert, J. Stckler, and D. Cremers. Visual-inertial mapping with non-linear factor recovery. *IEEE Robotics and Automation Letters*, 5(2):422–429, 2020.

[34] Ping Wang, Guili Xu, Yuehua Cheng, and Qida Yu. A simple, robust and fast method for the perspective-n-point problem. *Pattern Recognition Letters*, 108:31 – 37, 2018.