Leibniz Universität Hannover
Institut für Photogrammetrie und Geoinformation

Investigations on the application of collaborative visual SLAM using dynamic Ground Control Points

Master Thesis

submitted by

Yi Huang

at 06. Dec. 2019

Professor: Prof. Dr.-Ing. Christian Heipke
Supervisor: M.Sc. Philipp Trusheim

Erklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und ohne fremde Hilfe verfasst und keine anderen Hilfsmittel als angegeben verwendet habe. Die vorliegende Arbeit ist frei von Plagiaten. Alle Ausführungen, die wörtlich oder inhaltlich aus anderen Werken entnommen sind, habe ich als solche kenntlich gemacht. Diese Arbeit wurde in gleicher oder ähnlicher Form noch bei keinem anderen Prüfer als Prüfungsleistung eingereicht und ist auch noch nicht veröffentlicht.

Hannover, den 06. Dec. 2019

Yi Huang

Abstract

With the development of autonomous driving, more and more people are studying the technology of simultaneous localization and mapping based on the use of multiple lightweight and cheap camera sensors. Car2X research has also been developed alongside autonomous driving, and information exchange has become an indispensable part. Therefore this thesis will combine information exchange and simultaneous localization and mapping. A vehicle that can transmit its own precise position information is referred to as a dynamic Ground Control Point (GCP), which is expected to assist in the positioning of receiving information vehicles. In order to investigate how to use dynamic GCPs in the collaborative visual simultaneous localization and mapping (CoSLAM), the simulation method is applied.

The simulated data include dynamic GCPs, dynamic cameras, and 3D static tie points. The entire implementation process is divided into three steps. The first step is to initialize the map. The exterior orientation of each camera in the first two frames is known, so using the matching 2D feature points and cameras' exterior orientation in all views of the first two frames to reconstruct 3D static tie points and calculate their covariance matrix. The second step is motion estimation. According to the matching relationship between 3D static tie point and 2D feature point in the new frame, the initial exterior orientation of the camera in the new frame is calculated by EPnP. Then the exterior orientation is optimized by the Levenberg-Marquardt algorithm and the corresponding covariance matrix could be obtained. The third step utilizes the new exterior orientation of each camera to update the old map points and generate new map points. The third and second steps are cycled until the end of the simulation.

Afterward, three scenarios of real road conditions are simulated in the experiment, and the difference between whether to use dynamic GCPs is analyzed. It is found that the camera's pose error grows slower and even changing its original error trend after using dynamic GCPs when the camera contains motion on both translation and rotation.

Keywords: CoSLAM, dynamic GCPs

Contents

List of Figures

List of Tables

Acronyms

SLAM	Simultaneous Localization and Mapping
GNSS	Global Navigation Satellite System
CoSLAM	Collaborative visual SLAM
GCPs	Ground Control Points
SFM	Structure from Motion
BA	Bundle Adjustment
EKF	Extended Kalman Filter
MonoSLAM	Monocular Simultaneously Localization and Mapping
PTAM	Parallelization of Tracking and Mapping
IMU	Inertial Measurement Unit
VO	Visual Odometry
DoG	Difference-of-Gaussian
SVD	Singular Value Decomposition
PnP	Perspective-n-Point
DLT	Direct Linear Transformation
GN	Gauss-Newton
LM	Levenberg-Marquardt
BoW	Bag of Words
RANSAC	Random Sample Consensus
RMSE	Root Mean Square Error

1. Introduction

Localization and mapping are essential parts of autonomous driving. Localization determines exterior orientation (pose), mapping integrates the surrounding environment into a model with the help of exterior orientation and observations. The problem of simultaneous localization and mapping is called SLAM [6]. For localization and mapping, the external sensors commonly used are Global Navigation Satellite System (GNSS), camera and laser scanner. It is well known that the above sensors are noisy and there are restrictions on use. Laser scanners accurately sense the environment, but because they are very expensive and cumbersome, they can not be widely used or mounted on small carriers. GNSS has a high demand for the environment. It is suitable for areas with wide fields of vision and fewer obstacles. As a result, it usually can not work well in narrow streets. Due to the great attenuation of satellite signals indoors, which results in reduced accuracy, thus GNSS generally does not work well indoors. However, cameras are more likely to work in an unknown environment and could perceive more information such as colors and textures. Hence in recent years, people have gradually explored more possibilities for lighter, smaller, and cheaper cameras [7]. This thesis also aims to investigate the application of the camera. The camera is used as the only external sensor for simultaneous localization and mapping, known as visual SLAM.

With the development of Car2X research in the past years, more and more information exchange will occur. According to the distribution of data information processing space, it can be divided into the centralized processing method and the distributed processing method [8]. In this thesis, localization is based on distributed processing of information, multiple independent cameras are used to improve the localization by collaboration between these sensors. Mapping is based on centralized processing of information, using all useful information from cameras to generate a more accurate global map. The multi-camera collaboration application in visual SLAM is called collaborative visual SLAM (CoSLAM) [9].

Vehicles with accurate position information are called dynamic Ground Control Points (GCPs). The specific task of the thesis is to investigate how dynamic GCPs with known coordinates in image and object space can be used in a CoSLAM by means of simulations. To investigate this question different scenarios should be generated using a given simulation tool. A scenario consists of multiple cameras observing each other and dynamic GCPs. The simulation tool handles three different objects:

1. Dynamic cameras: These are moving cameras that capture the images. The exterior orientation is unknown.

2. Dynamic GCPs: These are moving objects with known positions in a global coordinate system at all times.

3. Static tie points: These points are stable in position, their 3D position is unknown.

In the simulation tool, the number of cameras, dynamic GCPs and static tie points and their trajectories vary depending on the needs of the scenario. The simulation tool projects the object information into image space and provides the image coordinates of the seen objects. In addition, it also provides the 3D position of every dynamic GCP at each epoch. The main task of this thesis is that using the above data by the CoSLAM to calculate the position of every static tie point and the exterior orientation of every camera in each epoch. In the end, the effect of dynamic GCP on CoSLAM is evaluated by the correctness and accuracy of the results.

The main problem in the implementation process is how to calculate the camera exterior orientation and the application of dynamic GCPs. Since using only the camera as the external sensor means that there is no other information to predict its own motion, the localization can merely rely on the observations obtained by itself. From a mathematical point of view, there is no motion equation for the model, only the observation equation. Thus, dynamic GCPs are expected to improve the accuracy of cameras' exterior orientation and static tie points' position, i.e. controlling the drift of the error. Nonetheless, the number of dynamic GCPs seen by each camera per frame is different. How many dynamic GCPs should be set in order to maximize the positive impact on the results remains to be discussed. At the same time, the distribution of dynamic GCPs should also be analyzed, because the centralized distribution of dynamic GCPs may lead to poor results.

In order to make the structure of this thesis and the content of each chapter clearer, the overall structure of the thesis is presented as follows:

- In **chapter 2** the development of SLAM is described. It mainly introduces the work and important achievements of the predecessors in the field of SLAM and highlights the major breakthroughs made in the field of visual SLAM.

- In **chapter 3** the theoretical background for visual SLAM is presented. The basic knowledge required to apply the visual SLAM method is explained in detail (such as the theory and formula of photogrammetry and computer vision) and some of the knowledge needed in CoSLAM is also supplemented.

- In **chapter 4** the specific operations and implementation steps to achieve localization and mapping are explained. Principally expounding the implementation method and the setting of the parameters, as well as the application of simulated data.

- In **chapter 5** the form and content of the scenario and the setting of the parameters are described. The results of different scenarios about the application of dynamic GCPs are discussed and the correctness and accuracy of the results are also analyzed.
- In **chapter 6** the conclusion with the application of dynamic GCPs in CoSLAM is in made. Then the outlook on the next steps to be taken in the methodologies is given.

2. Related Work

SLAM allows a robot to perceive the unknown environment, build an environmental map and continuously determine its position according to the environmental map. In 1986 R.C. Smith and P. Cheeseman first propose the probabilistic SLAM problem [10], which is considered the key to achieving a truly autonomous mobile robot. Then in the early 1990s the research group of Hugh F. Durrant-Whyte discusses and solves an open problem (which can be seen as "which came first, the chicken or the egg?"): localization and mapping depend on each other, accurate localization depends on the correct map, and correct mapping needs accurate location [11]. This discovery stimulates the subsequent research on the SLAM algorithm in terms of computational complexity and approximate solution. In 1998 Kalman filter-based SLAM is combined with probabilistic localization and mapping by Thrun [12]. After that, the filter-based SLAM algorithm is widely used and becomes the mainstream of the SLAM algorithm [7]. After 2000, SLAM gradually replaces the laser scanner with various cameras as a new research direction, as computer processing performance has been improved significantly. Because of this, SLAM researchers get ideas from the Structure from Motion (SFM) problem [13] and start to introduce the optimization method bundle adjustment (BA) [14] into SLAM. The difference between the optimization method and the filtering method is that the optimization method is not an iterative process, but considers the information in all past frames, i.e. a difference between least squares and maximum likelihood. The first and second sections focus on the important methods of filter-based SLAM and SFM-based SLAM in visual SLAM.

In the past decades, most visual SLAM techniques have been in view of the assumption that the surrounding environment is static. In fact, the surrounding environment exists a lot of moving objects. When these visual SLAM techniques are applied to actual scenes, various problems or even failures occur. Therefore, in recent years, how to apply SLAM in a dynamic environment has become a direction of SLAM research. The third section mainly introduces the SLAM in a dynamic environment closely related to this thesis.

2.1. Filter-based SLAM

In 1990 Smith et al. firstly present the stochastic map which is the representation of uncertain spatial relationships between objects and use the Extended Kalman Filter (EKF) to find an exactness solution [15]. They use EKF to estimate the position of feature (landmark) and of the robot in the state space at the same time. Yet the disadvantage is obvious, the problem of high computational complexity always exist.

Monocular Simultaneously Localization and Mapping (MonoSLAM) is the first real-time monocular visual SLAM system [16]. MonoSLAM uses EKF as the filtering, tracks the sparse feature points, and updates their mean and covariance with state vector that contains the camera's current state and all landmarks. In EKF, the position of each feature point is subject to a Gaussian distribution and an ellipsoid can be used to represent its mean and uncertainty. One of the disadvantages of this algorithm is that sparse points are easily lost. The main disadvantage is that, no matter how the filter equation is sorted and calculated, its computational complexity is at least proportional to the square of the number of landmarks. The reason is that landmarks' locations are added to the estimated state vector, this is also the reason for difficulty meeting the requirements of constructing large-scale maps and real-time.

In order to be able to apply visual SLAM in a large number of landmarks, Montemerlo et al. [17] propose a new algorithm for updating particle filter which is called FastSLAM. It divides the joint SLAM state into the motion part and the conditional map part. For the purpose of reducing the sampling space, the pose of the robot is expressed by particles with different weights and pose state is recursively estimated with the aid of the particle filter method. In particular, the map is represented by an independent Gaussian distribution, the recursive estimation of the map state uses the EKF method. The advantage of the algorithm is that on the one hand the computational complexity is reduced. On the other hand, the particle filter method directly approximates the model, does not require the control vector and the observation to satisfy the Gaussian distribution. However, the disadvantage is how to determine the number of particles. A large number of particles means they require a large amount of memory and calculation time, but a small number of particles lead to inaccurate results [7].

2.2. SFM-based SLAM

Nistr et al. [18] publish the visual odometry (VO) to gradually solve SFM-based SLAM. For the first time, a system for ego-motion estimation in real-time of a single camera or stereo rig is introduced in detail, including feature extraction, feature matching, and robust estimation. In the following study, the front end mainly refers to the VO.

Klein and Murray [19] propose and implement the parallelization of tracking and mapping (PTAM), distinguish the front and back ends for the first time (the tracking needs real-time response image data, the map optimization is placed on the back end). PTAM is based on keyframes and two parallel processing threads. The keyframe means, instead of finely processing each image, several key images are stringed together to optimize their trajectory and map. Two parallel processing threads are tracking and mapping. Specifically, the tracking thread does not modify the map but uses known maps for fast tracking; while the mapping thread focuses on the creation, maintenance, and updating of the map. Even if the mapping thread takes a long time, the tracking thread still has a map to track (if the device is still within the scope of the built map). In addition, PTAM also implements the strategy of relocation. If the number of successful inliers is insufficient (such as image blurring, fast motion, etc.), the tracking fails. Then the relocation is started, the current frame already compares the thumbnails of keyframes and selects the most similar keyframe as the prediction of the current frame orientation. The disadvantage of PTAM is that the scene is small and the tracking is easy to lose when the camera moves fast or tracks moving objects.

In 2015 Mur-Artal et al. propose the monocular ORB-SLAM [20] and in 2016 expand ORB-SLAM to get ORB-SLAM2 which supports for stereo and RGBD sensors [21]. ORB-SLAM innovatively uses three threads to complete SLAM, which adds a separate loop closing thread to the PTAM algorithm framework. In addition, the PTAM algorithm framework has been improved: 1) ORB-SLAM tracking, mapping, relocation and loop closing all use uniform ORB features [22], ORB feature's calculation efficiency is better than SIFT [23] or SURF [24] and has good rotation and scaling invariance; 2) thanks to the use of the visibility graph, the tracking and mapping operations are concentrated in a local cross-view area, enabling them to operate in real-time on a wide range of scenes without depending on the size of the overall map; 3) uses a unified Bag-of-Words model to perform relocation, loop closing and indexing to improve detection speed; 4) improves the lack of PTAM (can only manually select the initialization from planar scenes) and proposes a new automatic robust system initialization strategy on the basis of model selection, allows reliable automatic initialization from planar or non-planar scenes. Yet the downside is that on the one hand, it takes a lot of time to calculate the ORB feature for each image and the three-threads architecture puts a heavy burden on the CPU. On the other hand, a sparse feature point map can only meet the positioning requirements, but can not provide navigation, obstacle avoidance or other functions.

2.3. SLAM in a dynamic environment

To deal with SLAM in a dynamic environment, mainly distinguishing between static and dynamic objects. That is motion segmentation which classifies dynamic and static features, which relies chiefly on computational geometric models (e.g., fundamental matrix, homography) and sometimes requires the help of an inertial measurement unit (IMU). There are five technologies to segment static and dynamic features: background-foreground initialization, geometric constraint, optical flow, ego-motion constraint, and deep learning [25]. The simulation in this thesis already contains distinguished static and dynamic points. Since the idea of the thesis is based on CoSLAM in Zou and Tan's paper, the motion segmentation of Zou and Tan's paper applies the reprojection error method of geometric constraint technology, thus only geometric constraint is introduced.

Zou and Tan [9] first present CoSLAM which uses multiple cameras to handle pose estimation and distinction between static and dynamic features. In terms of pose estimation, for the dynamic scene they design a multi-camera collaborative optimization scheme and add the dynamic point constraint to the optimization function (minimization of reprojection error). As for the distinction between static points and dynamic points, firstly all points are treated as static points and then are discriminated according to the reprojection error: if the reprojection error of a point for a single camera frame exceeds the threshold, the category of the point is marked as "unknown", then combined with the information of other cameras, it can be judged whether the point is a dynamic point. But the biggest limitation when using this method in the real world is to require time synchronization for each camera.

Tan et al. [26] use a similar approach with Zou and Tan [9], but they consider occlusion to make the results more robust. Projecting feature points to the current frame can compare appearance and structure, then it could be found that if this area has changed, i.e. whether the area may be occluded due to the change of perspective. The invalid 3D points can be deleted and updated in a timely and efficient manner so that the system can cope well with gradually changing scenes. The problem remains that fast moving objects can lead to failure, and the system can only work in a small range.

3. Theoretical Background

In 2016 Cadena et al. [1] talk about the past, present and future of SLAM. They not only evaluate and affirm the past work, but also propose the direction for the development of SLAM, as well as a typical system of SLAM is given in In Fig. 3.1. This system consists mainly of two parts: the front end and the back end. The front end performs data processing and integration, the back end performs inference estimation, the important loop closing step requires feedback from the back end and data from the front end to promote a more robust system.

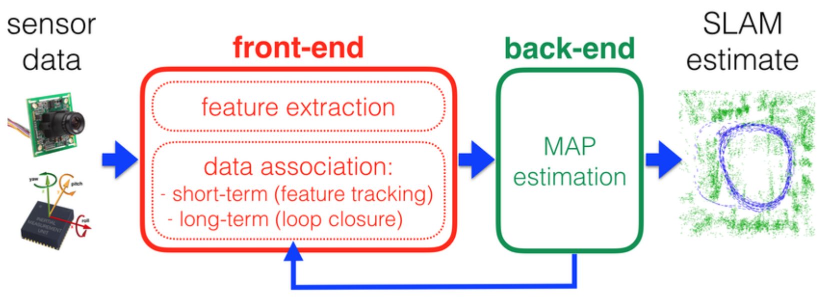


Figure 3.1.: Front end and back end in a typical SLAM system [1]

On the basis of the SLAM structure in the above Fig. 3.1, the basic visual SLAM framework is summarized in Fig. 3.2. The basic visual SLAM framework includes 5 components: sensor data, front end, back end, mapping and loop closing. Since this thesis is based on monocular cameras, the detailed description of these important components is only relevant to monocular cameras. Section 3.1 explains the staple model of the camera, including imaging principle, coordinate transformation, and distortion. The section 3.2 introduces the reading and preprocessing of camera data. The front end is presented in section 3.3 and is also known as VO. It expounds on how to quantitatively estimate the motion of the camera from images of adjacent frames. Section 3.4 describes the back end which optimizes the results of the front end. Loop closing in the section 3.5 usually determines whether the camera appears in the previous position by judging the similarity of the image, thus solving the problem of position drift over time. Mapping is based on different sensor types and application requirements. Section 3.6 briefly introduces the common maps.

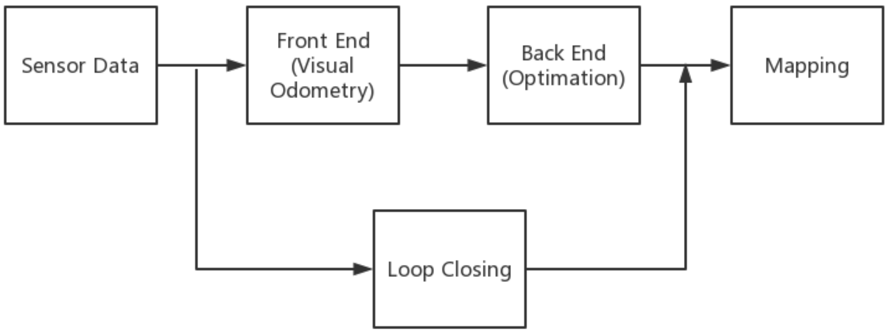


Figure 3.2.: Basic visual SLAM framework

3.1. Camera model

The process of a monocular camera mapping points in the world coordinate system to the image pixel coordinate system can be expressed by a simple pinhole model. However, by reason of the presence of the camera lens, the above process is distorted. Wherefore the next two sections focus on a pinhole model and distortion.

3.1.1. Pinhole camera model

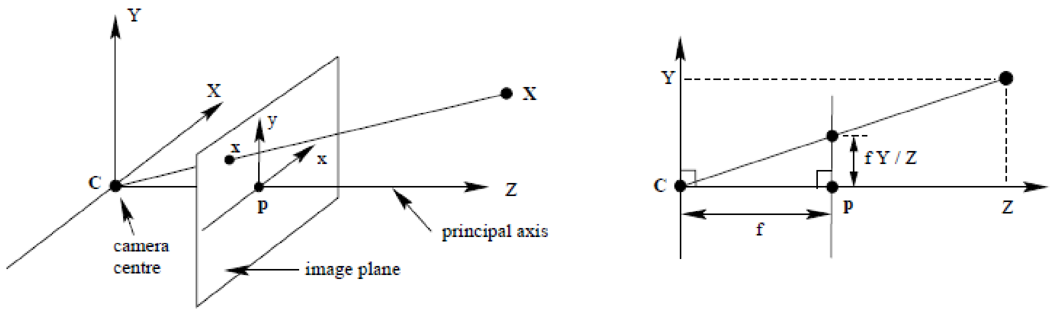


Figure 3.3.: Pinhole camera model [2]

The pinhole camera model in Fig. 3.3 shows the central projection of a point with global coordinate $\mathbf{X} = [X, Y, Z]^T$ onto the image plane $Z = f$. The line connecting point $\mathbf{X}$ and camera center **C** intersects the image plane at point **x**. Assuming that the principle point **p** is the origin of coordinate in the image plane, according to the similarity of triangles the coordinate of point **x** is $[f\frac{X}{Z}, f\frac{Y}{Z}, f]^T$. Ignoring the last item, the 2D camera coordinates of point **x** is $[f\frac{X}{Z}, f\frac{Y}{Z}]^T$. Yet actually it exists principal point offset shown in Fig. 3.4, the true image coordinates of point **x** is $[f\frac{X}{Z} + p_x, f\frac{Y}{Z} + p_y]^T$ [2].

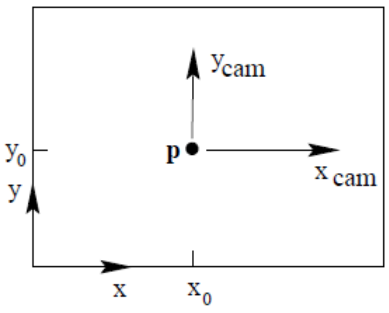


Figure 3.4.: Image (x, y) and camera (x_{cam}, y_{cam}) coordinates [2]

If point **X** and point **x** are represented in homogeneous coordinates, the central projection is able to be simply signified as a linear mapping between their homogeneous coordinates, which is expressed as a matrix multiplication:

$$\begin{pmatrix} fX + Zp_x \\ fY + Zp_y \\ Z \end{pmatrix} = \begin{bmatrix} f & 0 & p_x & 0 \\ 0 & f & p_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \tag{3.1}$$

$$K = \begin{bmatrix} f & 0 & p_x \\ 0 & f & p_y \\ 0 & 0 & 1 \end{bmatrix} \tag{3.2}$$

The matrix **K** is called calibration matrix, the 3 parameters of calibration matrix **K** are called interior orientation. The above formula is derived based on assumptions that camera center **C** is the origin of the global coordinate, it means that the point **X** is in the camera coordinate system, so $[X, Y, Z, 1]^T$ can be written as X_{cam}. Then the Equ. 3.1 can be represented as:

$$x = K\,[I \mid 0]\,X_{cam} \tag{3.3}$$

However generally, the camera center **C** is not the origin of the global coordinate system. Therefore point **X** should be convert from global coordinate system to the camera coordinate system through the transformation matrix $\mathbf{T}^{cam}_{global}$ first. The transformation matrix $\mathbf{T}^{global}_{cam}$ include 3×3 rotation matrix **R** that represents orientation of camera coordinate system and 3×1 translation vector **t** that represents camera center position in global coordinate system. Rotation matrix **R** and translation vector **t** together are called exterior orientation.

$$T^{global}_{cam} = \begin{bmatrix} R & t \\ 0^T & 1 \end{bmatrix} \tag{3.4}$$

$$T^{cam}_{global} = T^{global-1}_{cam} = \begin{bmatrix} R^T & -R^T t \\ 0^T & 1 \end{bmatrix} \tag{3.5}$$

The mapping of point **X** from global coordinate to camera coordinate is shown in Equ. 3.6:

$$X_{cam} = \begin{bmatrix} R^T & -R^T t \\ 0^T & 1 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} = \begin{bmatrix} R^T & -R^T t \\ 0^T & 1 \end{bmatrix} X \tag{3.6}$$

Combining Equ. 3.3 with the above formula 3.6 to get the following formula:

$$x \sim K\,R^T\,[I \mid -t]\,X \tag{3.7}$$

The projection matrix **P** is determined by the calibration matrix and the transformation matrix:

$$P = K\,R^T\,[I \mid -t] \tag{3.8}$$

3.1.2. Distortion

The lens in front of the camera affects the propagation of light during imaging. For example, irregular refraction of light is produced as it passes through the lens. The closer to the edge of the image, the more obvious the tendency that a straight line through the lens becomes a curve on the image. This phenomenon is called distortion. Distortion can be generally grouped into radial distortion and tangential distortion [27]. Radial distortion is induced by an imperfect lens manufacturing process that can result in defects in the shape of the lens. Radial distortion [3] mainly includes barrel distortion with positive radial displacement and pincushion distortion with negative radial displacement. Normal image without distortion (a), barrel distortion (b) and pincushion distortion (c) are shown in Fig. 3.5:

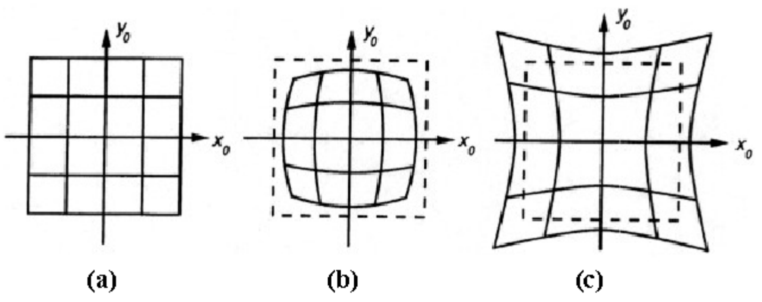


Figure 3.5.: Radial distortion [3]

A radial distortion model is exhibited in Equ. 3.9 and Equ. 3.10 [28]:

$$\Delta x_r = x_{cam}(k_1(x_{cam}^2 + y_{cam}^2) + k_2(x_{cam}^2 + y_{cam}^2)^2 + k_3(x_{cam}^2 + y_{cam}^2)^4) \tag{3.9}$$

$$\Delta y_r = y_{cam}(k_1(x_{cam}^2 + y_{cam}^2) + k_2(x_{cam}^2 + y_{cam}^2)^2 + k_3(x_{cam}^2 + y_{cam}^2)^4) \tag{3.10}$$

where x_{cam} and y_{cam} are arbitrary coordinates in the image coordinate system, k_1, k_2, k_3 are parameters of radial distortion.

Due to the fact that the lens is not parallel to the image plane, tangential distortion takes place [29]. In general, radial distortion affects the image a lot than tangential distortion, which means radial distortion is less credible and needs to be corrected [30]. A tangential distortion model is shown in Equ. 3.11 and Equ. 3.12 [28] with p_1, p_2 as parameters of tangential distortion:

$$\Delta x_t = 2p_1 x_{cam} y_{cam} + p_2((x_{cam}^2 + y_{cam}^2)^2 + 2x_{cam}^2) \tag{3.11}$$

$$\Delta y_t = p_1((x_{cam}^2 + y_{cam}^2)^2 + 2y_{cam}^2) + 2p_2 x_{cam} y_{cam} \tag{3.12}$$

3.2. Sensor data

In visual SLAM the common visual sensors include monocular, stereo and RGB-D cameras. A monocular camera has only one camera, a stereo camera has two cameras, and an RGB-D camera usually carries multiple cameras. In addition, RGB-D cameras could capture color images. In this thesis, only needed monocular cameras is presented.

The advantages of the monocular camera are the simple structure, low cost, besides that it is easy to calibrate and identify. The disadvantage is that in a single image, the true size of an object cannot be determined. If two images are obtained by the motion of the camera, the distance of the camera motion can be calculated. However, this distance is uncertain and diffs from the real distance by a scale. For this reason that in feature-based monocular slam, the monocular camera is usually initialized by panning. Then the absolute length of the relative translation is fixed to 1, the depth of objects can also be calculated by triangulation. Hence the size of the camera trajectory and the map can be gained in the monocular SLAM, although it still differs from the real trajectory and map by a real scale.

Calibration should be performed first before the experiment begins. The purpose is to establish the relationship between the global coordinate system and the image coordinate system. Therefore, the parameters required to be solved include interior parameters, exterior parameters, and distortion parameters. "Zhang's Algorithm" is a camera calibration method using a one-sided checkerboard [31]. This method not only conquers the disadvantages of high-precision calibration required by the traditional calibration but also supplies higher accuracy and more convenient operation compared to self-calibration. Thus, Zhang's algorithm is encapsulated as the function and widely used in computer vision. The actual steps for camera calibration refer to the paper [31].

3.3. Front end

VO estimates the relative pose of the camera on the basis of two adjacent frames. Since this estimate is affected by noise, the estimation error of the previous frame is added to the motion of the subsequent frame, this phenomenon is called drift. In other words, the cumulative error of VO over time leads to pose drift. Therefore, VO can only be used as the front end which estimates the rough result that used as the initial value of the back end. According to the difference of used image information the implementation method of VO can be grouped into the direct method using gray information and the feature-based method. The next two sections will focus on the feature-based method and briefly introduce the direct method for the reason that the feature method is steady and insensate to illumination and dynamic objects.

3.3.1. Direct method

The direct method doesn't depend on feature points because not only it takes a lot of time and computation to extract and match feature points, but also the process of extracting feature points from an image discards a lot of useful information in the image. Thence the direct method skips the step of extracting the feature points and estimates the motion by minimizing the photometric error instead of minimizing the reprojection error. In addition, because the direct method uses more information about image pixels, it is more robust in the poorly textured part than the feature-based method. The direct method saves the time of feature extraction, but if a large amount of image information is used for motion estimation, the large optimization problem in the later stage requires a large amount of computation to solve, so that the direct method can only achieve real-time through GPU acceleration.

According to the number of used pixels, the direct method is grouped into sparse direct method, dense direct method and semi-dense direct method [32]:

- Sparse Direct Method: Choosing sparse keypoints but not requiring descriptors, so the calculation and matching of descriptors can be avoid to make calculations simpler. The reconstruction depends on sparse keypoints.
- Semi-dense Direct Method: Consider using only pixels with gradients and discarding the pixels whose gradients are not obvious because such pixels have no positive effect in motion estimation. Finally performing semi-dense reconstruction.
- Dense Direct Method: All pixels are used for motion estimation and reconstruction, which has high requirements for computation. Yet the built dense map has many functions, such as path planning and obstacle avoidance.

The direct method is on account of the assumption that the gray value is constant, i.e. assuming that the imaged gray value of a point is constant at various views. However, when the illumination changes, the direct method is easy to fail. The reason is that the constraint of gray value invariance requires the luminosity error between two images to be as small as possible. In Fig. 3.6 shows the relationship of point P between space and image and the relationship between two frames. The corresponding pixel points of the point P in the space on the first frame and the second frame image are P_1 and P_2, respectively. Beyond that, the relative pose in the following figure is represented by rotation matrix R and translation vector t, it is able to be expressed by the Lie algebra ξ (see in appendices A).

In order to obtain the pose of the camera in the second frame relative to the first frame, an optimization problem is established according to the hypothesis of gray value invariance. As shown in Equ. 3.13, the optimal solution is gained by minimizing the photometric error which is represented by the variable e [4].

$$e = I_1(P_1) - I_2(P_2) \tag{3.13}$$

The optimization function exhibited in Equ. 3.14 is the L_2 norm of the photometric error.

$$\min_{\xi} J(\xi) = ||e||^2 \tag{3.14}$$

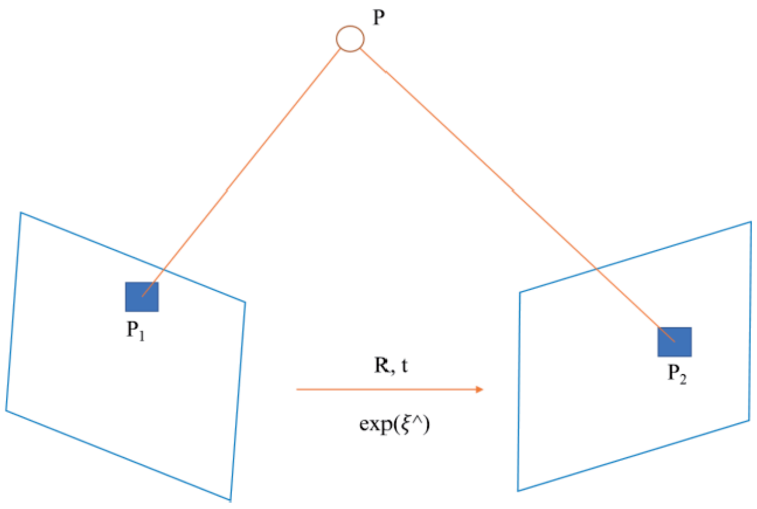


Figure 3.6.: Schematic diagram for direct method [4]

There are endless points in reality, if each point is represented as P_i, then the whole camera pose estimation problem becomes the Equ. 3.15:

$$\min_{\xi} J(\xi) = \sum_{i=1}^{N} e_i^T e_i \tag{3.15}$$

where e_i is expressed in Equ. 3.16.

$$e_i = I_1(P_{1,i}) - I_2(P_{2,i}) \tag{3.16}$$

The specific solution to the above formulas will not be outlined here.

3.3.2. Feature-based method

The feature-based method considers that some representative points should be picked first which called feature points. Thereafter, the motion of the camera is estimated only for these feature points while reckoning the 3D position of the feature points. However, information about other non-feature points in the image is discarded. For each feature point, in order to explain its difference from other points the ”descriptor” is created. A descriptor is usually a vector containing information about feature points and surrounding areas. If the descriptors of two feature points are similar, they can be considered to be the same point. On the basis of the information of the feature points and the descriptors, the matching points in the two images could be calculated. Once the matched feature points are found, the relative pose is able to be calculated by the epipolar geometry. After the feature points are reconstructed by relative pose, the reprojection errors of the feature points are minimized to obtain the best solution for the pose.

The following flow chart 3.7 shows the entire process of feature-based VO including feature detection, feature matching, motion estimation, and triangulation. They are an indispensable part of VO and have been gradually improved by predecessors [18–20]. Thence, the next few sections will elaborate on the specific implementation steps of the main components of feature-based VO.

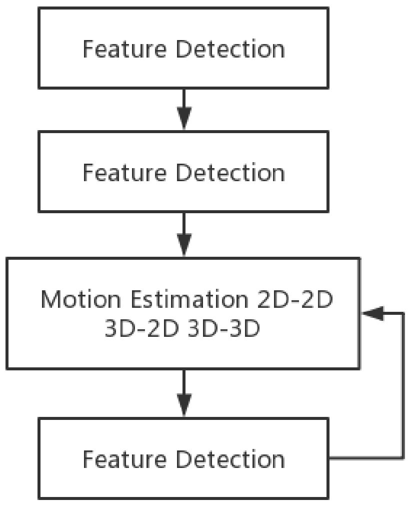


Figure 3.7.: Feature-based VO

3.3.2.1. Feature detection

The feature is the representation of image information, high standards are generally required for selected features. For example, they should have invariance to changes of perspective and illumination and have some flexibility for blur and noise. Relative to the edges and pixel blocks of the image, the corner points with strong local grey value gradients in different directions are easier to calculate and compare their similarities in the two images. For this reason, the corner points become a so-called feature. However, when the observed distance or the perspective changes, the shape or type of the corner point changes so that it cannot be recognized. Thus, scientists in the area of computer vision have studied many more steady local image features, such as SIFT [23], SURF [24], ORB [22], etc. They have different performance in the aspect of rotation, scale invariance, and computational speed.

The feature consists of two parts: keypoint and descriptor. Examples for keypoint are Harris corners [33], Shi-Tomasi corners [34] and FAST corners [35]. Feature descriptor contains BRIEF [36], BRISK [37], SURF, SIFT, ORB, etc. The next paragraphs will deal with the basic steps of SIFT, SURF and ORB and their respective highlights.

The full name of SIFT is the Scale Invariant Feature Transform which presented in 2004 by Canadian professor David G. Lowe [23]. SIFT feature is a very stable local feature because it is scale and rotation invariant. The SIFT algorithm has the following main steps:

- Create different resolutions of image: That is the construction of Difference-of-Gaussian (DoG) Pyramid. A pyramid with a linear relationship (scale space) is constructed so that the feature points of the image can be found on the continuous Gaussian kernel scale. In addition, using a first-order DoG function to approximate a Gaussian Laplacian is equivalent to approximately calculate the most stable feature of an image, while greatly reducing the amount of computation.
- Keypoint localization: First, keypoint is selected by the local extrema of the DoG scale space, and then the low-contrast points and the unstable edge response points are removed to obtain the true keypoint.
- Orientation assignment: A gradient direction histogram is generated on account of the local image gradient direction, and the peak of the gradient direction histogram yields a dominant keypoint direction. All further orientations are computed relative to this dominant direction.
- Keypoint descriptor construction: Based on position, direction, and scale information of the SIFT keypoint a set of vectors is applied to depict the information of the keypoint and its surrounding neighborhood pixels.

SIFT is based on lots of heuristics and free parameters, and it is extensively used in computer vision and photogrammetry. Nonetheless, the shortcoming of SIFT is that there is no global control in local method and no perspective invariance.

On the basis of the SIFT algorithm SURF (Speeded Up Robust Features) [24] mainly improves the defects of the SIFT algorithm, such as slow operation speed and large calculation amount. The SURF process is similar to SIFT, hence only improvements are reflected in the following aspects:

- SURF relies on Hessian matrix to transform images, and the positions of keypoints are detected according to the extremum of the Hessian matrix determinant. Apart from the above step, using box blur filtering to approximate Gaussian blur.
- SURF builds scale pyramids by keeping the image size constant but changing the size of the box filter instead of downsampling.

- SURF uses the response of the first-order Haar wavelet in both x and y directions as the distribution information of the constructed feature vector which improves matching speed.

ORB (Oriented FAST and Rotated BRIEF) [22] is a very good real-time image feature extraction and description algorithm. ORB improves the FAST (Features from Accelerated Segment Test) feature extraction algorithm and uses the extremely fast binary descriptor BRIEF (Binary Robust Independent Elementary Features). FAST detects where the gray value of a local pixel changes significantly. This means if a pixel is obviously different from the surrounding neighborhood, the pixel can be considered a corner point. Within the neighborhood of a keypoint in BRIEF, n pairs of pixels p_i, q_i (i = 1, 2, ..., n) are selected. Then comparing the gray value of each pixel pairs, if $I(p_i) > I(q_i)$, 1 is generated in the binary string, otherwise 0. All pixel pairs are compared to generate a binary string of length n, this binary string is a feature descriptor in BRIEF. The improvements implemented by ORB based on FAST feature extraction algorithm are explained below:

- In order to avoid the excessively concentrated corner points extracted by FAST, non-maximum suppression is applied.
- Aim to extract corner points with high quality, ORB can specify the number n of extracted corner points and then calculate the Harris response value for the extracted corner points. The response values are sorted from large to small, and the first n corner points are selected as the final extracted feature point set.
- Image pyramids are used and corner points on each layer of pyramids are detected to retain scale invariance.
- Intensity Centroid method is employed to maintain the rotation invariance.

3.3.2.2. Feature matching

Feature matching solves data association problems in SLAM, which is to associate the same image parts seen in multiple perspectives. This is accomplished by comparing the distances between descriptors to determine the similarity.

Different distance metrics can be chosen depending on the descriptor. If the type of descriptor is floating point, Euclidean distance [38] can be chosen. For a binary [39] descriptor (such as BRIEF), Hamming distance [40] is more suitable. The Hamming distance between two different binaries refers to the number of different bits of two binary strings.

The simplest and most intuitive method is the Brute-Froce Matcher, which calculates the distance between a feature descriptor and all other feature descriptors, and then sorts the resulting distances to match the nearest one keypoint. This method is simple, but there are a large number of error matches, which requires some strategy to filter out the wrong matches. Some methods for optimizing Brute-Froce Matcher are described below.

- Twice the minimum distance: choosing twice the minimum distance found in all matched point pairs as the judging criteria. If Hamming distance between a matched point pair is greater than the value, it is considered to be a wrong match and filtered out; if it is less than the value, it is considered to be a correct match.
- Cross matching: Cross matching carries out a Brute-Froce Matcher again after Brute-Froce Matcher. If keypoint A is matched to keypoint B by Brute-Froce Matcher, and point B is once again matched to keypoint A by Brute-Froce Matcher, it is considered to be a correct match.
- KNN matching: K-nearest neighbor matching means that picking K points which have the most similarity with the feature point. If K is 2, that is called KNN bidirectional matching method. For each feature point, there will be 2 matches. If the distance ratio of the first match and the second match is large enough, then this is considered to be a correct match [41].

3.3.2.3. Motion estimation

Motion estimation can be performed based on matched feature points. There are three types of situations in motion estimation. The first is the relative pose estimation between 2D images, the second is the projection relationship calculation between the 3D point cloud and the 2D image, and the third is the similarity transformation reckoning between 3D point clouds.

2D-2D motion estimation The 2D-2D correspondence is usually used for the initialization of the visual SLAM system because there is only 2D-2D data association at the beginning. First of all, the pose estimation between 2D images is explained. The geometric constraint between any two images can be represented by Fig. 3.8. The center of the left camera is O_l, the center of the right camera is O_r, and the line connecting center O_l and center O_r is called the baseline. Assuming that the feature points are correctly matched, so the feature point in the left projective plane is p_l and its corresponding feature point in the right projective plane is p_r. Image ray $O_l p_l$ and $O_r p_r$ intersects at point P in 3D space. Point O_l, O_r and P determines a plane called epipolar plane. The intersections of the baseline and the projective planes are respectively e_l and e_r called epipole. The intersection lines of the epipolar plane and the projective planes are called epipolar line.

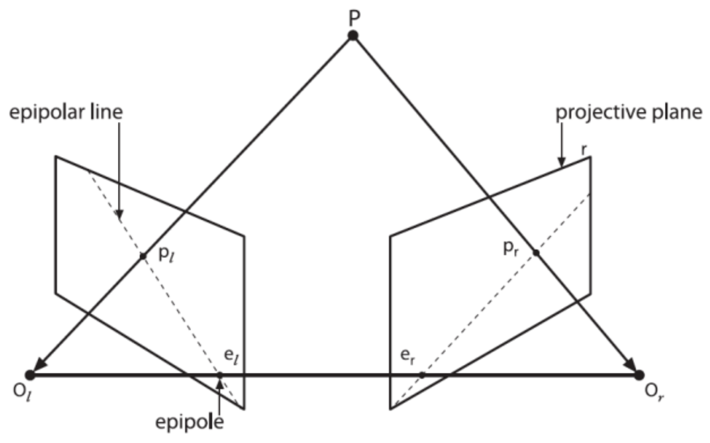


Figure 3.8.: Epipolar geometry constraint [5]

Coplanarity of point O_l, O_r and P is called epipolar constraint which could be represented using fundamental matrix **F** in Equ. 3.17 [2, 42].

$$p_r^T \cdot F \cdot p_l = 0 \tag{3.17}$$

Fundamental matrix **F** is a 3×3 matrix that expresses the correspondence between the feature points of an image pair. The **F** matrix contains the spatial geometric relationship (exterior orientation) of the two images and the camera calibration parameters (interior orientation). Since the rank of the **F** matrix is two and it can be freely scaled, at least seven pairs of feature points are needed to estimate the **F** matrix.

Essential matrix **E** is a 3×3 matrix which differs from the fundamental matrix **F** only by the calibration matrix. K_l and K_r (see in Equ. 3.2) are the calibration matrix of the left and right cameras respectively. The relationship between the essential matrix **E** and the fundamental matrix **F** can be indicated by Equ. 3.18.

$$E = K_r^T \cdot F \cdot K_l \tag{3.18}$$

The **E** matrix has nine unknown parameters. Since the rank of **E** matrix is two, singular value has two constraints and epipolar constraint exists, E consists of five degrees of freedom which means that make use of at least five pairs of feature points to estimate **E** matrix is allowed. In addition to the fundamental matrix and the essential matrix, there is also a matrix called the homography matrix **H**. **H** matrix describes the mapping of plane in global coordinate system and image coordinate system, it is not introduced here.

Solving the relative pose depends on **E** matrix and **F** matrix. The normalized eight-point-algorithm [43] is the easiest way to calculate the fundamental matrix. Therefore, the basic process of solving the fundamental matrix by the eight-point-algorithm is elaborated in detail below, and then the essential matrix through the relationship between the essential matrix and the fundamental matrix is obtained.

To avoid numerical issues, it is necessary to condition the image coordinates within the feature point sets before estimating the fundamental matrix. Firstly determining the center of gravity $(x_{C,l}, y_{C,l})$ and $(x_{C,r}, y_{C,r})$ of the feature points in two images. Secondly calculating the average distances $\bar{S}_{img,l}$ and $\bar{S}_{img,r}$ of all feature points from the centres of gravity in image space. Thirdly obtaining the scales $S_{2D,l} = \sqrt{2}/\bar{S}_{img,l}$ and $S_{2D,r} = \sqrt{2}/\bar{S}_{img,r}$ for image coordinate. Supposing $N \geq 8$ feature point pairs extracted on the left and right images are $x_{l,i}$ and $x_{r,i}$ with $i = 1, 2, 3...N$, thence conditioning can be expressed in Equ. 3.19.

$$x^{'} = T_{2D} \cdot x, \quad T_{2D} = \begin{bmatrix} S_{2D} & 0 & -S_{2D} \cdot x_C \\ 0 & S_{2D} & -S_{2D} \cdot y_C \\ 0 & 0 & 1 \end{bmatrix} \tag{3.19}$$

After that the conditioned feature points in image coordinate are $x^{'}_{l,i}$ and $x^{'}_{r,i}$. For each point pair $x^{'T}_{r,i} \cdot F \cdot x^{'}_{l,i} = 0$ is satisfied. This constraint is able to expressed via the Kronecker product and the vec-Operator: $(x^{'}_{l,i} \otimes x^{'}_{r,i})^T \cdot vec(F) = 0$. The expanded description can be seen in the Equ. 3.20.

$$\begin{pmatrix} (x^{'}_{l,1} \otimes x^{'}_{r,1})^T \\ ... \\ (x^{'}_{l,N} \otimes x^{'}_{r,N})^T \end{pmatrix} \cdot vec(F) = 0 \tag{3.20}$$

The above formula 3.20 can also be written as Equ. 3.21.

$${}_N A_9 \cdot_9 f_1 =_N 0_1 \tag{3.21}$$

If there is a certain (non-zero) solution, the rank of the coefficient matrix A is at most eight. As F is a homogeneous matrix, the solution is unique in the absence of a scale factor and can be directly solved by a linear algorithm under the premise that the rank of matrix A is eight. Matrix A's rank may be greater than eight due to noise in the coordinates of the feature points, thus a least-square solution is required which can be solved by Singular Value Decomposition (SVD) in Equ. 3.22.

$$A = U \cdot S \cdot V^T \tag{3.22}$$

The solution of f is the singular vector corresponding to the smallest singular value of the coefficient matrix A, that is, the last column vector of the matrix V. The rank of the **F** matrix is two because the fundamental matrix has an important characteristic that is the singularity. For the fundamental matrix, nonsingularity means that the calculated epipolar lines don't coincide. Therefore, a singular constraint is added to correct the matrix F' derived from f. First of all, SVD is carried out for F' in Equ. 3.23:

$$F' = \begin{bmatrix} f(1:3) & f(4:6) & f(7:9) \end{bmatrix} = U \cdot S' \cdot V^T \tag{3.23}$$

where $S' = diag(\sigma_1, \sigma_2, \sigma_3)$. Then smallest singular value is set to 0: $S = diag(\sigma_1, \sigma_2, 0)$. The corrected **F** matrix is obtained by product of U, the corrected S and V^T in Equ. 3.24:

$$F = U \cdot S \cdot V^T \tag{3.24}$$

The final **F** matrix is recovered from the conditioning:

$$F = T_{2D,r}^T \cdot F \cdot T_{2D,l} \tag{3.25}$$

For the essential matrix, the normalized point pairs are substituted into the eight-point-algorithm of **F** to get the initial **E** matrix. Then **E** matrix is also subjected to SVD. Finally reconstructing **E** matrix with the singular value constraint through replacing $S' = diag(\sigma_1, \sigma_2, \sigma_3)$ by $S = diag(1/2(\sigma_1 + \sigma_2), 1/2(\sigma_1 + \sigma_2), 0)$.

The next step is to recover the camera's motion based on the estimated essential matrix **E**, i.e. calculating the external orientation (**R** is rotation matrix and **t** is translation vector) of the camera. This process is derived from SVD: $E = U \cdot S \cdot V^T$ with $S = diag(1, 1, 0)$. There are two solutions for **t**: $t_1 = v_3$ and $t_2 = -v_3$ (v_3 refers to the last column of matrix V). And **R** has also two solutions: $R_1 = V \cdot W \cdot U^T$ and $R_1 = V \cdot W^T \cdot U^T$ with $W = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$.

Fig. 3.9 shows the combinations of two rotation matrix **R** and two translation vector **t** to get four possible solutions. Only in solution (a) the 3D point in both cameras has a positive depth. Thus, as long as substituting any point into four solutions and calculating the depth of the point in two cameras, the correct solution could be outcropped.

3D-2D motion estimation The characteristics of 3D-2D are commonly used in the operation phase of the visual SLAM system. The previous camera pose and the points in 3D space are known. It is necessary to estimate the correspondence between these 3D points and 2D feature points in the current frame. With this correspondence, the Perspective-n-Point

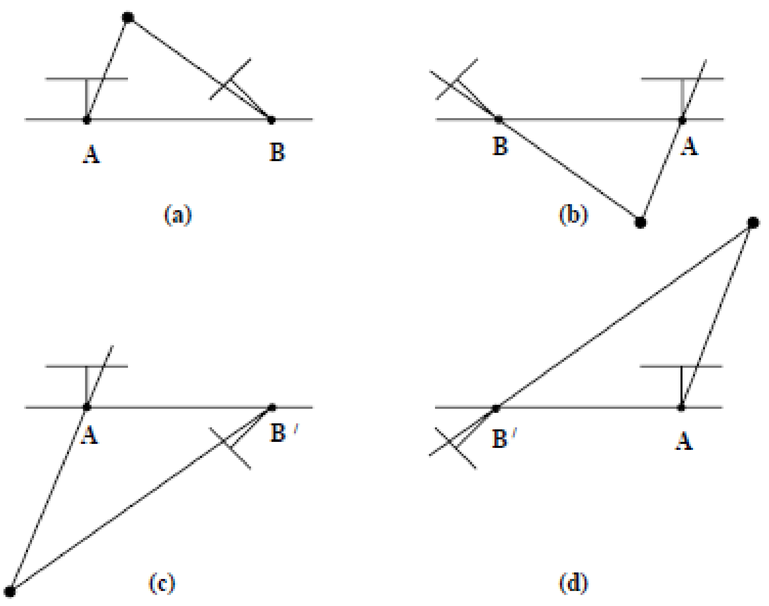


Figure 3.9.: Four possible solutions for relative orientation parameters from **E** [2]

(PnP) method can be used to solve the pose. There are many ways to solve PnP problems, such as P3P [44], Direct Linear Transformation (DLT) [2], EPnP [45] and UPnP [46]. Apart from this, BA is able to resolve the PnP problem. DLT and BA are interpreted next.

First introducing DLT, assuming that a 3D homogeneous point X is projected onto the image plane by the 3×4 projection matrix P with twelve unknown parameters to get the feature point $x = [u, v, 1]^T$ which can be represented in Equ. 3.26:

$$x = \lambda P X, \qquad P = \begin{bmatrix} p_1^T \\ p_2^T \\ p_3^T \end{bmatrix} \tag{3.26}$$

The cross product can be applied to eliminate the unknown λ: $x \times x = 0$ i.e. $x \times PX = 0$. This formula can be described in detail by the following equation:

$$\begin{bmatrix} 0 & -1 & v \\ 1 & 0 & -u \\ -v & u & 0 \end{bmatrix} \begin{bmatrix} p_1^T X \\ p_2^T X \\ p_3^T X \end{bmatrix} = 0 \tag{3.27}$$

Expanding the above formula 3.27 to get

$$up_3^T X - p_1^T X = 0, \quad vp_3^T X - p_2^T X = 0, \quad up_2^T X - vp_1^T X = 0 \tag{3.28}$$

Obviously, the third equation in the above equation can be obtained by the first two equations which means each feature point offers two linear constraints on P. Making an assumption that the number of feature points is N ($N \geq 6$), a linear system of equations could be listed:

$$\begin{bmatrix} -X_1^T & 0 & u_1 X_1^T \\ 0 & -X_1^T & v_1 X_1^T \\ \vdots & \vdots & \vdots \\ -X_N^T & 0 & u_N X_N^T \\ 0 & -X_N^T & v_N X_N^T \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \\ p_3 \end{bmatrix} = 0 \tag{3.29}$$

The Equ. 3.29 can be written as ${}_{2N}A_{12} \cdot_{12} P_1 = 0$. Because of the existence of the error AP does not equal zero, the $||p||$ is also fixed to 1 due to homogeneous coordinates, thus the SVD can be utilized to solve the problem. Implementing SVD for A: $A = U \cdot S \cdot V^T$, P is the rightest column of V.

Local BA wants to minimize the reprojection error between image coordinate of observations and reprojected reconstructed 3D points, i.e. minimizing the sum of squares of a large number of nonlinear functions which can be solved by a nonlinear least-squares algorithm. The common algorithms are Gauss-Newton (GN) algorithm and Levenberg-Marquardt (LM) algorithm [47]. In the following description, x is a state vector containing rotation and translation parameters, h is the increment of x. $f(x)$ is a series of nonlinear equations, i.e. reprojection error equations for 3D points. For small $||h||$ the GN algorithm performs a first-order Taylor expansion on $f(x)$ to obtain a Jacobian matrix J of the derivative of $f(x)$ with respect to x, which can be seen in Equ. 3.30 [48].

$$f(x+h) \simeq l(h) = f(x) + \frac{\partial f(x)}{\partial x} h = f(x) + J(x)h \tag{3.30}$$

Then the function $F(x)$ is defined in the following formula, which depicts the sum of squares of $f(x+h)$:

$$\begin{aligned} F(x+h) \simeq L(h) &= \frac{1}{2} l(h)^T l(h) \\ &= \frac{1}{2} f^T f + h^T J^T f + \frac{1}{2} h^T J^T J h \\ &= F(x) + h^T J^T f + \frac{1}{2} h^T J^T J h \end{aligned} \tag{3.31}$$

where $f = f(x)$ and $J = J(x)$. Then calculating the first derivative of L(h) and making it equals zero: $L^{'}(h) = J^T f + J^T J h = 0$, the GN step h_{gn} which minimizes L(h) can be found in Equ. 3.32.

$$J^T J h_{gn} = -J^T f \tag{3.32}$$

However, this algorithm does not converge when $J^T J$ is a singular matrix or in ill condition. If the size of the step is too large, the local approximation may not be accurate. LM algorithm implements some improvements on the basis of Gauss-Newton algorithm. Next, the basic principle and specific steps for LM algorithm are explained.

LM algorithm adds damping parameter μ to the Equ. 3.32 and the formula can be expressed in Equ. 3.33.

$$(J^T J + \mu I)h_{lm} = g, \quad g = -J^T f \ \ and \ \ \mu \geq 0 \tag{3.33}$$

If the damping parameter μ is small, $J^T J$ dominates which signifies that the quadratic approximation is good and the LM algorithm is closer to the GN algorithm. Otherwise when μ is large, μI occupies a dominant position and h_{lm} appears in the steepest descent direction, which means that the nearby quadratic approximation is bad.

The steps of LM algorithm are then shown in Alg. 1. Firstly the initial value of the parameters should be determined. k is the number of iterations and starts from zero, k_{max} is the maximum number of iterations and is chosen by user. The factor v is initialized to two. The state vector x is first assigned by the initial input value and then updated by the calculated increment h_{lm}. Matrix A and matrix g come from the Equ. 3.33. Matrix Q_{xx} is the covariance matrix of state vector x, which is the inverse of matrix A. The initial value of the damping parameter μ is related to the size of the elements in A, where τ is also selected by user. In stopping criteria, ε_1 and ε_2 are small and positive numbers, which are chosen by the user. During the iterative process, the condition for controlling the update is the gain ratio ϱ, where the numerator is the gain of the actual nonlinear model and the denominator is the gain predicted by the linear model. If ϱ is large, it means that the linear result nears to the real model, μ should be reduced to bring the next LM algorithm step closer to the GN algorithm step. If ϱ is small, then the linear approximation is very poor, μ is needed to be increased. The purpose is to make the next LM algorithm step approaches the steepest descent direction and reduce the step length. At each iteration, judging the value of ϱ and checking if the exit condition is met. If any exit condition is met, outputting the state vector x and exiting the LM algorithm.

3D-3D motion estimation The 3D-3D data correspondence is mainly used to estimate and correct the cumulative error of the loop, and to calculate a similar transformation that

Algorithm 1 Levenberg-Marquardt method [48]

```
$k := 0;\ \ v := 2;\ \ x := x_0$
$A := J(x)^T J(x);\ \ g := J(x)^T f(x)$
$found := (||g||_\infty \leq \varepsilon_1);\ \ u := \tau \cdot \max\{a_{ii}\}$
while $(not\ found)\ and\ (k < k_{max})$ do
  $k := k+1;\ \ Q_{xx} := A^{-1};\ \ Solve\ \ (A + \mu I)h_{lm} = -g$
  if $||h_{lm}|| \leq \varepsilon_2(||x|| + \varepsilon_2)$ then
    $found :=$true
  else
    $x_{new} := x + h_{lm}$
    $\varrho := (F(x) - F(x_{new}))/((L(0) - L(h_{lm}))$
    if $\varrho > 0$ then
      $x := x_{new}$
      $A := J(x)^T J(x);\ \ g := J(x)^T f(x)$
      $found := (||g||_\infty \leq \varepsilon_1)$
      $u := u \cdot \max\left\{\frac{1}{3}, 1 - (2\varrho - 1)^3\right\};\ \ v := 2$
    else
      $u := u \cdot v;\ \ v := 2 \cdot v$
    end if
  end if
end while
```

can align the loop. It is mainly used for RGB-D cameras and stereo cameras, so it is not mentioned here.

3.3.2.4. Triangulation

After gaining relative pose between two images or among multiple images, triangulation can be exploited to acquire the coordinates of feature points in 3D space. For example in two views, assuminting that $P_1 = [p_{1,1}^T, p_{1,2}^T, p_{1,3}^T]^T$ and $P_2 = [p_{2,1}^T, p_{2,2}^T, p_{2,3}^T]^T$ are projection matrix for two images, the desired homogeneous coordinates of 3D point is X and its corresponding feature points are $x_1 = [u_1, v_1, 1]^T$ and $x_2 = [u_2, v_2, 1]^T$. The principle is similar to the DLT (see in 3.3.2.3) which uses cross product to eliminate the unknown scale. Therefrom each feature point affords two linear constraints on X and the system of linear equations is shown in Equ. 3.34 [2].

$$AX = \begin{bmatrix} u_1 p_{1,3}^T - p_{1,1}^T \\ v_1 p_{1,3}^T - p_{1,2}^T \\ u_2 p_{2,3}^T - p_{2,1}^T \\ v_2 p_{2,3}^T - p_{2,2}^T \end{bmatrix} X = 0 \tag{3.34}$$

However, due to the influence of noise, the above equation is always not equal. Therefore, the problem could be solved by the least-squares method (such as SVD). X is the rightest column of V which is decomposed from $A = U \cdot S \cdot V^T$.

3.4. Back end

VO can get the relative pose between two frames. Due to the inevitable error accumulation, the absolute pose is drift that means it becomes increasingly inaccurate as time goes on. Therefore, with the help of the back end a reliable solution is able to be built. There are two main methods for back end optimization. One is based on the optimization of filtering theory, because of the simplicity EKF [49] is the mainstream method in the early stage which relies on the Markov assumption. Markov assumption is the state of this frame that is only related to the previous frame of the current frame and has nothing to do with the previous frames (like VO), thus it is difficult to achieve global optimization. The other one is nonlinear optimization, it takes all the data into consideration and puts them together for optimization. Although it will increase the amount of calculation, the result is much better. The state of the system is introduced first, then filter-based optimization and nonlinear optimization are expounded in the next two sections.

In the process of visual SLAM, the system can be described by motion equation and observation equation which are shown in Equ. 3.35:

$$\begin{cases} x^k = f_{k-1}(x^{k-1}, u^{k-1}) + w^{k-1} \\ l^k = h_k(x^k) + v^k \end{cases} \tag{3.35}$$

where f is the motion equation that infers the current state from the previous state based on the input information and h is the observation equation which gives the observation data generated when the camera sees a 3D point. x presents all unknowns at each time $k = 1, 2, ...$ including cameras' exterior orientation and landmarks' 3D position, u is the input control part, w is the system noise, v is the observation noise and l is the observation.

3.4.1. Filter-based optimization

In a linear Gaussian system, the equations of motion and observation are linear, and the two noise terms obey the Gaussian distribution of zero-mean. The linear system is expressed as follows [50]:

$$\begin{cases} x^k = \Phi^{k-1}x^{k-1} + L^{k-1}u^{k-1} + w^{k-1} \\ l^k = A^k x^k + v^k \\ w^k \sim N(0, Q_{ww}^k) \\ v^k \sim N(0, Q_{ll}^k) \end{cases} \tag{3.36}$$

where x^k or x^{k-1} is the state vector in the current or previous frame. Q_{ww}^k is a covariance matrix of system noise w. Q_{ll}^k is a covariance matrix of observation noise v. These two covariances are generally set in advance based on experience. Φ^{k-1} is the transfer matrix that is calculated based on the linear relationship between the previous state x^{k-1} and the current state x^k. L^{k-1} is the control matrix which can be obtained on the basis of the linear relationship between control part u^{k-1} and states. A^k is the design matrix that is gained from the relationship between observation l^k and current state x^k. For linear Gaussian systems, Bayesian rules can be used to calculate the posterior probability distribution of x which is the derived process of Kalman filter. Kalman filter is an unbiased optimal estimation of the recursive form of linear systems, the derived results are shown as follows:

$$\begin{cases} Q_{xx,-}^k = \Phi^{k-1} Q_{xx,+}^{k-1} {\Phi^{k-1}}^T + Q_{ww}^{k-1} \\ \hat{x}_-^k = \Phi^{k-1}\hat{x}_+^{k-1} + L^{k-1}u^{k-1} \\ K^k = Q_{xx,-}^k {A^k}^T \left(A^k Q_{xx,-}^k {A^k}^T + M^k Q_{ll}^k {M^k}^T\right)^{-1} \\ \hat{x}_+^k = \hat{x}_-^k + K^k(l^k - h^k(\hat{x}_-^k)) \\ Q_{xx,+}^k = (I - K^k A^k) Q_{xx,-}^k \end{cases} \tag{3.37}$$

where $\hat{x}_-^k$ is the predicted state vector and $\hat{x}_+^k$ is the corrected state vector in the current frame, $Q_{xx,-}^k$ and $Q_{xx,+}^k$ are covariance matrices corresponding to $\hat{x}_-^k$ and $\hat{x}_+^k$, respectively. K is the Kalman gain calculated by optimal estimation and used to scale the innovation $l^k - h^k(\hat{x}_-^k)$. When the predicted state is obtained by the motion equation (see the first equation of Equ. 3.36), the predicted state can be corrected according to the scaled innovation.

However, most systems in the real world are not linear, as well as states and noise are not affected by Gaussian distribution. Therefore, it is necessary to linearize the system and approximate the distribution of states and noises with a Gaussian distribution. For EKF, the linearization of the Equ. 3.35 can be performed using Taylor-series with the development point $x^{k-1} = \hat{x}_{+}^{k-1}$:

$$x^k = f_{k-1}(\hat{x}_{+}^{k}, u^{k-1}, w^{k-1}, 0) + \left.\frac{\partial f_{k-1}}{\partial x}\right|_{\hat{x}_{+}^{k-1}} (x^{k-1} - \hat{x}_{+}^{k-1}) + \left.\frac{\partial f_{k-1}}{\partial w}\right|_{\hat{x}_{+}^{k-1}} w^{k-1} \tag{3.38}$$

The nonlinear observation equations h are linearized analogously to f also on the basis of Taylor-series at the development point $x^k = \hat{x}_{-}^{k}$:

$$l^k = h_k(\hat{x}_{-}^{k}, 0) + \left.\frac{\partial h_k}{\partial x}\right|_{\hat{x}_{-}^{k}} (x^k - \hat{x}_{-}^{k}) + \left.\frac{\partial h_k}{\partial v}\right|_{\hat{x}_{-}^{k}} v^k \tag{3.39}$$

In the approximation of the linear system, the noise term and the state term are treated as Gaussian distribution. Thus, as long as their mean and covariance matrices are estimated, the state can be described. After such an approximation, the follow-up work is the same as the Kalman filter. So EKF is a direct extension of Kalman filter. The formula given by EKF is consistent with Kalman filter, and the linearized matrix is used instead of the transfer matrix and the design matrix in the Kalman filter. The calculated result of EKF 1. order for the Gauss-Markov-Model is summarized in following equation:

$$\begin{cases} Q_{xx,-}^{k} = \Phi^{k-1} Q_{xx,+}^{k-1} {\Phi^{k-1}}^{T} + G_{k-1} Q_{ww}^{k-1} {G_{k-1}}^{T} \\ \hat{x}_{-}^{k} = f_{k-1}(\hat{x}_{+}^{k}, u^{k-1}, w^{k-1}, 0) \\ K^k = Q_{xx,-}^{k} {A^k}^T (A^k Q_{xx,-}^{k} {A^k}^T + Q_{ll}^{k})^{-1} \\ \hat{x}_{+}^{k} = \hat{x}_{-}^{k} + K^k (l^k - h^k(\hat{x}_{-}^{k}, 0)) \\ Q_{xx,+}^{k} = (I - K^k A^k) Q_{xx,-}^{k} \end{cases} \tag{3.40}$$

where $\Phi^{k-1} = \left.\frac{\partial f_{k-1}}{\partial x}\right|_{\hat{x}_{+}^{k-1}}$, $G^{k-1} = \left.\frac{\partial f_{k-1}}{\partial w}\right|_{\hat{x}_{+}^{k-1}}$, $A^k = \left.\frac{\partial h_k}{\partial x}\right|_{\hat{x}_{-}^{k}}$, $M^k = \left.\frac{\partial h_k}{\partial v}\right|_{\hat{x}_{-}^{k}}$.

EKF has some limitations. For example, it only linearizes using first-order Taylor-series expansion at a fixed point and then directly calculates posterior probability based on the linearization result. However, the first-order Taylor-series expansion depends on the nonlinearity of the observation and state equations. If these equations have a higher-order term (such as 7 order), the first-order Taylor expansion does not necessarily approximate the whole function. Another drawback is the limited storage space of the EKF. EKF needs to store the mean and variance of the state, at the same time maintain and update them. There are a large amount of landmarks in the visual SLAM. If these landmarks are put into the state, the amount of storage will increase squared with the number of states, so EKF is not suitable for large scenes.

3.4.2. Nonlinear optimization

Local BA is used in VO for nonlinear optimization, but only the camera's pose and landmarks' position between two adjacent frames are optimized. For back end optimization, all poses and positions of all landmarks for the entire map need to be Optimized, which means that the amount of data is greatly increased. A huge amount of data can cause a sharp increase in computation which is not suitable for real-time. Until the last decade, people gradually realized the sparsity of BA in the SLAM problem, so that it can be used in real-time scenarios [51]. Sparsity refers to the additional sparsity of the Hessian matrix H when some camera states are not associated with landmarks in BA. The method for solving BA with the help of sparsity will be explained next.

The observation equation is expressed in Equ. 3.41:

$$h(x):\begin{cases} x' = x_c - f\frac{r_{11}(X-t_x)+r_{21}(Y-t_y)+r_{31}(Z-t_z)}{r_{13}(X-t_x)+r_{23}(Y-t_y)+r_{33}(Z-t_z)} \\ y' = y_c - f\frac{r_{12}(X-t_x)+r_{22}(Y-t_y)+r_{32}(Z-t_z)}{r_{13}(X-t_x)+r_{23}(Y-t_y)+r_{33}(Z-t_z)} \end{cases} \tag{3.41}$$

where camera's rotation matrix is $R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}$ and camera's translation vector is $t = [t_x\ t_y\ t_z]^T$. x_c, y_c and f are interior orientation. $[X\ Y\ Z]^T$ is the position of 3D point in global coordinate system. $[x'\ y']^T$ is the image coordinate of the 3D point projected on the image. $x = [c_1, ..., c_m, p_1, ..., p_n]^T$ stands for camera's exterior orientation c_i (include rotation and translation parameters) and landmark's 3D position p_j. l is the observations, then the error of this observation is $e = l - h(x)$. Therefore, the overall cost function can be expressed as [52]:

$$\frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{n}||e_{ij}||^2 = \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{n}||l - h(x)||^2 \tag{3.42}$$

When adding a small increment Δx to the argument x, the objective function becomes:

$$\frac{1}{2}||f(x+\Delta x)||^2 \approx \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{n}||e_{ij} + J_{c,ij}\Delta c_i + J_{p,ij}\Delta p_j||^2 \tag{3.43}$$

where $J_{c,ij}$ represents the partial derivative of the entire cost function 3.42 to the camera pose in the current state, and $J_{p,ij}$ represents the partial derivative of the function to the location of the landmark. Then organizing camera poses and landmarks' positions separately:

$$x_c = [c_1, c_2, ..., c_m]^T \in \mathbb{R}^{6m}, \quad x_p = [p_1, p_2, ..., p_n]^T \in \mathbb{R}^{3n} \tag{3.44}$$

Consequently, the Equ. 3.43 can be simplified as follows:

$$\frac{1}{2}||f(x+\Delta x)||^2 = \frac{1}{2}||e + J_c\Delta x_c + J_p\Delta x_p||^2 \tag{3.45}$$

where Jacobian matrices J_c and J_p are the derivatives of the overall objective function to the overall variable. Finally, an incremental linear equation like LM and GN should be solved, taking GN as an example:

$$H\Delta x = g \tag{3.46}$$

The Hessian matrix H can be represented by the Jacobian matrix:

$$H = J^T J = \begin{bmatrix} J_c^T J_c & J_c^T J_p \\ J_p^T J_c & J_p^T J_p \end{bmatrix}, \qquad J = [J_c \ J_p] \tag{3.47}$$

The sparse structure of Hessian matrix H caused by Jacobian matrix has been discovered in recent years and can be explicitly represented by graph optimization [53]. For the error function e_{ij}, its constraint depends only on the values of the two nodes i and j, i.e. its corresponding Jacobian matrix has the following form:

$$J_{ij} = \left(0_{2\times6}, ..., \frac{\partial e_{ij}}{\partial c_i}, 0_{2\times6}, ..., 0_{2\times3}, \frac{\partial e_{ij}}{\partial p_j}, ..., 0_{2\times3}\right) \tag{3.48}$$

Afterwards the Hessian matrix $H = \sum_{i,j} J_{i,j}^T J_{i,j}$ obtained by Jacobian matrix can be divided into four blocks:

$$H = \begin{bmatrix} H_{cc} & H_{cp} \\ H_{cp}^T & H_{pp} \end{bmatrix} \tag{3.49}$$

where H_{cc} and H_{pp} are diagonal matrices which have non-zero blocks only at $H_{i,i}$ and $H_{j,j}$. The block structure of the matrix H is the adjacency matrix of the graph, that is to say, in the non-zero matrix block of the H matrix which is not diagonal, there is a side between the variables corresponding to the position.

Subsequently, the Equ.3.46 could be replaced by the Equ. 3.50:

$$\begin{bmatrix} H_{cc} & H_{cp} \\ H_{cp}^T & H_{pp} \end{bmatrix} \begin{bmatrix} \Delta x_c \\ \Delta x_p \end{bmatrix} = \begin{bmatrix} g_c \\ g_p \end{bmatrix} \tag{3.50}$$

Since the diagonal matrix is less difficult to invert, the Schur elimination can be used to eliminate the non-diagonal portion H_{cp} of the upper right corner to obtain the incremental equation about Δx_c:

$$[H_{cc} - H_{cp}H_{pp}^{-1}H_{cp}^{T}]\Delta x_c = g_c - H_{cp}H_{pp}^{-1}g_p \tag{3.51}$$

After getting x_c, x_p can be solved by $x_p = H_{pp}^{-1}(g_p - H_{cp}^{T}\Delta x_c)$.

In this process, only the Schur elimination is introduced to marginalize the landmarks' positions, but the Cholesky decomposition can also be used for marginalization. Of course, the pose variable can also be eliminated. However, the number of landmarks' positions is greater than the number of poses, so it is easier to calculate the pose variables by eliminating the landmarks' positions first.

3.5. Loop closing

The purpose of loop closing is to eliminate the cumulative error caused by relative pose estimation, the approach is that comparing the similarity of two frames of images to establish the constraint between distant matching frames. The popular loop closing method in the existing SLAM system is a method of combining feature points with Bag of Words (BoW) [54]. The main steps of this BoW-based approach are briefly described, including building dictionary, BoW vector, similarity calculation, and loop closing verification:

- Building dictionary: This process of producing a dictionary is equivalent to the clustering process of a descriptor, which can be generated using the K-means algorithm [55]. Since the generated dictionary is too large, it will cost much time on searching matched words one by one, a k-d tree is used to express the dictionary [56].

- BoW vector: Each descriptor in the image searches its corresponding word in the dictionary tree and receives its ID and weight. If two descriptors map to the same word, the weights are added together to get a fixed length vector. After all the descriptors find the corresponding word, a vector is generated about whether the word is present on the image.

- Similarity calculation: There are many ways to calculate similarity, such as the L1 norm of two BoW vectors' difference.

- Loop closing verification: Check whether the matching two frames are close, or there are enough matching points, if a pose is consecutively matched several times with several frames near a pose in history; feature matching is performed on the two frames detected by the loop closing, and then the motion of the camera is estimated, after that the motion is placed in the previous pose map to check whether there is a big difference between the previous estimation and estimation after loop closing.

3.6. Mapping

Mapping is the process of rebuilding a map. A map is a description of the environment, but this description is not fixed and depends on the application of the SLAM. In general, maps can be divided into the metric map and the topological map [57].

The metric map emphasizes the precise representation of the positional relationship of objects in the map, it is usually classified with the sparse map and the dense map. The location of the landmark points discussed earlier can produce sparse map, which is mainly used for positioning in VO and loop closing. Conversely, dense map focus on modeling everything which can be seen, it is primarily for navigation or avoiding barriers. The monocular camera's dense map does not use the matched sparse feature points, but rematches each pixel. It requires the use of epipolar search and block matching techniques [58]. Once the coordinate of each pixel is known in each image, triangulation could be applied to determine the depth. It is worth noting that triangulation needs to be performed multiple times to allow depth to converge. In other words, the depth estimation gradually converges from an uncertain value to a stable value as the measurement increases [59]. The advantage of the metric map is that it is easy to build, save, and plan for short paths. Yet building it requires accurate position estimation. In most cases, many details of the dense map are useless which consumes a lot of storage space.

The topological map underlines the relationship between map elements compared to the accuracy of metric maps. A topology map is a graph consisting of nodes and edges, it considers only the connectivity between nodes. The topological map doesn't require precise location information, has low spatial complexity, and allows for efficient path planning. However, it is difficult to construct a map in a large environment when the sensor information is blurred. In addition, it may create a path that does not meet the best criteria and is difficult to identify the location.

4. Methodologies

This chapter discusses the main method using in the thesis. Firstly the known information of simulated data is introduced. To get close to reality, zero-mean Gaussian noise is added to all simulated data. The maximum variance is added to the simulation data under the precondition that the system is working properly. The position of the 3D dynamic GCPs and the matched feature points between adjacent frames and among cameras are known in this thesis. For 3D dynamic GCPs with the highest accuracy, the variance of Gaussian noise is assumed to be 10^{-5} m. For feature detection error, the presumed variance of Gaussian noise is one pixel. Aim to ensure that the reconstruction of the 3D static points and the camera pose (exterior orientation) have a true scale, the true pose of every camera in the first two frames is given. Since the rotation matrix is more sensitive to the reconstruction of the 3D static tie point than the translation matrix, a smaller error is added to the rotation parameter. Gaussian noise with variances of 10^{-2} m and 10^{-4} rad is added to the rotation and translation parameters respectively. In addition, the camera moves at a speed of 36km/h and takes a picture at a frequency of 20Hz. The calibration matrix **K** of each camera is set the same and it is shown in Equ. 4.1.

$$K = \begin{bmatrix} f & 0 & c_x \\ 0 & f & c_y \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 10000 & 0 & 6911.5 \\ 0 & 10000 & 3839.5 \\ 0 & 0 & 1 \end{bmatrix} [px] \tag{4.1}$$

The transformation matrix $\mathbf{T}_{cam}^{global}$ of the camera in the global coordinate system including the rotation matrix **R** and translation vector **t** is:

$$T_{cam}^{global} = \begin{bmatrix} R & t \\ 0^T & 1 \end{bmatrix} \tag{4.2}$$

In that way the transformation matrix $\mathbf{T}_{global}^{cam}$ from the camera coordinate system to the global coordinate system could be written as:

$$T_{global}^{cam} = T_{cam}^{global-1} = \begin{bmatrix} R^T & -R^T t \\ 0^T & 1 \end{bmatrix} \tag{4.3}$$

This means that the projection matrix **P** from 3D space to image space can be expressed as:

$$P = K(T^{cam}_{global})_{(1:3)} = KR^T[I \mid -t] \tag{4.4}$$

So the projection model used throughout the system is:

$$x \sim PX \tag{4.5}$$

where X is the position of the 3D point and x is the projected image coordinate. This thesis is on the strength of the above model, the main steps are exhibited in Fig. 4.1. The following sections have a detailed introduction to the specific operations of initializing the map, estimating the motion and updating the map.

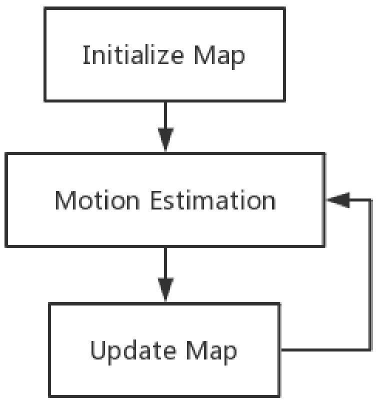


Figure 4.1.: Flow chart of main process

4.1. Initialize map

For all cameras, the poses (exterior orientations) in the global coordinate of the first two frames are known. In order to make the simulation of this thesis more realistic, Gaussian noise is added to all poses before starting all steps. Rotation is represented by a matrix in 3D space, but the operation of adding noise can only be used at the parameter level. First of all, the parameters representing the orientation in the rotation matrix should be extracted. Euler angle, quaternion, and Lie algebra are often employed to express rotation. However, the Euler angle has a very large defect, i.e. gimbal deadlock. Therefore, usually in the application scenario related to rotation, the quaternion is used rather than the Euler angle. Yet among these three methods, Lie algebra is the only one that is possible to represent rotation and translation at the same time. In addition, when solving the nonlinear least-squares optimization used by the pose, the relative pose calculated on the Lie algebra can avoid singular points and ensure that a small transformation matrix can also be shown [60]. Wherefore this thesis utilizes Lie algebra to represent rotation and translation, and performs mathematical calculations. Lie algebra $se(3)$ is exhibited in Equ. 4.6 (see in appendices A):

$$se(3) = \left\{ \exp(\xi) = \exp\begin{pmatrix} w_\times & u \\ 0^T & 0 \end{pmatrix} \in \mathbb{R}^{4\times 4} \middle| w_\times \in so(3), u \in \mathbb{R}^3, \xi = \begin{pmatrix} u \\ w \end{pmatrix} \in \mathbb{R}^6 \right\} \tag{4.6}$$

where u is the translation parameter and w is the rotation parameter, which can be obtained by logarithmic mapping of the transformation matrix owing to $\exp\begin{pmatrix} w_\times & u \\ 0^T & 0 \end{pmatrix} = \begin{bmatrix} R & Vu \\ 0^T & 1 \end{bmatrix}$.

After that, Gaussian noise $N(0, 10^{-4}m^2)$ and $N(0, 10^{-8}rad^2)$ can be added to u and w, separately. Then the noisy Lie algebra $\xi = (u\ w)^T$ is converted into Lie group by exponential mapping, so the required noisy transformation matrix (pose) T_{cam}^{global} is got. The covariance matrix of the pose is shown in Equ. 4.7.

$$Q_{\xi\xi} = \begin{bmatrix} 10^{-4}m^2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 10^{-4}m^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 10^{-4}m^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 10^{-8}rad^2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 10^{-8}rad^2 & 0 \\ 0 & 0 & 0 & 10 & 0 & 10^{-8}rad^2 \end{bmatrix} \tag{4.7}$$

Similarly, there is also an error of the feature detection in reality. Therefore, the variance of the Gaussian noise is set to one pixel and added to the image coordinates, the noisy image coordinates of the feature points are obtained. Then the covariance matrix of the feature point is displayed in Equ. 4.8.

$$Q_{ff} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} px^2 \tag{4.8}$$

The error of the observation is caused by camera pose error and feature detection error, the covariance matrix of the observation is shown in Equ. 4.9:

$$Q_{ll} = AQ_{\xi\xi}A^T + Q_{ff} \tag{4.9}$$

where matrix A is the derivative of the projection equation 4.5 to the exterior parameter ξ of camera. The derivative of the transform equation A.12 to the exterior parameter ξ is known in Equ. A.14, as a result the 2×6 matrix A can be derived by multiplying the derivative of the transformation equation to the exterior parameter ξ and the derivative of the projection equation to the point X_{cam}.

$$A = \frac{\partial x}{\partial \xi} = \frac{\partial x}{\partial X_{cam}} \frac{\partial X_{cam}}{\partial \xi} = \begin{bmatrix} \frac{f}{X_{cam,z}} & 0 & -\frac{fX_{cam,x}}{X^2_{cam,z}} \\ 0 & \frac{f}{X_{cam,z}} & -\frac{fX_{cam,y}}{X^2_{cam,z}} \end{bmatrix} [I \mid -X_{cam\times}] \tag{4.10}$$

In Equ. 4.10 $X_{cam\times}$ represents skew-symmetric matrix of $X_{cam} = R^T[I \mid -t]X$.

When reconstructing a 3D static tie point for each feature point, searching the feature point which exists in at least 2 views. At the same time, the image coordinate and camera pose of this feature point in the appearing view are recorded. If a feature point exists just in one view, the feature point is preserved until it will be observed again. Afterward, triangulation (see in section 3.3.2.4) could be performed using the recorded poses and image coordinates of the feature point, as well as its covariance matrix is calculated. In this thesis, multiple cameras work together to build a map, which means that a 3D static tie point can be reconstructed together by corresponding feature points of different cameras in different frames. It is then updated by the latest observations, thus the position of the 3D static tie point obtained by all information is more accurate. The covariance matrix for the 3D static point is shown in Equ. 4.11 [9]:

$$Q_{XX} = (J^T Q_{ll}^{-1} J)^{-1} \tag{4.11}$$

where 2×3 matrix J is the derivative of the projection equation 4.5 to the position X of 3D static tie point which can be seen in Equ. 4.12.

$$J = \frac{\partial x}{\partial X} = \frac{\partial x}{\partial X_{cam}} \frac{\partial X_{cam}}{\partial X} = \begin{bmatrix} \frac{f}{X_{cam,z}} & 0 & -\frac{fX_{cam,x}}{X^2_{cam,z}} \\ 0 & \frac{f}{X_{cam,z}} & -\frac{fX_{cam,y}}{X^2_{cam,z}} \end{bmatrix} R^T \tag{4.12}$$

Storing the position and covariance matrix of all generated 3D static tie points in the second frame is equivalent to obtaining an initialized sparse map.

4.2. Motion estimation

After acquiring the position of the 3D static tie point from the initialized map, the camera pose can be calculated from the third frame. The projection matrix estimated from the correspondence between 3D static tie point and the 2D image feature points belongs to the PnP problem mentioned above (see section 3.3.2.3). In the above described method of solving PnP, the EPnP with the complexity O(n) uses a linear method to solve the problem. Compared to the DLT that calculates the linear solution, the premise of EPnP is the same as the thesis, i.e. the calibration matrix is known. Moreover, it also considers nonlinear constraints and has low sensitivity to noise, so results are more accurate. Compared to P3P which can process three point pairs, but has no effect on more points. EPnP can handle a large number of points and is effective for both planar (only three) and non-planar ($\geq four$) points. The most important thing is that EPnP does not require an initial value, while other iterative methods are more affected by the initial value. For the above reasons, EPnP is applied herein. In the global coordinate system, the 3D coordinates are signified as a weighted sum of a set of virtual control points by EPnP algorithm. For the ordinary case, it requires four control points and they can't be on the same plane. Then the camera pose could be further solved after calculating the coordinates of the four control points in the camera coordinate system.

The main steps of EPnP is introduced below. For more details, please refer to the paper [45].

- Selecting control points: The first control point selects the centroid position of all 3D points and selects the remaining control points to constitute a foundation that is consistent with the data' main direction, the process is similar to principal component analysis. This method is also similar to the normalized point coordinates of DLT and can improve the stability of EPnP.

- Calculating the weight sums of eigenvectors: Multiplying by the camera coordinates of the virtual control point and the calibration matrix to obtain the image coordinates. By expanding this equation two linear equations of a 3D point can be obtained. Combining the linear equations of all 3D points to gain a $2n \times 12$ linear equation M, the unknown variable x with twelve unknown parameters is the coordinates of the four control points in the camera coordinate system. The solution of x is the null space of M, so x is able to be represented as the sum of the N rightmost column singular vectors that correspond to M's N null singular values.

- Choosing the right linear combination: Determining the coefficients of the rightmost column singular vector in x to get a deterministic solution. N is related to the number of point pairs, control points, camera focal length, and noise, considering $N = 1, 2, 3, 4$, the initial values of the coefficients can be obtained separately.

- GN optimization: Using GN to minimize the difference between control points in the two coordinate systems yields an exact solution of four coefficients. Then, according to the control point, the coordinates of all 3D points in the camera coordinate system are restored. That means, all the 3D-3D matches are known. Thus, by solving the known matched iterative closest points problem, the transformation matrix between the global coordinate system and the camera coordinate system can be gained.

Aim to get a more precise result, nonlinear optimization can be performed after EPnP. Therefore, the transformation matrix in the projection matrix estimated by EPnP is usually used as the initial value of the camera exterior orientation, and then BA is utilized to gain the optimal solution. In this thesis, Random Sample Consensus (RANSAC) [61] is also employed to detect the solution obtained by EPnP. In RANSAC the most amount of inliers obtained by EPnP and their corresponding solution are selected, afterward applying BA to optimize the approximate solution. The LM algorithm (see in section 3.3.2.3) in BA is chosen in this thesis. The reason is that the solution of LM could shun the non-singular and ill-conditioned problems of linear equations' coefficient matrix to some degree, as well as supply more robust and precise increments. Before performing this part of the operation, Gaussian noise is added to the dynamic GCPs. This thesis is based on the higher accuracy of dynamic GCPs than any other parameters, so Gaussian noise $N(0, 10^{-10}m^2)$ is added to the dynamic GCPs.

The main steps of using RANSAC to pick an approximate solution for every camera are described in the Alg. 2. First of all, the initial value is needed to be determined. p_1 indicates that there is a final probability of 0.995 to select a subset without outliers. p_2 represents a probability of 0.7 that contains inliers in each iteration. The threshold is a condition for judging the inlier. If the reprojection error of a point is less than five, it will be regarded as an inlier. n is the number of point pairs that need to be entered in EPnP. Dynamic GCPs are always considered as inliers because of their extremely high accuracy, then they are also inputted into EPnP. The number of dynamic GCPs seen by each camera at each frame is recorded as n_d. If the number of seen dynamic GCPs n_d is greater than or equal to the number of required point pairs n, then all seen dynamic GCPs are input to EPnP and the number of RANSAC iterations is set to one. Otherwise, randomly selecting $n_s = n - n_d$ static tie points for each iteration and determining iteration number according to p_1, p_2 and n_s. Within each iteration, a projection matrix P can be got. According to the projection matrix P, the image coordinates of all 3D points (including all 3D static tie points and 3D dynamic GCPs) projected on the image of the current frame are calculated. Afterward, the L2 norm between image coordinates of all 3D points and of their corresponding feature points could be gained, which is the so-called reprojection error. Hereafter finding all the inliers whose reprojection error is smaller than the threshold, storing the inliers with more quantities and the projection matrix in each iteration. After finishing the iteration, the projection matrix and inliers with most quantities are obtained.

Recalling the LM Alg. 1 mentioned before, τ, k_{max}, ε_1, ε_2 should be chosen by user. In this thesis, $\tau = 10^{-3}$, $\varepsilon_1 = 10^{-12}$, $\varepsilon_2 = 10^{-12}$ and $k_{max} = 200$. Try the experiment according to the parameters set in the thesis and found that no more than 50 iterations will trigger

Algorithm 2 RANSAC+EPnP

```
$p_1 := 0.995$;  $p_2 := 0.7$;  $threshold := 5$;  $num_{inliers} := 0$
n :=number of point pairs selected in EPnP
found :=use dynamic GCPs
if found:= true then
    $n_d$ :=the number of seen dynamic GCPs
else
    $n_d := 0$
end if
if $n_d >= n$ then
    $num_{iter} := 1$
else
    $num_{iter} := \frac{\log(1-p_1)}{\log(1-p_2)^{n_s}}$
    $n_s := n - n_d$ the number of chosen static tie points in RANSAC
end if
for $i \in num_{iter}$ do
    random choose $n_s$ static tie points
    use $n_d + n_s$ points in EPnP to get projection matrix P
    e := reprojection error for all points
    countset := sets for $e < threshold$
    if $len(countset) > num_{inliers}$ then
        $num_{inliers}$:=len(countset)
        $P_{best}$ :=P
        inliers :=countset
    end if
end for
```

the conditions for exiting the LM algorithm, so setting the maximum number of iterations to 200 is sufficient. The most important part of the LM algorithm is also the derivation of the projection matrix to the exterior parameters, which has been derived in Equ. 4.10. In addition, when implementing the LM algorithm, the iteration of the exterior parameters is implemented based on Lie algebra, but the performing of the matrix operation makes use of the transformation matrix. Therefore, there is always a mutual conversion between the Lie algebra and the transformation matrix. Finally, the optimal exterior parameters and corresponding covariance matrix $Q_{\xi\xi} = (A^T Q_{ll}^{-1} A)^{-1}$ are obtained for each camera.

4.3. Update map

During the update process $k = 3, 4, 5...$, $k-1$ represents the previous frame and k represents the current frame. After obtaining the new pose ξ^k of each camera in the motion estimation, the position and covariance matrix of the 3D static tie point can be updated. First of all, traversing the 3D static tie points that have been generated, and finding the corresponding feature points in current frame. The covariance matrix Q_{ll}^k of observation is then calculated from the covariance matrix $Q_{\xi\xi}^k$ of the new pose, the equation is given at Equ. 4.9. Because the 3D static tie point is still, the predicted position and covariance matrix in the current frame is equivalent to the corrected position and covariance matrix in the previous frame, they are displayed in Equ. 4.13:

$$X_-^k = X_+^{k-1}, \quad Q_{XX,-}^k = Q_{XX,+}^{k-1} \tag{4.13}$$

The derivative of the projection equation 4.5 to the position X_-^k of 3D static tie point is supplied in Equ. 4.12. Thus, the Kalman gain used to update 3D static points is shown below [9]:

$$K^k = Q_{XX,-}^k J^{kT} (J^k Q_{XX,-}^k J^{kT} + Q_{ll}^k)^{-1} \tag{4.14}$$

Then the position of 3D static tie point could be renewed:

$$X_+^k = X_-^k + K^k(l^k - h(X_-^k)) \tag{4.15}$$

where l^k is observation in current frame and $h(X_-^k)$ is observation equation which comes from Equ. 4.5.

At the same time the covariance matrix of 3D static point is able to updated:

$$Q_{XX,+}^k = (I - K^k J^k) Q_{XX,-}^k \tag{4.16}$$

Afterward, searching matching feature points of 3D static tie points that do not appear on the map. These feature points should exist in the current frame and are previously retained which has only one view. Then using the image coordinates and camera poses of all views where the feature point is located to carry out triangulation (see in section 3.3.2.4), the position and covariance matrix in Equ. 4.11 of the new 3D static tie point could be received.

After completing all the steps, the pose and covariance matrix for each camera at each frame can be got, which means that the trajectory of each camera could be displayed. Simultaneously, the position and covariance matrix of the 3D static tie point are also obtained, and the sparse map based on the 3D static tie point is able to be built. Later the actual situation on the road will be simulated to evaluate the performance of the entire system and the dynamic GCPs.

5. Experiments and Results

Before doing the experiment, the number of point pairs entered in EPnP need to be determined. Since this experiment simulates the actual road conditions, the 3D points seen by the camera in the actual situation are often not in one plane. Therefore, this experiment simulates the seen points which are non-planar ($\geq four$). In the trial run, it is found that when the number of point pairs is just four, the program could not run normally. In addition, the paper of EPnP also chose at least five point pairs. Because up to six dynamic GCPs are set in thesis experiment, the number of point pairs that need to input EPnP is six. That means, $six - n_d$ static points are selected each time in RANSAC.

Then some basic settings of the experiment are introduced, a total of three scenarios are set up in this experiment. Scenario A comes from the face-to-face driving of the vehicle, which contains one camera and three dynamic GCPs. Scenario B originates from one-way driving of the vehicle, including three cameras and three dynamic GCPs. Scenario C is the vehicle driving at the junction of three roads, three cameras and six dynamic GCPs are set. In the above scenarios, the trajectories of the camera and dynamic GCPs are fixed manually. There are 400 static tie points in the experiment. These static tie points are generated by a random number generator, and the random number seed is fixed at 400, so the generated random number sequence generated each time is the same. This method ensures that the experimental environment is identical for each scenario, and the distribution of static tie points is shown in Fig. 5.1. The area of the entire simulated field is just $60 \times 60 \times 5$ m, so all simulations only run a limited and less time.

Due to the randomness of results caused by RANSAC, the simulation of each scenario runs 10 times to get average value, which ensures that the results are more convincing and not affected by a single run. After obtaining the average pose of each camera in each frame, it is necessary to evaluate the estimated pose. Since the true camera pose of each frame is known, it is treated as the reference, the difference between the estimated average pose and the reference value can be compared. Here, the estimated average pose and reference are converted to Lie algebras, and then the L2 norm (Euclidean norm) between the translation and rotation parameters and the reference value are calculated separately. The calculation formula for translation parameters is given in the following Equ. 5.1:

$$difference_{translation} = \sqrt{(u_{e,1} - u_{r,1})^2 + (u_{e,2} - u_{r,2})^2 + (u_{e,3} - u_{r,3})^2} \tag{5.1}$$

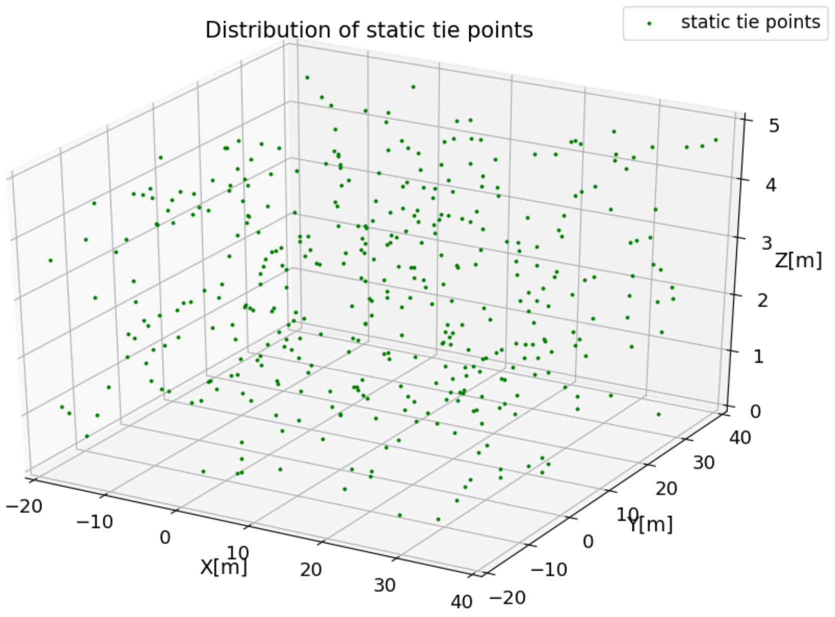


Figure 5.1.: Distribution of static tie points

where u_e represents the estimated average translation parameter and u_r is translation part from reference. The numerical subscript represents three parameters related to corresponding coordinate axes. The rotation formula is the same as the translation formula, please see Equ. 5.2. w_e is the estimated average rotation parameter and w_r is rotation part from reference.

$$difference_{rotation} = \sqrt{(w_{e,1} - w_{r,1})^2 + (w_{e,2} - w_{r,2})^2 + (w_{e,3} - w_{r,3})^2} \tag{5.2}$$

The prediction is that using dynamic GCPs can effectively suppress the increase of error, it will also reduce the overall error, so the root mean square error (RMSE) is used as the evaluation standard too. The calculation formulas for RMSE is written as follows:

$$RMSE_{translation} = \sqrt{\frac{1}{n}\sum_{t=1}^{n}(u_{e,t} - u_{r,t})^2} \tag{5.3}$$

$$RMSE_{rotation} = \sqrt{\frac{1}{n}\sum_{t=1}^{n}(w_{e,t} - w_{r,t})^2} \tag{5.4}$$

where n is the number of frames. In the following sections the three scenarios are described in detail, as well as the experimental results are elaborated, the role of dynamic GCPs is also analyzed and evaluated.

5.1. Scenario A

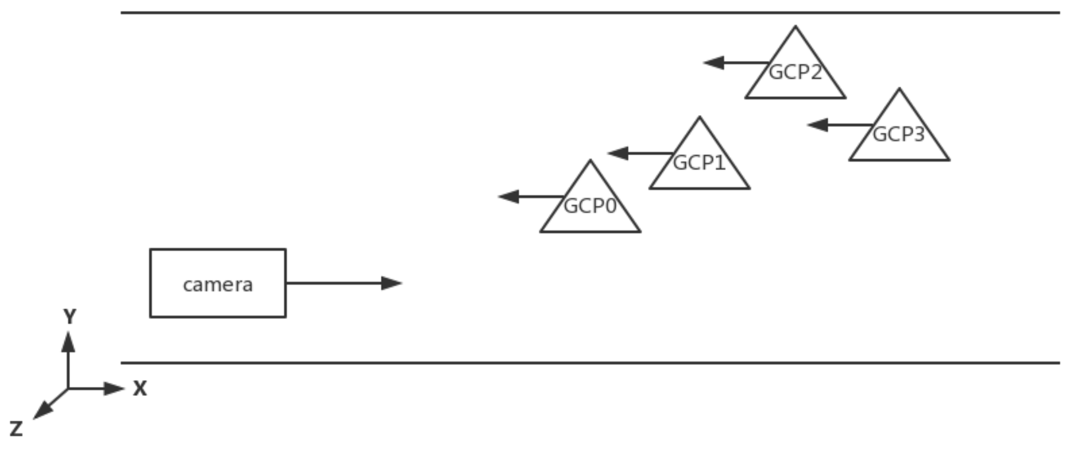


Figure 5.2.: Model of scenario A

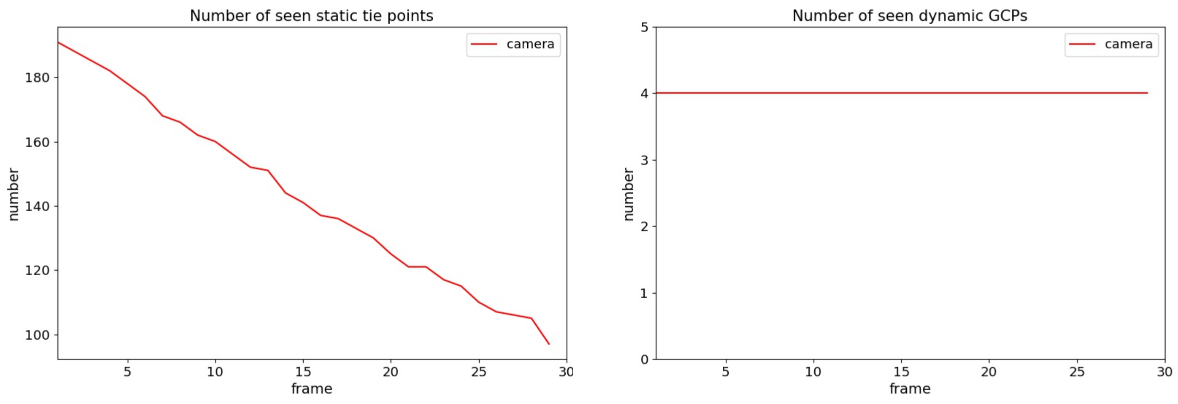
Table 5.1.: The initial positions of camera and dynamic GCPs [m]

	X	Y	Z
camera	-10.0	10.0	2.5
dynamic GCP0	24.0	11.0	4.0
dynamic GCP1	26.0	13.0	1.5
dynamic GCP2	28.0	17.0	3.25
dynamic GCP3	30.0	15.0	2.4

Fig. 5.2 shows the model of the scenario A. On the two-way street, the lower camera goes straight to the right, and four dynamic GCPs above go straight to the left. These four dynamic GCPs are not on the same plane. The initial positions of the camera and four dynamic GCPs are displayed in Tab. 5.1. In X direction the initial distance between the camera and the nearest dynamic GCP is 34 m, and each dynamic GCP differs by 2 m. Eventually

stopped when the camera is 6 m from the nearest dynamic GCP. The camera or dynamic GCPs' entire travel distance is 14 m, the travel distance per frame is 0.5 m.

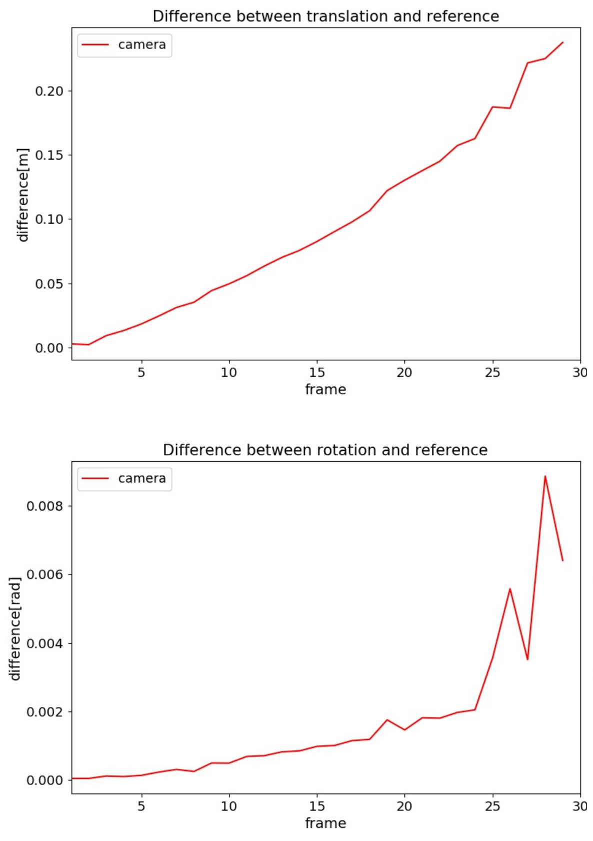
In order to see the effect of dynamic GCPs on camera pose, whether to use dynamic GCPs is tested separately in the same scenario. When dynamic GCPs are not used, the camera locates itself with the static tie points it sees, while updating and generating the static tie points with its new pose. When using dynamic GCPs, dynamic GCPs are always involved in each calculation as the inliers. The number of seen 3D points for the camera in every frame is shown in Fig. 5.3. Fig. 5.3a shows the number of static tie points that the camera can see in each frame of scenario A. Because the scenario has borders, fewer static tie points can be seen as the camera goes straight towards the borders. Aim to understand the impact of dynamic GCPs distribution on camera positioning, the number of dynamic GCPs that the camera can see per frame is fixed to four and is displayed in the Fig. 5.3b.



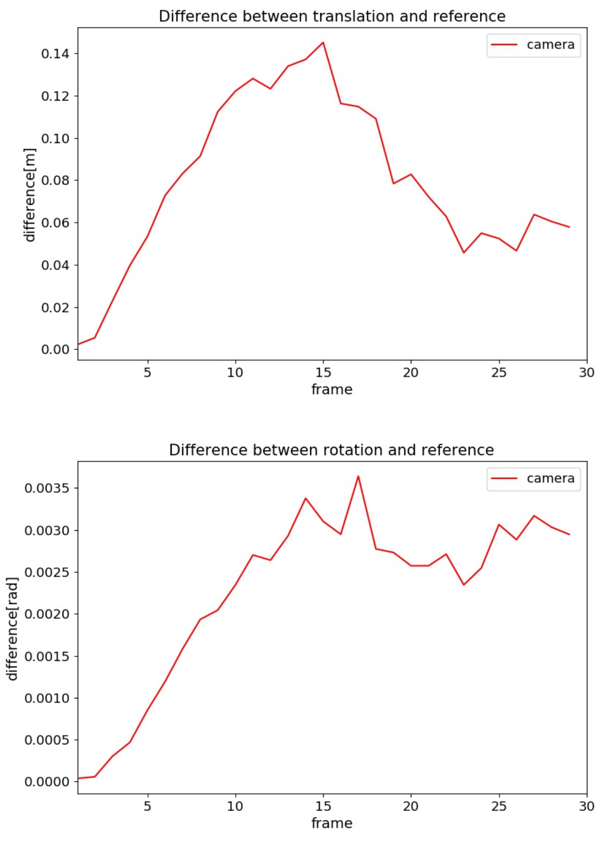
(a) Static tie points **(b)** Dynamic GCPs

Figure 5.3.: The number of seen 3D points

Fig. 5.4 shows the absolute error between the estimated pose and the reference value. Figures in the same column belong to the same test. Fig. 5.4a shows the translation and rotation errors caused by not using dynamic GCPs, and Fig. 5.4b exhibits the translation and rotation errors produced when four dynamic GCPs are seen per frame. For translation errors, the overall error is gradually increasing when GCPs are not applied. Because the camera is getting closer to the far point that is originally generated and greatly affected by noise, now these seen noisy static tie points are used to locate the camera, so the error accumulates more and the error grows faster later. In addition, fewer static tie points over time also mean fewer matching points. Suitable points may not be found to help the camera locate, which will also have a positive effect on the accumulation of errors. The rotation error rises slowly until the 25th frame because the camera goes straight without rotation, so the estimated rotation is accurate. After the 25th frame, the rotation error increases rapidly probably due to the sharply reduced matching points and the corresponding less accurate static tie points.



(a) No dynamic GCPs



(b) Total 3 dynamic GCPs

Figure 5.4.: The difference between estimated pose and reference

The translation error using dynamic GCPs grows very fast before the 15th frame. Since the dynamic GCPs are far away from the camera at the beginning, the four dynamic GCPs seen by the camera are concentrated in a small area and cannot play a larger role. After the 15th frame, the dynamic GCPs are getting closer to the camera, and the four GCPs seen by the camera start to be scattered throughout the image plane. At this time, dynamic GCPs can use its high precision to accurately determine the position of the camera, so the pose error is reduced. For the rotation error, it has a similar trend to the translation error. When the seen dynamic GCP is scattered in the view of the camera and gradually stabilized, the error fluctuates during this period.

Then the RMSE of the translation and rotation parameters are obtained. It can be seen from the Tab. 5.2 and Tab. 5.3 that the RMSE with four dynamic GCPs is smaller than without dynamic GCPs.

Table 5.2.: RMSE in translation [m]

	No dynamic GCPs	Total 4 dynamic GCPs
camera	0.1189	0.0879

Table 5.3.: RMSE in rotation [rad]

	No dynamic GCPs	Total 4 dynamic GCPs
camera	0.0026	0.0025

Compared to the absence of dynamic GCPs, translation and rotation errors grow slowly and overall RMSE is reduced when using dynamic GCPs. It can be found that the usage of dynamic GCPs has a positive impact on the positioning of the camera and decentralized dynamic GCPs can effectively control the growth of errors.

5.2. Scenario B

It can be seen from scenario A that when the dynamic GCPs are closer to the camera, the dynamic GCPs are more conducive to camera positioning. Thus in scenario B, let dynamic GCPs and cameras drive in the same direction to make dynamic GCPs play a greater role. The model of scenario B is displayed in Fig. 5.5. Three cameras and three dynamic GCPs go straight on a one-way street, three cameras have the same height. The initial positions of three cameras and four dynamic GCPs are listed in Tab. 5.4. In Y direction each camera is 4 m apart and a dynamic GCP is distributed 2 m in front of each camera. The overall travel distance of the camera or dynamic GCPs is 14 m, and the travel distance of each frame is 0.5 m.

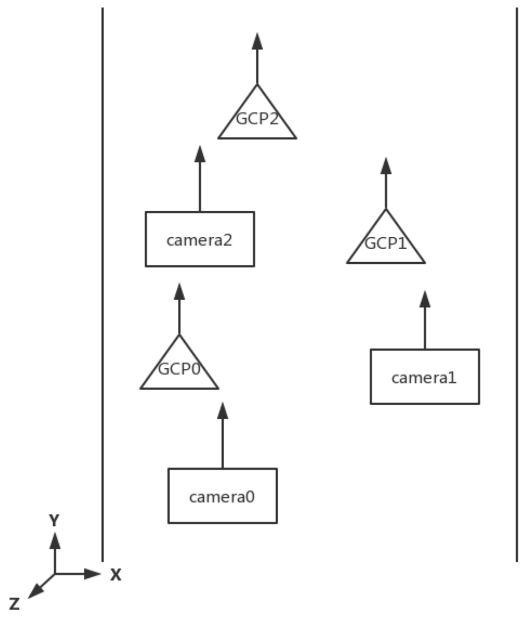
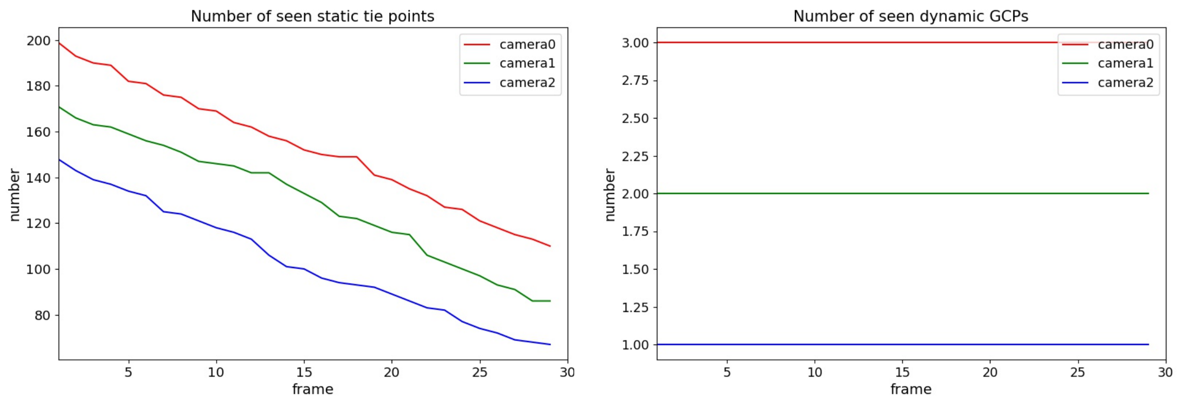


Figure 5.5.: Model of scenario B

Table 5.4.: The initial positions of cameras and dynamic GCPs [m]

	X	Y	Z
camera0	7.6	-10.0	2.5
camera1	11.2	-6.0	2.5
camera2	6.8	-2.0	2.5
dynamic GCP0	7.0	-8.0	3.25
dynamic GCP1	11.0	-4.0	2.75
dynamic GCP2	7.6	0.0	2.4

For scenario B, whether to use dynamic GCP is also tested separately. Like scenario A, the cameras in scenario B go straight, but not rotate. Due to the influence of the boundaries, the number of static tie points observed in each frame decreases over time, as shown in Fig. 5.6a. Aim to compare the impact of the quantity of dynamic GCPs used on the positioning of each camera in scenario B, let each camera see a different number of dynamic GCPs as shown in Fig. 5.6b. The bottom camera0 can always see three dynamic GCPs, the middle camera1 can see two dynamic GCPs all the time and the top camera2 can only see one dynamic GCP.

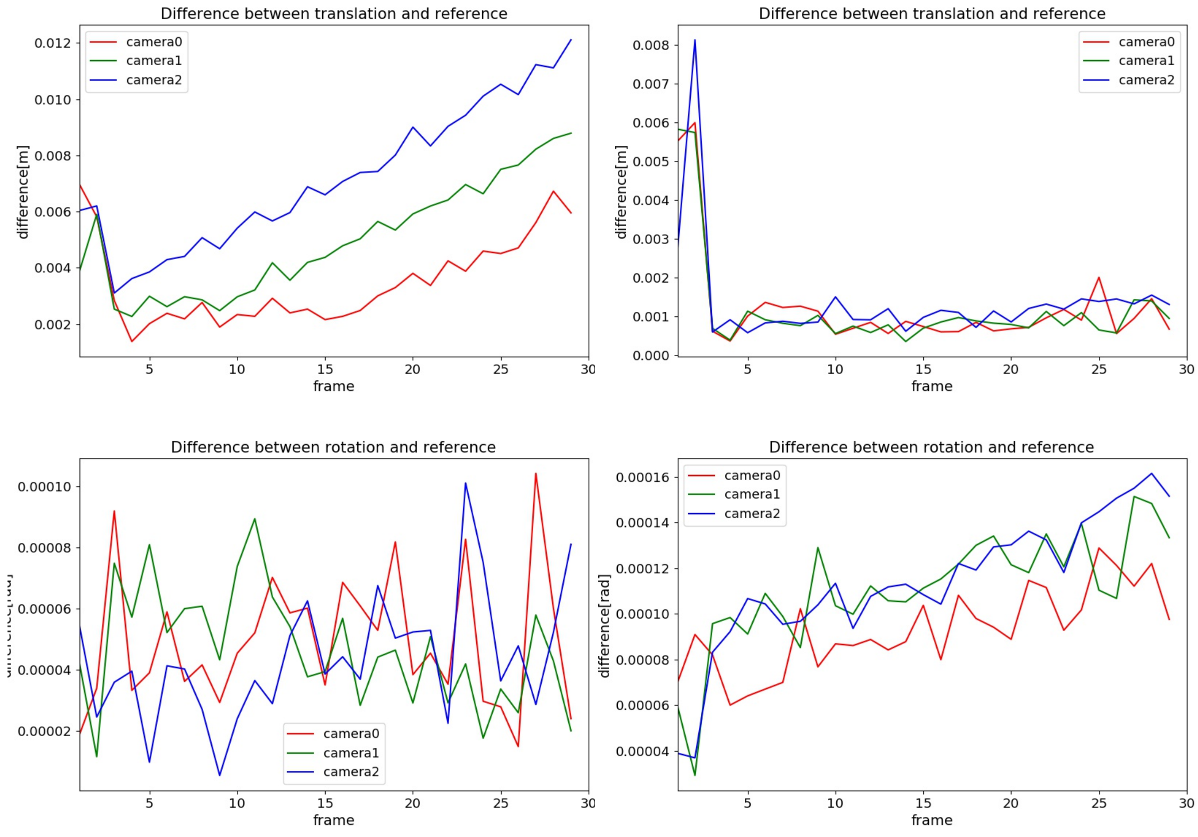


(a) Static tie points **(b)** Dynamic GCPs

Figure 5.6.: The number of seen dynamic GCPs

Fig. 5.7 shows the translation and rotation errors for whether to use dynamic GCPs. When dynamic GCPs are not used, firstly the translation errors of the three cameras decrease in third and fourth frames. Because a large number of map points produced by the initial collaboration can accurately locate the camera and reduce the initial error. Then translation errors grow at the same speed, but the more rear the camera is, the camera translation error is smaller. The reason is that when collaborative mapping, the rear camera0 can see more static tie points, which can choose more precise map points to determine its position. Later, the number of static tie points that each camera can observe is getting less, so each camera may not be able to choose a particularly accurate map points to calculate its location. As a result, the translation error of each camera begins to grow rapidly. Since each camera has only linear motion and no rotation, the rotation error of each camera is only fluctuating and is extremely small.

For the usage of dynamic GCPs, the shortcomings of static tie points are reduced, i.e. the decrease in the number of static tie points seen later will roughly not negatively affect the positioning of the camera. So that the translation error of each camera becomes very small and there is basically not related to the number of dynamic GCPs seen by each camera. For rotational errors, since the dynamic GCPs remain relatively stationary with the camera, they may not have a positive impact on the rotation parameters when going straight. At the same time, it may still be affected by the difference in the number of seen dynamic GCPs, resulting in a difference in the magnitude of the rotation error of each camera.



(a) No dynamic GCPs **(b)** Total 3 dynamic GCPs

Figure 5.7.: The difference between estimated pose and reference

The results in the Tab. 5.5 and Tab. 5.6 are similar to the error trends shown in the above Fig. 5.7. The RMSE of the translation when using the dynamic GCPs is smaller than the case of the unused dynamic GCPs, whereas the RMSE of the rotation with using the dynamic GCPs is greater than the case of the without dynamic GCPs.

Table 5.5.: RMSE in translation [m]

	No dynamic GCPs	3 dynamic GCPs
camera0	0.003805	0.001773
camera1	0.005366	0.001738
camera2	0.007608	0.001914

Table 5.6.: RMSE in rotation [rad]

	No dynamic GCPs	3 dynamic GCPs
camera0	0.000054	0.000094
camera1	0.000051	0.000114
camera2	0.000048	0.000117

When multiple cameras are driving in one direction, the use of dynamic GCPs can reduce translation errors and RMSE, but the number of dynamic GCPs will not play a significant role in it. The rotation errors and RMSE will increase slightly, and the error may be smaller due to the larger number of dynamic GCPs. Compared with scenario A, the RMSE of scenario B is much smaller. On the one hand, dynamic GCPs are closer to the camera, they are more scattered on the image plane, so they can play a good role. Another more important aspect is that scenario B uses multiple cameras to map the same area, thus the generated static tie points are more accurate and the positioning of the camera is more precise.

5.3. Scenario C

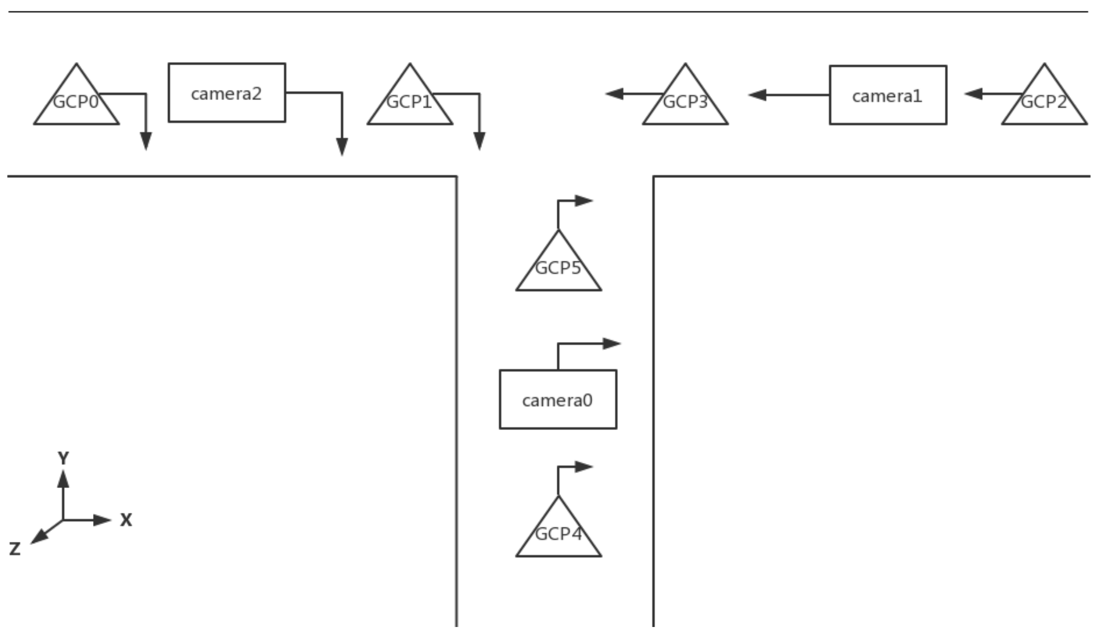


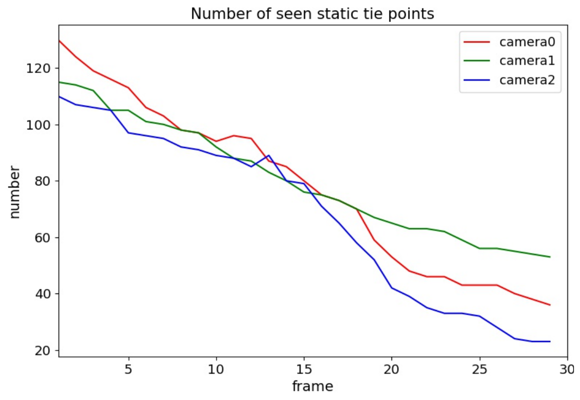
Figure 5.8.: Model of scenario C

The model of scenario C in Fig. 5.8 simulates the situation at the junction of three roads. There are a total of three cameras and six dynamic GCPs, the three cameras have the same height, but the six dynamic GCPs are not coplanar. The initial positions of every camera and every dynamic GCP are shown in Tab. 5.7. A dynamic GCP is placed at 2 m front and rear of each camera. One dynamic GCP is put in front of each camera is to ensure that each camera can see at least one dynamic GCP. All points in Fig. 5.8 follow the direction of the arrow. The camera0 at the bottom goes straight up and then turns right. The camera1 on the right moves all the way to the left. The camera2 on the left advances to the right and then turn right. The total travel distance of each point is 14 m, and the travel distance of each frame is 0.5 m.

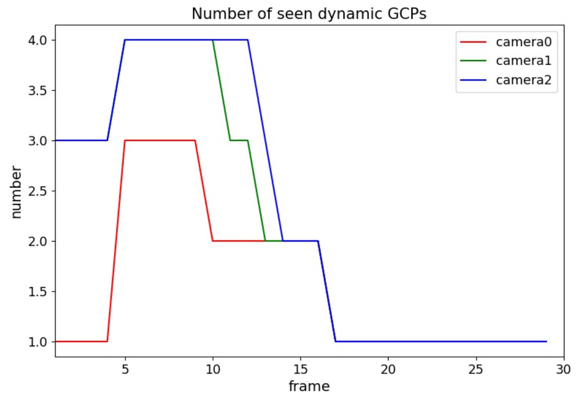
Table 5.7.: The initial positions of cameras and dynamic GCPs [m]

	X	Y	Z
camera0	10.0	2.0	2.5
camera1	18.0	10.0	2.5
camera2	2.0	10.0	2.5
dynamic GCP0	0.0	10.0	1.5
dynamic GCP1	4.0	10.0	2.5
dynamic GCP2	20.0	10.0	2.25
dynamic GCP3	16.0	10.0	1.75
dynamic GCP4	10.0	0.0	2.75
dynamic GCP5	10.0	4.0	2.0

In scenario C, camera0 and camera2 both go straight and rotate 90 degrees, while camera1 only goes straight. The number of static tie points they observe is shown in Fig. 5.9a. Similarly owing to the existence of the boundaries, the number of observed static points will decrease over time. One benefit of scenario C is that because of the different motion trajectories of each camera, it can see different numbers of dynamic GCPs at each moment. In this way, it is possible to compare the effects of changes in the number of observed dynamic GCPs on camera positioning. The number of observed dynamic GCPs by each camera per frame is exhibited in Fig. 5.9b.



(a) Static tie points

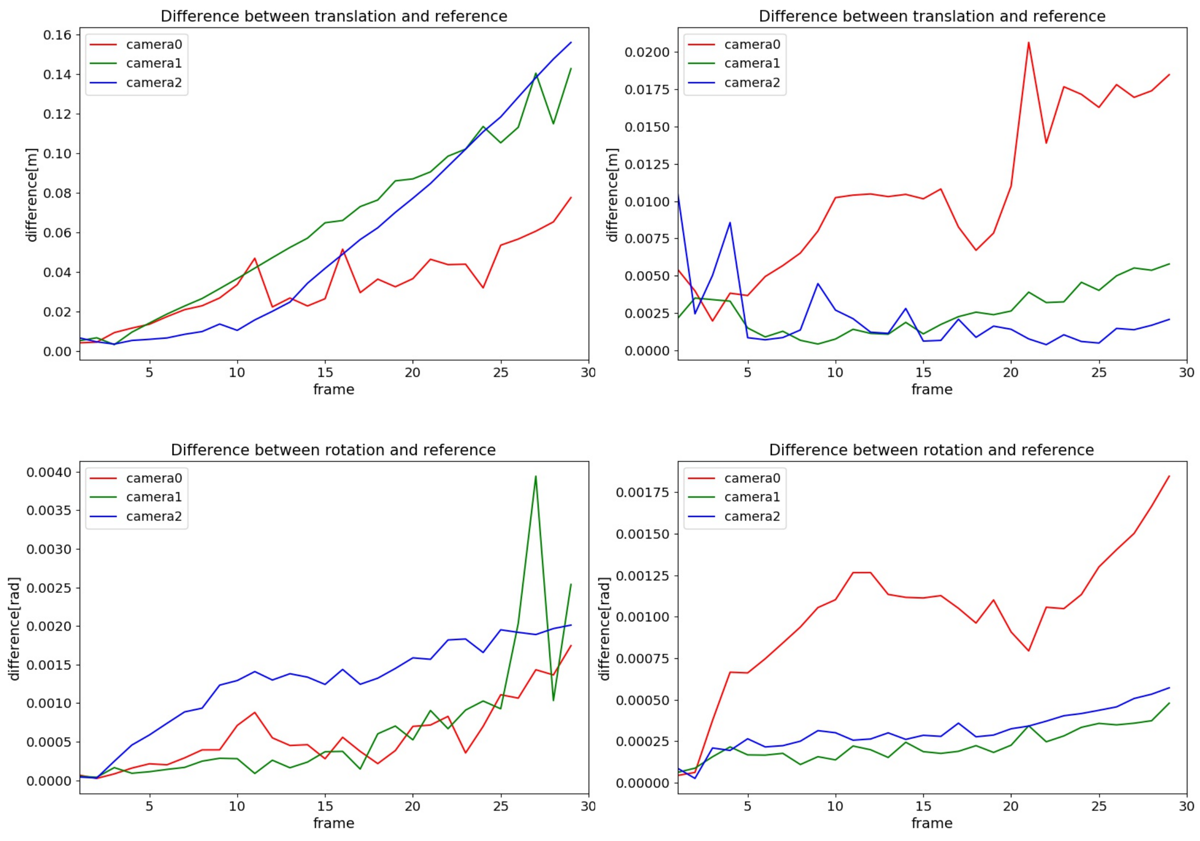


(b) Dynamic GCPs

Figure 5.9.: The number of seen dynamic GCPs

Test implementation is still divided into whether to use dynamic GCPs. Fig. 5.10 depicts the translation and rotation errors for both tests. For the case where dynamic GCP is not used, the errors of translation and rotation are basically in an increasing state. The error of camera1 and camera2 is the fastest growing, while the error of camera2 is slowly increasing. Camera1 has been in a straight line state, the static tie point it sees has been decreasing and the static tie points which camera1 needs to locate are only generated and updated by camera0 and camera2 at the beginning, after that the static tie points are updated by the new pose obtained by itself. The static tie points that are updated by camera1 has lower accuracy than by collaborative mapping of other cameras. Moreover, these points are also the points far from the camera1 which have larger noise, so the pose error of the camera1 that loses the collaborative mapping becomes larger. The case of camera2 is similar to camera1. When it turns, most of the static tie points it sees are not on the map. Thus the static tie points used to locate camera2 can only be generated and updated by itself, and its pose error grows very quickly. Camera0 can achieve small pose errors through collaborative mapping like the other two cameras in the beginning. When camera0 turns to the right, most of the static tie points with higher accuracy it sees are generated and updated by the collaborative mapping of previous camera1 and camera2, as a result the pose error of camera2 has the slowest growth rate.

The above situation has been changed for the case of using dynamic GCPs. Since camera1 and camera2 see more dynamic GCPs than camera2 in the approximately first 14 frames, the error growth of camera1 and camera2 is not only suppressed but also has a small fluctuation. After that, the number of seen dynamic GCPs decreases, thence the errors of camera1 and camera2 slowly increase. Camera0 sees fewer dynamic GCPs than the other two cameras from the beginning, so its pose error increased significantly faster than the other two cameras. In the middle, camera0 is also positively affected by dynamic GCPs, thus the error slowly increased by about six frames. Afterward, the number of seen dynamic GCPs reduces, therefore the pose error begins to increase rapidly.



(a) No dynamic GCPs **(b)** Total 6 dynamic GCPs

Figure 5.10.: The difference between estimated pose and reference

Table 5.8.: RMSE in translation [m]

	No dynamic GCPs	Total 6 dynamic GCPs
camera0	0.0384	0.0118
camera1	0.0761	0.0031
camera2	0.0743	0.0032

Table 5.9.: RMSE in rotation [rad]

	No dynamic GCPs	Total 6 dynamic GCPs
camera0	0.0007	0.0011
camera1	0.0011	0.0002
camera2	0.0014	0.0003

Tab. 5.8 and Tab. 5.9 give the RMSE for each camera about translation and rotation. It can be seen that in addition to the RMSE of camera0 about rotation, the RMSE of the other cameras with respect to translation and rotation in the condition of using a total of six dynamic GCPs is less than the unused case. Probably because the rotation error of camera0 grows too fast at an early stage, the inhibition of error growth by dynamic GCP is limited. Another obvious finding is that without dynamic GCPs the RMSE of camera0 is smaller than the other two cameras, but with six dynamic GCPs it is larger than the other two cameras. This finding also matches the situation in the above Fig. 5.10.

It can be found that camera0 with the slowest error growth and the smallest RMSE becomes the one with the fastest error growth and the largest RMSE among all cameras because it sees the least number of dynamic GCPs. Therefore, the number of seen dynamic GCPs can change the trend of camera error growth. Dynamic GCPs are beneficial to assist camera positioning and control error growth. Compared with scenario A, the dynamic GCPs in scenario C are close to the camera and collaborative mapping is applied, camera positioning in scenario C performs better than scene A. Compared with scenario B, although pose errors in scenario C are bigger than scenario B, scenario C can better reflect the positive role of the number of dynamic GCPs in camera positioning.

6. Conclusion and Outlook

In the experiment, a total of 3 scenarios are planned to test the impact of dynamic GCPs on camera positioning. The first scenario is about a two-way street, a camera and four non-planar dynamic GCPs move face to face. As they approach the increase in camera errors is controlled after the 15th frame due to the increasing dispersion of dynamic GCPs on the image plane as shown in Fig. 5.4, which is in line with expectations. The second scenario regards to vehicles driving on a one-way street, each of the three cameras has a dynamic GCP in front of it. The rear camera can see more dynamic GCPs. It is found in Fig. 5.7 that dynamic GCPs can greatly reduce the translation error, but the magnitude of the translation error is similar for different cameras, which means it is almost not affected by the number of seen dynamic GCPs. Another discovery is that the rotation error with dynamic GCPs is greater than without dynamic GCPs. This is not in line with expectations, maybe dynamic GCPs and cameras remain relatively stationary which has no positive impact on rotation. The further experiment can make dynamic GCPs walk randomly in front of the camera to see if it can reduce the rotation error. The third scenario simulates the situation at the junction of three roads, three cameras come from different directions and each of them has a dynamic GCP in front and back. The number of dynamic GCPs seen by each camera is changing, and a camera that can see a larger number of dynamic GCPs can more effectively reduce its own pose error and suppress the growth of error as displayed in Fig. 5.10. This phenomenon fits expectations. In these three scenarios, scenario A can conclude that decentralized dynamic GCPs are effective in suppressing error growth. Scenario B has the smallest error due to cooperative mapping and close-range dynamic GCPs. Scenario C reflects the impact of the seen changing number of dynamic GCPs on camera positioning, more seen dynamic GCPs have a more positive influence on reducing error.

However, this ideal result only exists in the simulated experiment, because the Gaussian noise added to the known parameters in this experiment is relatively small, and the final error does not increase so much. Yet the actual situation is much more complicated. For example, when dealing with real data, the actual noise may be much larger, and there is also a mismatch that not added to the simulated data. In addition, when the number of required dynamic GCPs increases, the cost of communication increases. How many dynamic GCPs are needed to optimize the results is also a problem. Thus there is so much future work that needs to be implemented to make the results more robust and realistic.

Due to the usage of simulated data, feature detection and feature matching are not included in this thesis. In practice, this is a part which has huge impact on pose accuracy and time consuming. Therefore, in the actual operation process, this part needs to be further understood and optimized. In addition to this, there is no back end optimization. Thus further work can include back end optimization to reduce pose error more effectively. At the same time, if the camera repeatedly appears in a certain place, the loop closing has a great positive effect on the decrease of the pose error. In this thesis, the first two frames of the camera pose are known to reconstruct the 3D static tie point, the scale of the initial map is constructed, and the PnP problem is used to solve pose. In fact, the initialization of the monocular camera is done by panning the camera, and then obtaining a fixed scale about pose and map. Aim to find the true scale, the step size of the vehicle in each frame can be obtained by means of other sensor data, such as data from an inertial measurement unit.

For information exchange, the actual exchange of information needs to consider the cost. For example, the information within how much distance should be accepted, it should be judged which information is useful, how long should the information be retained to ensure the real-time and validity of this information without occupying a large amount of storage space. In addition, this thesis uses a vehicle with a determined location as a dynamic GCP. In fact, the vehicle is not a point, but an object. The vehicle seen in the camera is also composed of a lot of discrete points, then some meaningful points from these points can be used as the dynamic GCPs. The question about how many dynamic GCPs are needed to optimize the results can be turned into another question, i.e. how many vehicles can be seen in the camera, and how many meaningful points should be extracted from the vehicle. Another problem is that this thesis is based on the assumption that dynamic GCP has higher accuracy, but the vehicle itself does not have such high pose accuracy which comes from sensors. Thus these issues should be discussed in real-world experiments.

This thesis uses dynamic GCPs in CoSLAM based on simulated data and finds that dynamic GCPs help in positioning and suppression of error drift. However, the implementation process still lacks back end optimization, which needs to be improved. Apart from this, more experiments should be set up and more realistic situations ought to be considered to analyze the application of dynamic GCPs.

Appendices

A. Lie group and Lie algebra

Lie group refers to a group with continuous (smooth) properties. The Lie groups mentioned in this thesis is the real matrix groups. They are continuous in real space and closed with respect to multiplication and inversion, that is to say consistent with the multiplication and inversion of the matrix. All the three-dimensional rotation matrix sets constitute a special orthogonal group $SO(3)$ in Equ.A.1 [62, 63]:

$$SO(3) = \left\{ R \in \mathbb{R}^{3\times3} | RR^T = I, \det(R) = 1 \right\} \tag{A.1}$$

For SO(3), each differential rotation axis has one degree of freedom with a total of three degrees of freedom. Lie algebra $so(3)$ is a set of skew-symmetric 3×3 matrices that describe the tangent space around the identity element of the group, i.e. differential rotation around each axis. The derivatives of rotation for each axis is as follows:

$$G_1 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix}, \quad G_2 = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{bmatrix}, \quad G_3 = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{A.2}$$

$so(3)$ can be obtained by a linear combination of the above differential rotations. The three-dimensional vector $w = [w_1, w_2, w_3]^T \in \mathbb{R}^3$ represents the coefficients. Then the linear combination can be written in Equ. A.3.

$$w_1G_1 + w_2G_2 + w_3G_3 = \begin{bmatrix} 0 & -w_3 & w_2 \\ w_3 & 0 & -w_1 \\ -w_2 & w_1 & 0 \end{bmatrix} \in so(3) \tag{A.3}$$

It is obvious that the above formula is also the skew-symmetric matrix of w which is called $w_\times$. The relationship between Lie algebra and Lie group can be represented by exponential mapping and logarithmic mapping. The exponential mapping can transform the Lie algebra into the Lie group, conversely the logarithmic mapping converts the Lie group into the Lie algebra. The exponential map is given below:

$$\exp(w_\times) = I + (\frac{\sin\theta}{\theta})w_\times + (1 - \frac{\cos\theta}{\theta^2})w_\times^2 \tag{A.4}$$

where $\theta = \arccos \frac{tr(R)-1}{2}$ is the radian of the rotation around the axis provided by w. And vector w can be extracted from the off-diagonal elements of $\ln(R) = \frac{\theta}{\sin\theta} \cdot (R - R^T)$. The premise is that θ is so small that the above Rodrigues formula can be successfully implemented by using Taylor expansion.

If a vector $x \in R^3$ becomes a vector $y \in R^3$ through the rotation matrix $R \in SO(3)$, it can be linearly represented by the following equation:

$$y = Rx \tag{A.5}$$

Then the differentiation of y about x is:

$$\frac{\partial y}{\partial x} = R \tag{A.6}$$

The differentiation by the rotation parameter can be expressed as multiplication of the rotation and the exponential of the tangent vector:

$$\begin{aligned}
\frac{\partial y}{\partial R} &= \frac{\partial}{\partial w}\bigg|_{w=0} (\exp(w) \cdot R) \cdot x \\
&= \frac{\partial}{\partial w}\bigg|_{w=0} \exp(w) \cdot (R \cdot x) \\
&= \frac{\partial}{\partial w}\bigg|_{w=0} \exp(w) \cdot y \\
&= (\ G_1 y \mid G_2 y \mid G_3 y\ ) \\
&= -y_\times
\end{aligned} \tag{A.7}$$

All the transformation matrix sets constitute a special Euclidean group SE(3) in Equ.A.8:

$$SE(3) = \left\{ T = \begin{bmatrix} R & t \\ 0^T & 1 \end{bmatrix} \in \mathbb{R}^{4\times4} | R \in SO(3), t \in \mathbb{R}^3 \right\} \tag{A.8}$$

The Lie algebra $se(3)$ corresponding to $SE(3)$ a linear combination of differential translations and rotations, and the corresponding coefficient is $\xi = (u\ w)^T \in \mathbb{R}^6$. The differential translations and rotations are presented in the following equations.

$$G_1 = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad G_2 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad G_3 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$
$$G_4 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad G_5 = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad G_6 = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{A.9}$$

Afterwards, the element of $se(3)$ can be written as the linear combination: $u_1G_1 + u_2G_2 + u_3G_3 + w_1G_4 + w_2G_5 + w_3G_6 \in se(3)$. The exponential mapping form on se(3) is briefly described as follows:

$$\exp(\xi) = \exp \begin{pmatrix} w_\times & u \\ 0^T & 0 \end{pmatrix} = \begin{bmatrix} \exp(w_\times) & Vu \\ 0^T & 1 \end{bmatrix} = \begin{bmatrix} R & Vu \\ 0^T & 1 \end{bmatrix} \tag{A.10}$$

where $R = I + Aw_\times + Bw_\times^2$ and $V = I + Bw_\times + Cw_\times^2$. A, B, C are expressed in Equ. A.11:

$$\begin{aligned} \theta &= \sqrt{w^T w} \\ A &= \tfrac{\sin\theta}{\theta} \\ B &= \tfrac{1-\cos\theta}{\theta^2} \\ C &= \tfrac{1-A}{\theta^2} \end{aligned} \tag{A.11}$$

For the above formula, Taylor expansion can still be used when θ^2 is small.

Similarly assuming a vector $x \in R^3$ is converted to a vector $y \in R^3$ through the transformation matrix $T = \begin{bmatrix} R & t \\ 0^T & 1 \end{bmatrix} \in SE(3)$, it can be written as:

$$y = Tx = [R \mid t]x = Rx + t \tag{A.12}$$

The differentiation by vector x is:

$$\frac{\partial y}{\partial x} = R \tag{A.13}$$

With the help of the derivation of the Equ.A.7, the differentiation by the transformation parameters is:

$$\frac{\partial y}{\partial T} = (\, G_1 y \mid G_2 y \mid G_3 y \mid G_4 y \mid G_5 y \mid G_6 y) = (I \mid -y_\times) \tag{A.14}$$

Bibliography

[1] Cesar Cadena, Luca Carlone, Henry Carrillo, Yasir Latif, Davide Scaramuzza, Jose Neira, Ian Reid, and John Leonard. Simultaneous localization and mapping: Present, future, and the robust-perception age. *IEEE Transactions on Robotics*, 32, 06 2016. vii, 9

[2] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision.* Cambridge University Press, Cambridge, UK; New York, 2003. vii, 10, 11, 21, 24, 27

[3] Junhee Park, Seong-Chan Byun, and Byung-Uk Lee. Lens distortion correction using ideal image coordinates. *IEEE Transactions on Consumer Electronics*, 55, 2009. vii, 13

[4] Zhetao Zhang and Wanggen Wan. Dovo: Mixed visual odometry based on direct method and orb feature. *2018 International Conference on Audio, Language and Image Processing (ICALIP)*, pages 344–348, 2018. vii, 15, 16

[5] Ha Quang Thinh Ngo and Mai-Ha Phan. Design of an open platform for multi-disciplinary approach in project-based learning of an epics class. *Electronics*, 8:200, 02 2019. vii, 21

[6] Hugh Durrant-Whyte, David Rye, and Eduardo Nebot. Localization of autonomous guided vehicles. In Georges Giralt and Gerhard Hirzinger, editors, *Robotics Research.* Springer London, 1996. 1

[7] Jorge Fuentes-Pacheco, Jose Ascencio, and J. Rendon-Mancha. Visual simultaneous localization and mapping: A survey. *Artificial Intelligence Review*, 43(1):55–81, Jan 2015. 1, 5, 6

[8] Lawrence A. Sharrott. Centralized and distributed information systems: Two architecture approaches for the 90s. In Douglas J.V. Albright J.W. Ball M.J., ODesky R.I., editor, *Healthcare Information Management Systems.* Springer, New York, NY, 1996. 1

[9] Danping Zou and Ping Tan. Coslam: Collaborative visual slam in dynamic environments. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 2013. 1, 8, 38, 42

[10] Randall C. Smith and Peter Cheeseman. On the representation and estimation of spatial uncertainty. *The International Journal of Robotics Research*, 5(4):56–68, 1986. 5

[11] J. J. Leonard and H. F. Durrant-Whyte. Simultaneous map building and localization for an autonomous mobile robot. In *Proceedings IROS '91:IEEE/RSJ International Workshop on Intelligent Robots and Systems '91*, pages 1442–1447 vol.3, Nov 1991. 5

[12] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. A probabilistic approach to concurrent mapping and localization for mobile robots. *Machine Learning*, 31(1):29–53, Apr 1998. 5

[13] Marc Pollefeys, Luc Van Gool, Maarten Vergauwen, Frank Verbiest, Kurt Cornelis, Jan Tops, and Reinhard Koch. Visual modeling with a hand-held camera. *International Journal of Computer Vision*, 59(3):207–232, Sep 2004. 5

[14] Bill Triggs, Philip F. McLauchlan, Richard I. Hartley, and Andrew W. Fitzgibbon. Bundle adjustment — a modern synthesis. In Bill Triggs, Andrew Zisserman, and Richard Szeliski, editors, *Vision Algorithms: Theory and Practice*, pages 298–372, Berlin, Heidelberg, 2000. Springer Berlin Heidelberg. 5

[15] Randall Smith, Matthew Self, and Peter Cheeseman. Estimating uncertain spatial relationships in robotics. In Ingemar J. Cox and Gordon T. Wilfong, editors, *Autonomous Robot Vehicles*, pages 167–193, New York, NY, 1990. Springer New York. 5

[16] 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(6):1052–1067, June 2007. 6

[17] Michael Montemerlo, Sebastian Thrun, Daphne Koller, and Ben Wegbreit. Fastslam: A factored solution to the simultaneous localization and mapping problem. *Proceedings of the National Conference on Artificial Intelligence*, 11 2002. 6

[18] David Nistr. Oleg Naroditsky. James Bergen. Visual odometry. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 1:652659, 2004. 6, 17

[19] 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, Nov 2007. 6

[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, Oct 2015. 7, 17

[21] R. Mur-Artal and J. D. Tards. Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE Transactions on Robotics*, 33(5):1255–1262, Oct 2017. 7

[22] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. Orb: An efficient alternative to sift or surf. In *2011 International Conference on Computer Vision*, pages 2564–2571, Nov 2011. 7, 17, 19

[23] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, Nov 200. 7, 17, 18

[24] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In Aleš Leonardis, Horst Bischof, and Axel Pinz, editors, *Computer Vision – ECCV 2006*, pages 404–417, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg. 7, 17, 18

[25] Muhamad Risqi Utama Saputra, Andrew Markham, and Niki Trigoni. Visual slam and structure from motion in dynamic environments: A survey. *ACM Computing Surveys*, 51:1–36, 02 2018. 7

[26] Wei Tan, Haomin Liu, Zilong Dong, Guofeng Zhang, and Hujun Bao. Robust monocular slam in dynamic environments. *2013 IEEE International Symposium on Mixed and Augmented Reality, ISMAR 2013*, pages 209–218, 10 2013. 8

[27] D. C. Brown. Decentering distortion of lenses. *Photogrammetric Engineering and Remote Sensing*, 1966. 12

[28] J. Nico P. de Villiers, F. Wilhelm Leuschner, and Ronelle Geldenhuys. Centi-pixel accurate real-time inverse distortion correction. In *International Symposium on Optomechatronic Technologies*, 2008. 13

[29] Pierre Drap and Julien Lefvre. An exact formula for calculating inverse radial lens distortions. *Sensors*, 16:807, 06 2016. 13

[30] J. Weng, P. Cohen, and M. Herniou. Camera calibration with distortion models and accuracy evaluation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(10):965–980, Oct 1992. 13

[31] Z. Zhang. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(11):1330–1334, Nov 2000. 14

[32] Z. Chen, W. Sheng, G. Yang, Z. Su, and B. Liang. Comparison and analysis of feature method and direct method in visual slam technology for social robots*. In *2018 13th World Congress on Intelligent Control and Automation (WCICA)*, pages 413–417, July 2018. 15

[33] Chris Harris and Mike Stephens. A combined corner and edge detector. In *In Proc. of Fourth Alvey Vision Conference*, pages 147–151, 1988. 18

[34] Jianbo Shi and Tomasi. Good features to track. In *1994 Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 593–600, June 1994. 18

[35] Edward Rosten and Tom Drummond. Machine learning for high-speed corner detection. In Aleš Leonardis, Horst Bischof, and Axel Pinz, editors, *Computer Vision – ECCV 2006*, pages 430–443, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg. 18

[36] Michael Calonder, Vincent Lepetit, Christoph Strecha, and Pascal Fua. Brief: Binary robust independent elementary features. In Kostas Daniilidis, Petros Maragos, and Nikos Paragios, editors, *Computer Vision – ECCV 2010*, pages 778–792, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg. 18

[37] S. Leutenegger, M. Chli, and R. Y. Siegwart. Brisk: Binary robust invariant scalable keypoints. In *2011 International Conference on Computer Vision*, pages 2548–2555, Nov 2011. 18

[38] Howard Anton and Chris Rorres. *Elementary linear algebra : applications version.* New York : Wiley, 7th ed edition, 1994. 19

[39] Leibniz G. Explication de l'arithmtique binaire, die mathematische schriften. 7:223, 1879. 19

[40] Derek J. S. Robinson. *An Introduction to Abstract Algebra.* Walter de Gruyter, 2003. 19

[41] Y. Tian, L. Deng, and Q. Li. A knn match based tracking-learning-detection method with adjustment of surveyed areas. In *2017 13th International Conference on Computational Intelligence and Security (CIS)*, pages 447–451, Dec 2017. 20

[42] Christian Heipke and Franz Rottensteiner. Photogrammetric computer vision. 21

[43] R. I. Hartley. In defense of the eight-point algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(6):580–593, June 1997. 22

[44] Xiao-Shan Gao, Xiao-Rong Hou, Jianliang Tang, and Hang-Fei Cheng. Complete solution classification for the perspective-three-point problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(8):930–943, Aug 2003. 24

[45] Vincent Lepetit, Francesc Moreno-Noguer, and Pascal Fua. Epnp: An accurate o(n) solution to the pnp problem. *International Journal of Computer Vision*, 81, 02 2009. 24, 39

[46] Adrián Peñate Sánchez, Juan Andrade-Cetto, and Francesc Moreno-Noguer. Exhaustive linearization for robust camera pose and focal length estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35:2387–2400, 2013. 24

[47] Philip E. Gill and Walter Murray. Algorithms for the solution of the nonlinear least-squares problem. *SIAM Journal on Numerical Analysis*, 15(5):977–992, 1978. 25

[48] Kaj Madsen, Hans Nielsen, and O Tingleff. Methods for non-linear least squares problems (2nd ed.). page 60, 01 2004. 25, 27

[49] Daniel Simon. *Optimal State Estimation: Kalman, H Infinity, and Nonlinear Approaches*. John Wiley & Sons, 01 2006. 28

[50] Hamza Alkhatib. Script for filtering in the state space. 2018. 29

[51] Cesar Cadena, Luca Carlone, Henry Carrillo, Yasir Latif, Davide Scaramuzza, José Neira, Ian D. Reid, and John J. Leonard. Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *IEEE Transactions on Robotics*, 32:1309–1332, 2016. 31

[52] Zhongnan Qu. Efficient optimization for robust bundle adjustment. 2018. 31

[53] Rainer Kmmerle, Giorgio Grisetti, Hauke Strasdat, Kurt Konolige, and Wolfram Burgard. G2o: A general framework for graph optimization. *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 3607 – 3613, 06 2011. 32

[54] Nathaniel Merrill and Guoquan Huang. Lightweight unsupervised deep loop closure. *Robotics: Science and Systems 2018*, 06 2018. 33

[55] S. Lloyd. Least squares quantization in pcm. *IEEE Transactions on Information Theory*, 28(2):129–137, March 1982. 33

[56] D. Galvez-Lpez and J. D. Tardos. Bags of binary words for fast place recognition in image sequences. *IEEE Transactions on Robotics*, 28(5):1188–1197, Oct 2012. 33

[57] Sebastian Thrun. Learning metric-topological maps for indoor mobile robot navigation. *Artificial Intelligence*, 99:21–71, Feb 1998. 34

[58] Matia Pizzoli, Christian Forster, and Davide Scaramuzza. Remode: Probabilistic, monocular dense reconstruction in real time. *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2609–2616, 2014. 34

[59] George Vogiatzis and Carlos Hernandez. Video-based, real-time multi-view stereo. *Image and Vision Computing*, 29:434–441, 06 2011. 34

[60] H. Strasdat, J. M. M. Montiel, and A. Davison. Scale drift-aware large scale monocular SLAM. In *Robotics: Science and Systems VI*. Robotics: Science and Systems Foundation, Jun 2010. 37

[61] Martin A. Fischler and Robert C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24:381–395, 1981. 40

[62] Ethan Eade. Lie groups for computer vision. 12 2014. 61

[63] Ethan Eade. Lie groups for 2d and 3d transformations. 10 2018. 61