University of California
Los Angeles

Stereo Visual Odometry With Windowed Bundle Adjustment

A thesis submitted in partial satisfaction
of the requirements for the degree
Master of Science in Computer Science

by

Zhinan Xu

2015

Abstract of the Thesis

Stereo Visual Odometry With Windowed Bundle Adjustment

by

Zhinan Xu

Master of Science in Computer Science

University of California, Los Angeles, 2015

Professor Demetri Terzopoulos, Chair

Visual odometry (VO) is the process of estimating the egomotion of an agent (e.g., a vehicle, human, or robot) using only the input of a single or multiple attached cameras. In this thesis, we present a stereo visual odometry system for estimating the camera pose and surrounding three-dimensional structure from successive stereo image pairs. In order to generate the 3D-to-2D point correspondences, linear triangulation is performed between the stereo pair and a descriptor-based method is used to track feature points between frames. After obtaining the correspondences, the camera poses are initially estimated through a perspective-three-point (P3P) algorithm. During the initial estimation stage, the random sample consensus (RANSAC) method is used to perform outlier rejection. Finally, local estimation of the camera trajectory is refined by windowed bundle adjustment. Our algorithm differs from most visual odometry algorithms in three key respects: (1) it makes no prior assumptions about the camera motion, (2) it performs well on both small motions (e.g., wearable devices) and large motions (e.g., vehicles), and (3) it supports both indoor and outdoor environments. We test our system on a 22 km outdoor environment and a 0.9 km indoor environment and achieve 2.09% average translation error and 0.0067 deg/m average rotation error for every 800 m traveled.

The thesis of Zhinan Xu is approved.

Joseph M. Teran

Song-Chun Zhu

Demetri Terzopoulos, Committee Chair

University of California, Los Angeles

2015

Table of Contents

List of Figures

List of Tables

Acknowledgments

I am grateful to Professor Demetri Terzopoulos for being my thesis supervisor. He gave me flexibility in choosing the topic of my research and encouraged me to explore as much as possible to find my passion. My inquiry into virtual and augmented reality took me through automatic rigging, motion capture, simultaneous localization and mapping, and I finally settled on visual odometry. For each of the topics I explored, he gave me tremendous guidance about research methodology, such as which fields I should investigate further and how to make an impact given the existing techniques. I appreciated his careful and detailed reading of my thesis drafts. I have learned much from him about writing structure and style.

I am thankful to Professors Song-Chun Zhu and Joseph M. Teran for comprising the remainder of my committee and reviewing my thesis.

Finally, I am appreciative of the irreplaceable support of my parents Fajian Xu and Dongmei Li. They are my warm harbor and my sincere listeners, which encourages me to follow my heart throughout my life.

CHAPTER 1

Introduction

One of the goals of the field of Computer Vision is to help robots visually understand the world from both geometric and semantic perspectives. From the geometric aspect, the most well-known problem is the so-called structure from motion (SFM) problem, which is to recover the relative camera poses and three-dimensional structure of scenes from a set of camera images. SFM deals with sequentially ordered or unordered image sets. Like simultaneous localization and mapping (SLAM), it aims to obtain a globally consistent estimation of the camera pose and 3D structure, which is typically refined by an off-line optimization—e.g., global bundle adjustment (Triggs et al., 2000) or pose optimization. Visual odometry (VO) (Fraundorfer and Scaramuzza, 2011), which is a particular case of SFM, focuses on the local consistency of the camera trajectory and only takes sequentially ordered camera frames.

Most modern VO systems have real-time performance and they have been widely used in different application domains, including robotics, wearable computing, augmented reality, virtual reality, and automotive applications. Compared with other localization and tracking algorithms, VO has its own irreplaceable advantage. For example, unlike wheel odometry, it is not affected by wheel slip in uneven terrain or other unfavorable circumstances. Additionally, given the flexibility to combine VO with bundle adjustment and other pose optimization algorithms, it produces less drift over long-distance travel, which is unachievable even by high-precision inertial measurement units (IMUs).

In recent decades, many VO systems have been developed for both monocular and binocular scenarios. Most monocular VO systems require map initialization, which employs pure translation in the first few frames to generate an initial 3D estimate of the world. Additionally, resolving the scale ambiguity inherent to monocular VO usually requires assistance from other sensors (e.g., Lidar or IMUs). These characteristics of monocular VO make it less applicable in industry. Currently emerging depth-based VO systems, similar to stereo VO, generate high-precision depth information at each frame. However, most depth cameras use structured light to increase the matching accuracy, which makes them suitable only for indoor environments with a limited depth range.

In this thesis, we develop a high-precision stereo visual odometry system. There are several ways to perform pose estimation. The three most well-known methods are

- **2D to 2D:** Given 2D-to-2D point correspondences between two successive frames, the essential or fundamental matrix is calculated through Nister's five-point algorithm (Nister, 2003). Based on $E \cong \hat{t}R$, the camera pose is extracted from the essential matrix.

- **3D to 2D:** given the 3D-to-2D point correspondences between previous and current frames, the camera pose is calculated as the extrinsic camera matrix for the current frame.

- **3D to 3D:** given the 3D-to-3D point correspondences between two successive frames, the camera pose is computed by determining the aligning transformation of those two 3D point clouds.

In our pipeline, instead of dealing with the unknown scale in the 2D-to-2D case and the unpredictable error propagation in the 3D-to-3D case, we choose the 3D-to-2D case as Nister et al. (2004) recommend in their landmark paper. To support large

motions, we use a descriptor based method to find the correspondences across frames. We compute the initial estimation of camera pose by combining the RANSAC method (Fischler and Bolles, 1981) with the P3P (Kneip et al., 2011) method. Finally, we use a windowed bundle adjustment to locally optimize the camera trajectory.

1.1 Thesis Contributions

The contributions of this thesis are as follows:

- a simple but effective threshold method is developed to solve the "fake motion" problem due to imperfect triangulation.
- a Huber loss function is designed to reduce the weight of outliers in windowed bundle adjustment.
- throughout the literature on stereo VO, there are only a few discussions about how to choose the window size and what the effects of different window sizes are in bundle adjustment. Our evaluation includes an in-depth discussion of these issues.

Additionally, in this thesis we re-evaluate the effectiveness and efficiency of different key-point detectors and descriptors. We find that rotation and affine-transformation invariant descriptors actually increase the quantity of outliers, which causes instability in the RANSAC algorithm.

1.2 Thesis Overview

The remainder of this thesis is organized as follows: Chapter 2 reviews relevant prior work on monocular and stereo VO, as well as the key-point-based method and image-alignment-based method. Chapter 3 gives a overview of our stereo VO algo-

rithm. Chapter 4 includes a detailed discussion of scene reconstruction, including detectors and descriptors, stereo matching, and triangulation. Chapter 5 introduces the method used to perform motion estimation, including correspondence matching and pose estimation. Chapter 6 mainly discusses motion optimization, and presents the correspondence tracking and windowed bundle adjustment methods. Chapter 7 evaluates our system in indoor and outdoor environments, with small and large motions, and the performance of bundle adjustment with different window sizes. Chapter 8 concludes the thesis with a review of the entire pipeline and the contributions of our work.

CHAPTER 2

Relevant Prior Work

In recent decades, many visual odometry algorithms have been developed. They may be divided into two main categories based on the platform—approaches using a monocular camera (e.g., (Yamaguchi et al., 2006)) and approaches using a stereo camera (e.g., (Nister et al., 2004), (Kitt et al., 2010)). These approaches can be further separated into methods that either use feature-based (e.g., (Howard, 2008)) or appearance-based (e.g., (Engel et al., 2014), (Engel et al., 2013)) methods. In this section, we review prior work in both categories.

2.1 Monocular and Stereo Visual Odometry

Most VO research has used stereo rigs. If a well-calibrated stereo camera setup is available, the 3D scene can be reconstructed via triangulation or a block-matching algorithm. Given point clouds of the surrounding environment reconstructed from two consecutive images, either the Perspective-N-Point (Kneip et al., 2011) algorithm or Iterated Closest Point (Milella and Siegwart, 2006) algorithm is used for camera motion estimation. Since the depth camera is a stereo camera with a structured light system, we will not discuss it separately. In the landmark paper by Nister et al. (2004), the term VO was first introduced and a VO pipeline was presented. Based on error accumulation and propagation, the concept of "firewall" was introduced to avoid triangulating every stereo pair. Subsequently, many algorithms were proposed to optimize the core components of the general pipeline. Instead of using standard corner detectors, Konolige et al. (2011) suggested the

use of center surround extreme (CenSurE) features for correspondence matching. Moreover, IMUs were integrated into the VO system to increase stability. However, for the scale-invariant descriptors, the feature transformation and correspondence matching are both time consuming. In 2008, Howard (2008) designed an inlier detection algorithm that was capable of handling correspondences with a high percentage of outliers. By using the inlier detection method, a standard corner detector and an NCC dissimilarity score can be used. They implemented a real-time and robust stereo VO system with 0.25% translation error within 400 meters.

In monocular VO systems, both the relative motion and 3D structure must be computed from 2D bearing data. Since the absolute scale is unknown, the distance between two camera poses is usually set to one or extracted from other sensors, such as IMUs or wheel odometry. As a new image arrives, the relative scale and camera pose with respect to the first two frames are obtained using either the knowledge of 3D structure or the trifocal tensor (Hartley and Zisserman, 2003). In 2004, Nister et al. (2004) proposed the first real-time, large-scale VO system with a monocular camera. RANSAC was used for outlier rejection and the camera pose was calculated through 3D-to-2D camera pose estimation. Instead of using feature matching, Corke et al. (2004) provided an approach for monocular VO based on an omnidirectional camera and optical flow.

2.2 Feature-Based and Appearance-Based Methods

Feature-based methods employ salient and repeatable features that are tracked over multiple frames, which usually consists of two separate steps. First, the discrete feature observations are detected and matched to each other. Second, the camera pose is calculated based on a set of feature observations. The feature transformation step greatly reduces the complexity of the original problem

and allows it to be tackled in real time. Feature-based methods are a dominant approach to tackling the pose-estimation problem, and they were employed in all the publications cited in the previous section. However, there are two main problems with the feature-based approach: Only information conforming to the respective feature type is utilized. Moreover, the robust correspondence matching algorithm normally requires a huge amount of computation of scale-invariant and rotation-invariant descriptors.

Appearance-based methods use intensity information in all the pixels in the image or in image subregions. Based on a dense or semi-dense depth map, most appearance-based methods perform motion estimation using image alignment. Kerl et al. (2013) directly obtain a dense depth map from a depth camera. The surrounding environment is modeled as a dense surface and new frames are tracked using whole-image alignment. Due to strong ambient illumination, however, most structured-light-based depth cameras are not applicable in outdoor scenarios. Engel et al. (2013) proposed a monocular VO system with a semi-dense depth map. In their system, depth information is computed only for regions with non-negligible gradient. They proposed a probabilistic depth map representation, by modeling the depth of each pixel as a Gaussian distribution. By using a depth map, camera motion can be estimated through image alignment techniques. The probabilistic depth map is updated based on the obtained camera motion.

CHAPTER 3

System Overview

Similar to the general pipeline of feature point based VO, our system consists of three main components—scene reconstruction, motion estimation, and motion optimization. In the first stage, our system performs stereo camera calibration to extract the intrinsic matrix, distortion parameter, and length of the baseline. For each input image sequence, image undistortion and stereo rectification are applied during the preprocessing stage. After preprocessing, feature transformation performed to convert the image into a set of descriptors, which are image patches for the Harris (Harris and Pike, 1987) and FAST (Rosten and Drummond, 2006) corner detector cases. Then there are two major methods for correspondence matching—either feature tracking (e.g., optical flow (Berthold and Brian, 1981)) or feature matching (e.g., SIFT (Lowe, 2004)). It is well known that feature tracking is efficient and able to achieve great performance for small motions between two successive frames. Conversely, descriptor-based feature matching is much more time-consuming, but it supports both small and large motions. As mentioned earlier, our system imposes no constraints or prior assumptions regarding the camera motion. To accomplish our goals, we use the descriptor based method to perform correspondence matching and tracking in different stages. Based on the point correspondences, the scene reconstruction and pose estimation can be done through triangulation and the perspective-three-point algorithm. Finally given the result of continuous tracking, windowed bundle adjustment is applied to refine the camera trajectory. Rather than general bundle adjustment, only the

camera poses are optimized in our system instead of optimizing both camera poses and the 3D point cloud. There are two main factors that makes us choose this approach.

1. The 3D point cloud is generated from stereo views. If the stereo calibration is unbiased, we should obtain high-precision positioning for each 3D point.

2. Adding the 3D point cloud into the optimization dramatically increases the dimensionality of problem and the number of iterations required to reach convergence.

Comparing the generated result and ground truth data, we found that the windowed bundle adjustment largely reduces both translation and rotation drifting, especially over long distance travel. It is well known that there is another kind of motion optimization, closure detection, which could be used to detect loops and further improve the precision. However, closure detection is not included in our motion optimization, because the generated large-scale bundle adjustment is impossible to conduct in real time. Also, rather than achieving global consistency, the main focus of our VO system is a local optimization of the trajectory.

CHAPTER 4

Scene Reconstruction

4.1 Detector and Descriptor

In the first stage of our pipeline, the image is searched for salient feature points that are likely to match well in other images. Clearly, it is pointless to perform motion estimation with respect to undistinguished points (e.g., points located on a pure white wall), because it will be impossible to determine whether corresponding points are displaced relative to one another. Similarly, for image intensity edge points, only the motion perpendicular to edge can be detected, as is demonstrated by the barber pole illusion. For VO, there are two kinds of salient points, corners and blobs. As shown in Figure 4.1(a), a corner is defined as a point at the intersection of two or more edges. As shown in Figure 4.1(b), a blob is an image patch that differs from its adjacent pixels.

A great deal of relevant literatures investigates the efficiency and effectiveness

(a) (b)

Figure 4.1: Comparison between corners and blobs

of different salient points. Here we list several valuable properties that good detectors and descriptors should have:

- accuracy of localization for both position and scale; e.g., sub-pixel optimization of the corners and blobs;
- computational efficiency, e.g, around 2–5 ms;
- robustness; e.g, invariance to both photometric and geometric changes.

During our testing and evaluation, we found that the accuracy of key-point-based approaches is highly correlated with the accuracy of the correspondence matching, which depends on the qualities of the detector and descriptor. As a general investigation, we evaluate our correspondence matching system according to different detectors and descriptors, which is discussed thoroughly in Section 5.. Here, we give a brief introduction about different detectors and descriptors tested in our system.

4.1.1 The Harris Corner Detector

In recent decades, while many different definitions for selecting interest points have been proposed, the most popular approach remains the Harris corner detector (Harris and Pike, 1987), which was introduced almost three decades ago. Nowadays, it appears in hundreds of industrial applications and the original paper has been cited more than 10,000 times. It is reasonable to test our correspondence matching system on this classical detector first.

In general, for each target point (x, y) inside the image, the Harris corner response function is calculated over a small patch around it. Here, we derive the response function in an intuitive way. By shifting a small window over the image, the corner is recognized by observing the intensity change during shifting:

- flat region: no change in any directions;

- edge: no change along the edge direction;
- corner: significant change in all directions.

As the result, an energy function is constructed by aggregating intensity changes in the patch centered at (x, y) and over the direction (u, v), which is

$$E(u, v) = \sum_{x,y} w(x, y)[I(x + u, y + v) - I(x, y)]^2.$$

The ultimate goal is to find the (x, y) that has large energy for all directions (u, v). After applying a first-order Taylor expansion of $I(x + u, y + v)$, we obtain the approximated objective function

$$\begin{aligned} E(u, v) &= \begin{bmatrix} u & v \end{bmatrix} \begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} \\ &= DMD^T. \end{aligned}$$

By analyzing the correlation between the eigenvalues (λ_1 and λ_2) of M the point is classified as follows:

- if both λ_1 and λ_2 are small, then $E(x, y)$ is almost constant in all directions, thus the target point (x, y) is located in a constant-intensity region;
- if $\lambda_1 \gg \lambda_2$ or $\lambda_1 \ll \lambda_2$, then $E(x, y)$ increases only in one direction, thus the target point (x, y) is located on an intensity edge;
- if both λ_1 and λ_2 are large, then $E(x, y)$ increases in all directions, thus the target point (x, y) is a corner.

Instead of calculating the eigenvalues of M, we construct the response function as

$$R = \det(M) - k\,\mathrm{trace}(M).$$

According to the response function, $R(x, y)$ is calculated for each (x, y). To reduce the clustering and retrieve more valuable information, thresholding and non-maximum suppression are performed. Finally, a set of coordinates that represent the location of corners is obtained. Additionally, image patches around the extracted corners are used as descriptors. The sum of squared differences (SSD) is used to calculate the dissimilarity score between two patches. However, it varies with both geometric and photometric changes. As is shown in Section 6, the SSD is not robust and accurate for large-region correspondence matching, which is necessary in the presence of large camera motion. Unsurprisingly, it performs quite well for small-region and epipolar line searching.

4.1.2 The FAST Corner Detector

The Features from Accelerated Segment Test (FAST) corner detector was proposed by Rosten and Drummond (2006). Because of its efficiency, it has became a popular alternative to the Harris corner detector, especially for applications in mobile robotics, which have limited computational resources. Instead of evaluating the response function for each pixel, it first performs a high-speed test to eliminate bad candidates. If pixel p passes the previous test, the full-segment test is performed to decide whether p is a corner. Additionally, the parameters used in those two tests are generated through a training process.

As shown in Figure 4.2, for each target point $p(x, y)$, considering a circle of 16 pixels around it, the pixel p is a corner if there exist n contiguous pixels in the circle which are all brighter than $I_p + t$ or darker than $I_p - t$, where n is chosen by the training process.

The high-speed test proposed in FAST is actually used to eliminate the non-corners. Instead of testing all 16 pixels, it first applies the rule to the 4 pixels located at 0°, 90°, 180°, and 270° degrees of the circle. If p is a corner, then

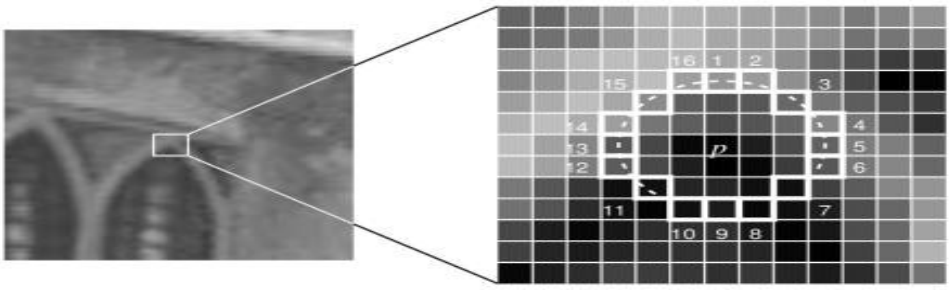


Figure 4.2: FAST corner detector full-segment test circle

at least three out of four points should be brighter than $I_p + t$ or darker than $I_p - t$. After passing the initial test, the algorithm applies the full-segment test by examining all pixels in the circle.

In our test, the FAST corner detector was approximately 1.5 times faster than the Harris corner detector. Even after non-maximum suppression, a far greater number of corners are detected by the FAST corner detector than by the Harris corner detector. However, the FAST corner detector is less flexible than Harris corner detector. For the Harris corner detector, it is possible to choose the most distinguished set of corners by ranking them based on the response function value. FAST relies only on the two rounds of tests, which are the high-speed test and the full-segment test. Without the response function, it is almost impossible to compare the qualities of two corners. Hence, due to the large number of outliers generated through correspondence matching, RANSAC takes many more iterations to achieve expected accuracy.

4.1.3 SIFT and SURF

The biggest problem with corner detectors such as the FAST and Harris detectors is that they are not scale invariant. These two detectors are slightly rotation-invariant. After applying a rotation transformation, a corner would be still recognized. Based on the correspondence matching scheme, they might have a large dissimilarity score, which would lead to a mismatch. But the corner is repetitive

after the rotation. However, based on the single-size sliding window method, they are definitely not scale-invariant. In addition, for the FAST and Harris detectors, without a good definition of the descriptor, the dissimilarity score is usually calculated as an SSD or NCC between two image patches. In this way, it is also illuminant-variant and contrast-variant.

Lowe (2004) proposed the Scale-Invariant Feature Transform (SIFT) by applying a Difference of Gaussians (DoG) across the discretized scale space and extracting the local maxima. Additionally, the descriptor of the feature points is well-defined. For each keypoint, a 16×16 neighborhood located in appropriate scale space is taken. It is divided into 16 sub-blocks, where each generates an 8-bin histogram. For each histogram, both the scale and orientation of the derivative is taken into account. By detecting local maxima in scale space and gathering 128-bin values as a descriptor, SIFT has the following advantages:

- By employing the derivative, the descriptor is illuminant-invariant.
- By normalizing the scale of the derivative, the descriptor is contrast-invariant.
- By performing non-maximum suppression and processing the image patch in scale space, the descriptor is scale-invariant.
- By assigning the orientation as the highest peak in the histogram, the descriptor is rotation-invariant.

However, SIFT is not an ideal descriptor. During the tests, we found that the speed of SIFT is about 7–10 times slower than the Harris corner detector. Even for the CUDA version, it is about 4–6 times slower. Additionally, the rotation-invariant and heuristic affine-invariant properties increase the total number of outliers during correspondence matching, especially for indoor environments with many repetitive patterns. For most VO applications (e.g., automotive, wearable glasses), we normally have only small rotations around the z (depth) axis. As the

result, we disable the rotation-invariant property, which works well in most cases, but is still too time-consuming.

To further increase the efficiency, we also evaluated the Speed-Up Robust Features (SURF) (Bay et al., 2006) detector for correspondence matching. Instead of using the DoG approximation of the Laplacian of Gaussian, SURF introduces the box filter to perform blob detection. Additionally, by disabling the rotation-invariant property, it generates decent results in correspondence matching. However, in our tests, the speed of SURF is very similar to that of SIFT, which is hardly acceptable for a real-time VO system.

4.1.4 ORB

Investigating different detectors and descriptors, we found that corner features and blobs have their advantages and disadvantages:

- Corner detectors are time-efficient since most of them are based on segmentation rules and the eigenvalues of the Hessian matrix inside a small image patch. However, the standard algorithms (e.g., Harris, FAST) do not have a well-defined descriptor, which makes them less robust during matching.

- Blob detection is time consuming, because the processing must be done in scale space to extract the scale information and the Laplacian of Gaussian is needed to accomplish blob detection. SIFT and SURF use different approximations to perform blob detection, but the processing time is still far from real time. However, the descriptor is invariant to most illuminant and geometric changes, which yields decent result during matching.

As the result of our investigation, we decided to find a corner detector with a well-defined descriptor, and finally settled on Oriented FAST and Rotated BRIEF (ORB). This algorithm was introduced by Rublee et al. (2011). The authors

claim that it is a good alternative to SIFT and SURF in computational cost and matching performance. ORB is basically a fusion of the FAST corner detector and the BRIEF descriptor with some modifications to make it scale-invariant and rotation-invariant. It first applies FAST to find keypoints, and then uses the Harris corner response function to find the top N points among them. Instead of processing in scale space, it uses an image pyramid to produce multiscale features. To extract the orientation information, it computes the intensity weighted centroid of the corner patch and represents the orientation θ as the angle between the x axis and the vector between the centroid and the corner. For the descriptor, ORB uses a modified version of the BRIEF descriptor. After constructing a $2 \times n$ matrix S for n binary tests, it generates S_θ as $R(\theta)S$, where $R(\theta)$ is the rotation matrix.

The ORB detector and descriptor have similar properties to SIFT.

- By performing key point detection and descriptor extraction in the pyramid, the descriptor is scale-invariant.
- By converting S to S_θ, the descriptor is rotation-invariant.
- Inheriting the properties of BRIEF, the descriptor is illuminant-invariant and contrast-invariant.

Additionally, the speed of the ORB detector is very similar to the Harris corner detector (1.5 times slower).

To build a real-time VO system with high precision, a robust and computationally-efficient descriptor is necessary. In our system, we use the ORB detector to perform the feature transformation.

4.2 Stereo Matching

In our system, we use the general pipeline to perform stereo matching, including stereo calibration, stereo rectification, and epipolar line searching. Stereo cali-

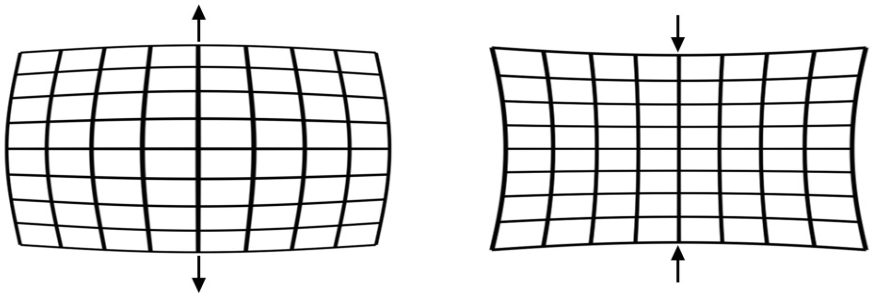


Figure 4.3: Radial distortion examples: barrel and pincushion effects

bration is accomplished in the first stage. Most modern stereo systems perform undistortion and rectification during the video recording stage with hardware acceleration. Our system undistorts and rectifies the image sequence in a preprocessing step.

4.2.1 Stereo Calibration

In order to build a high-precision VO system, unbiased stereo calibration is essential. While there exist many algorithms to perform monocular camera calibration, stereo camera calibration remains a non-trivial problem. Here, we briefly introduce the model we used to perform calibration. The first step of stereo calibration is to perform monocular calibration for both cameras.

The purpose of monocular camera calibration is to obtain the distortion coefficients and the intrinsic matrix. It is well-known that there are two main kinds of distortions:

1. Radial distortion as shown in Figure 4.3, manifested in the form of "barrel" and "pincushion" effects. The relationship between the original and undistorted coordinates could be formulated as

$$x_{corrected} = x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6),$$
$$y_{corrected} = y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6).$$

2. Tangential distortion, which is caused by imperfect parallel alignment between the lenses and the image sensor. The relationship between between the original and undistorted coordinates could be formulated as

$$x_{corrected} = x + [2p_1xy + p_2(r^2 + 2x^2)],$$
$$y_{corrected} = y + [p_1(r^2 + 2y^2) + 2p_2xy].$$

As a result of continuous improvements in the technology, modern camera systems normally do not suffer from much tangential distortion. However, in order to perform unbiased calibration, tangential distortion is also taken into account in our system. Additionally, considering the fact that both monocular and stereo VO systems use wide-angle lenses, radial distortion must be calculated to eliminate "barrel" or "fish-eye" effects.

After establishing the relationship between the original and undistorted coordinates, we must transform between the 3D and undistorted 2D coordinates. The standard approach is to define this transformation using homogeneous coordinates:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} fm_x & r & u_0 \\ 0 & fm_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} R & t \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}.$$

Here, $\begin{bmatrix} R & t \end{bmatrix}$ is the extrinsic camera matrix. During monocular camera calibration, the world coordinates are the same as the camera coordinates. So the extrinsic camera matrix is just the identity matrix, which can be ignored. By looking at the intrinsic camera matrix, f is focal length, m_x and m_y are the scale factor relating pixels to distance, γ is the skew coefficient between the x and y axes, and u_0 and v_0 are the coordinates of the principle point. Pixels are normally square nowadays and, as the result, it is easy to derive that $m_x = m_y$ and $\gamma = 0$.

To determine the intrinsic and distortion parameters, we use the standard chessboard approach incorporated into the OpenCV library. Given a sequence of chessboard images, the algorithm first detects "good" frames, where the all the corners are visible and sharp enough. For each good frame, by performing a sub-pixel optimization, the corner coordinates are extracted with sub-pixel precision. Since the 3D coordinates are known, the distortion and intrinsic parameters are generated by calculating a homography transformation between the chessboard and the image planes.

After obtaining the required parameters of both cameras, we perform stereo calibration to obtain the relative R and t between the two cameras. Given the 3D and undistorted 2D coordinates of the chessboard corners, the extrinsic camera matrix can be derived using the perspective-three-point algorithm (Kneip et al., 2011). According to $[R_1\ t_1]$ and $[R_2\ t_2]$, the relative transformation $[R\ t]$ can be easily calculated as a rigid-body transformation.

There are other methods for performing stereo calibration. For example, given the 3D and 2D chessboard corner correspondences of both image sequences, all the required information can be obtained by solving an optimization problem. The objective function is the reprojection error for both camera frames. But our tests indicate that this method can generate biased results. The dimension of this optimization is large, which requires a decent initial state. However, in most situations, the initial state is created based on empirical data. Due to the heuristic initial state, the optimization might converge to an unfavourable local minimum. As mentioned earlier, camera calibration errors might cause serious problems in the subsequent pipeline due to error propagation. As the result, we use a separate calibration method in our system.

4.2.2 Stereo Rectification

For an ideal stereo camera, the two image sensors should be parallel. However, according to the stereo calibration result, it is almost impossible to build an ideal model. The rotation matrix R obtained in the last step is not the identity matrix. There are two possible ways to perform epipolar line searching:

1. For each feature point in left image, the system performs th epipolar line searching on the right image. The corresponding epipolar line is generated according to $p'^T Ep = 0$.

2. For each feature point in left image, the system performs matching on the same row in the right image, after stereo rectification.

If the number of feature points is small, it is reasonable to use the first approach. Otherwise, the second approach is preferable because calculating the epipolar line for a large number of feature points would be too time-consuming. The ultimate goal of our VO system is not only calculating the 3D trajectory of the camera, but also performing a 3D reconstruction of the surrounding scene. Hence, we try to include as many key points as possible. Additionally, for modern stereo systems, image undistortion and rectification are usually conducted during the video recording stage with hardware acceleration. As a result, our system includes stereo rectification in the preprocessing stage. We will briefly introduce the stereo rectification algorithm that we use:

1. Rotate the left camera by R_{rect} so that epipole e_l goes to infinity along the horizontal axis.

2. Apply the same rotation to the right camera to recover the original geometry.

3. Rotate the right camera by R^{-1} to lay the two image planes into a common plane, where R is relative rotation between the left camera and right camera.

The construction of R_{rect} is illustrated next. To move epipole e_l to infinity along the x axis, the new x axis of the left camera coordinate system must have the same direction as $e_1 = \frac{T}{||T||}$. Because this problem is under-constrained, the only constraint on the y axis is that it must be orthogonal to e_1. To this end, the direction of the y axis is obtained by computing and normalizing the cross product of e_1 and the direction vector of the optical axis, $e_2 = \frac{1}{\sqrt{T_x^2+T_y^2}}[-T_y, T_x, 0]^T$. The z axis is unambiguously determined as $e_3 = e_1 \times e_2$. Therefore, the rotation matrix will be define as

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

Rotating the camera actually reprojects every pixel into a new position, which is a homography transformation H_{rect}. For the left camera, the original position x is obtained through

$$x = K[I|0] \begin{bmatrix} X \\ 1 \end{bmatrix} = KX.$$

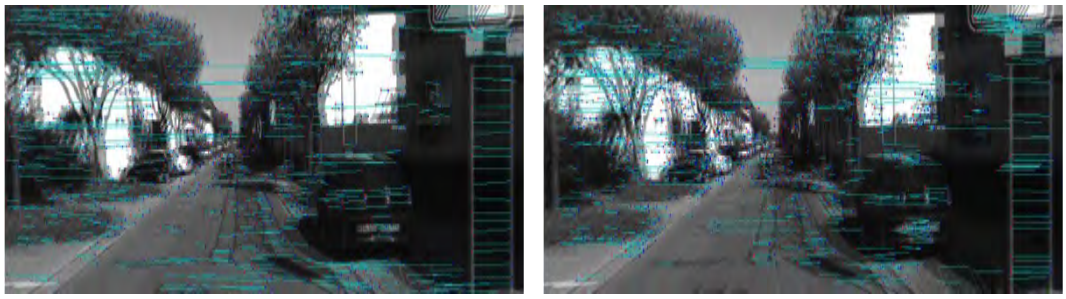
After applying the rotation, the new position x' is generated by

$$x' = K[R_{rect}|0] \begin{bmatrix} X \\ 1 \end{bmatrix} = KR_{rect}X,$$

where $H_{rect} = KR_{rect}K^{-1}$. The homography transformation for the right camera can be constructed in a similar way. In the end, a rectified image pair is obtained by applying the homographies to the corresponding left and right images.

4.2.3 Epipolar Line Search

After stereo rectification, for each feature point in left image, it suffices to search the same row in the right image. Two options are available to calculate the dissimilarity score:



(a) Classical epipolar line search (b) Descriptor based epipolar line search

Figure 4.4: Comparison between standard and descriptor-based epipolar line search.

1. Perform the feature transformation for both left and right image. For each feature point $p(x, y)$ in the left image, calculate the dissimilarity score only for feature points with same x in the right image.

2. Perform the feature transformation on the left image only. For each feature point $p(x, y)$ in the left image, perform the standard epipolar line search in the same row in the right image.

Our ultimate goal is to implement a high-precision and real-time VO system. We found that the first approach is much slower, but it generates only slightly better results than the second approach, as shown in Figure 4.4.

Actually, for most of stereo camera available in the market, the rotation and baseline are both small between the two cameras. It is unnecessary to use a scale-invariant and rotation-invariant descriptor to perform stereo matching. Additionally, both generating robust descriptors and calculating the distances between descriptors is time-consuming.

Next, we will briefly discuss the epipolar line search algorithm. For each feature point (x, y) in the left image, we use a 5-column image patch

$$[I_l(x, y-2), I_l(x, y-1), I_l(x, y), I_l(x, y+1), I_l(x, y+2)]$$

as the descriptor. Along the same row of the right image, we compute the dissimilarity score for each point by calculating the sum of absolute difference (SAD) between the two descriptors. Unsurprisingly, the dissimilarity score can be computed in an efficient way by using dynamic programming, where

$$d(x, y+1) = d(x, y) - |I_l(x, y-2) - I_r(x, y-2)| + |I_l(x, y+3) - I_r(x, y+3)|.$$

During the search, we record the lowest score d_1 with (x_1, y_1) and the second lowest score d_2 with (x_2, y_2). If $\frac{d_1}{d_2} < T$, the point (x, y) and (x_1, y_1) is a stereo pair. Otherwise, there is no stereo match for (x, y). Actually, the purpose of the ratio test is to eliminate ambiguous stereo correspondences.

4.3 Triangulation

There exist many triangulation methods; e.g., depth from disparity, the mid-point method, and linear triangulation. After considering both precision and efficiency, we chose the linear triangulation method, which obtains optimal 3D coordinates in the least-squares sense. Here, the detailed derivation of this method is introduced. Given $x = PX$, it is easy to obtain $x \times (PX) = 0$, which gives us

$$\begin{aligned} x(p^{3T}X) - (p^{1T}X) &= 0, \\ y(p^{3T}X) - (p^{2T}X) &= 0, \end{aligned}$$

where p^{iT} is row i of P. Combining $x' \times (P'X') = 0$, we write an equation of the form $AX = 0$ with

$$A = \begin{bmatrix} xp^{3T} - p^{1T} \\ yp^{3T} - p^{2T} \\ x'p'^{3T} - p'^{2T} \\ y'p'^{3T} - p'^{2T} \end{bmatrix}.$$

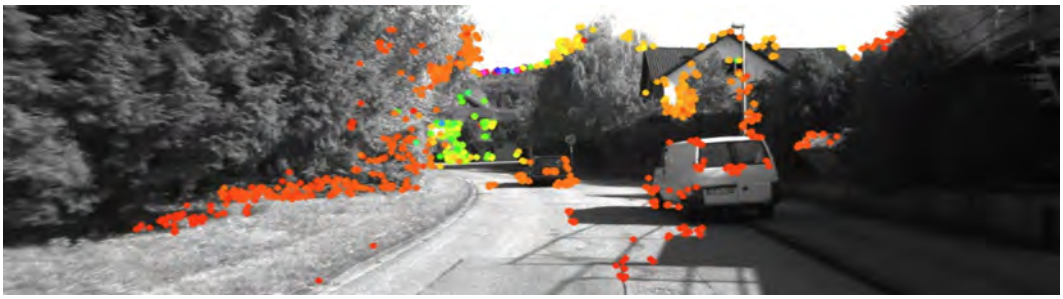


Figure 4.5: The sparse depth map obtained by triangulation.

Here, $AX = 0$ is a homogeneous equation, which may be solved using the singular value decomposition (SVD). However, it is well known that the SVD is one of the most time-consuming of all the matrix factorizations. In order to improve efficiency, a transformation of $AX = 0$ is adapted. Because the last element of the homogeneous coordinate X is 1, it is possible to rewrite the homogeneous function as

$$\begin{bmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \\ A_{41} & A_{42} & A_{43} \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} -A_{14} \\ -A_{24} \\ -A_{34} \\ -A_{44} \end{bmatrix}.$$

Now, the 3D coordinates (x, y, z) can be solved using the pseudo-inverse, which is much more efficient than the SVD.

This approach has a limitation, which is that the equation transformation is based the fact that point X is not located at infinity. However, in real-world applications, there are only a few infinite points. As the result, the pseudo-inverse approach will not affect the precision.

CHAPTER 5

Motion Estimation

Motion estimation is the core computational step performed on every image in a VO system. During motion estimation, the relative camera motion between the current frame and the previous frame is computed. By concatenating all the relative motions, the full trajectory of the camera can be recovered. This section mainly explains how the correspondence matching between two frames is performed and how the relative camera motion is calculated through the 3D-to-2D correspondences. Before describing the detailed implementation of our system, it is beneficial to discuss the alternative motion estimation techniques.

2D-to-2D method: This is the earliest approach developed by the SFM community. For simplicity, we will assume that the camera is calibrated and the image coordinates are normalized (the inverse intrinsic matrix K^{-1} has been applied). Given the 2D correspondences between the previous and current frames, the essential matrix E can be calculated. It is well-known that E contains the camera motion with an unknown scale for the translation vector, which is $E \cong \hat{t}R$, where $\hat{t}$ is the skew-symmetric matrix of t, which takes the following form:

$$\hat{t} = \begin{bmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_t & t_x & 0 \end{bmatrix}.$$

In general, for each essential matrix E, there are only a finite number of solutions for R and t. By triangulating a single point, the correct R and t can be identified

by choosing the solution where the point is in front of both cameras. However, the scale of t is still unknown. In industrial applications, people normally take advantage of IMUs or wheel odometry to determine the unknown scale.

Next, the calculation of the essential matrix is briefly discussed. The essential matrix E can be computed from 2D correspondences using the epipolar constraint $p'^T Ep = 0$, where p and p' is a correspondence pair. The essential matrix can be calculated through at least five point correspondences. In most cases, RANSAC is used to perform the outlier rejection and increase robustness. Instead of discussing the five-point algorithm, we discuss the famous Longuet-Higgins (1981) eight-point algorithm. Based on the epipolar constraint $p'^T Ep = 0$, each correspondence pair contributes one constraint on E. Because the estimation of E is up to an unknown scale, at least eight correspondence pairs are needed. Rearranging, an equation in the form of $AX = 0$ can be constructed:

$$\begin{bmatrix} x_1x'_1 & x_1y'_1 & x_1 & y_1x'_1 & y_1y'_1 & y_1 & x'_1 & y'_1 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_8x'_8 & x_8y'_8 & x_8 & y_8x'_8 & y_8y'_8 & y_8 & x'_8 & y'_8 & 1 \end{bmatrix} \begin{bmatrix} e_1 \\ e_2 \\ e_3 \\ e_4 \\ e_5 \\ e_6 \\ e_7 \\ e_8 \end{bmatrix} = 0.$$

This homogeneous equation can easily be solved using SVD with the constraint $||E|| = 1$. However, according to $p'^T Ep = 0$, the rank of E is 2. To enforce the rank-2 constraint, a new $\bar{E}$ is generated as

$$\bar{E} = \mathrm{U}\mathrm{diag}(\sigma, \sigma, 0)V^T,$$

where $E = \text{Udiag}(\sigma_1, \sigma_2, \sigma_3)V^T$ and $\sigma = \frac{\sigma_1+\sigma_2}{2}$.

3D-to-3D method: This method is the most used in stereo and depth camera VO systems. The camera motion can be computed by determining the aligning transformation between two 3D point clouds. Given stereo pairs in two consecutive frames $I_{l,k-1}$, $I_{r,k-1}$ and $I_{l,k}$, $I_{r,k}$, the general pipeline for the stereo VO system is as follows:

1. Perform the feature transformation for both $I_{l,k-1}$ and $I_{r,k-1}$.
2. Perform correspondence matching between the two sets of feature points.
3. Triangulate the matched features for the stereo pair to get the 3D points set X_k and X_{k-1}.
4. Compute the camera transformation T_k based on the 3D correspondences between X_{k-1} and X_k.

To calculate the relative camera motion, the general solution is to find the motion T_k that minimizes the L_2 distance for all 3D correspondences:

$$T_k = \arg\min_{T_k} \sum_i ||X_k^i - T_k X_{k-1}^i||.$$

To solve T_k, at least three non-collinear correspondences are necessary. According the method proposed by Arun et al. (1987), one possible solution is to compute the translation as the difference of the centroids of the two 3D feature sets, which are

$$t_k = \bar{X}_k - \bar{X}_{k-1},$$

where symbol $\bar{\cdot}$ denotes the arithmetic mean value. The rotation matrix can be obtained through SVD. Given $(X_{k-1} - \bar{X}_{k-1})(X_k - \bar{X}_k)^T = USV^T$, matrix R_k may be calculated as

$$R_k = VU^T.$$

5.1 Correspondence Matching

It is well known that there are two main approaches to performing correspondence matching—feature tracking and feature matching—which we we briefly introduce next.

Feature tracking: based on the Lambertian assumption, the pixels of the same 3D point should have the same intensity in successive images. Additionally, the relative camera motion must be a rigid-body transformation. Here, by assuming that the relative motion is very small, the algorithm simplifies the rigid body motion to a translation model, where

$$I(x(t), t) = I(x(t) + udt, t + dt).$$

Approximating $I(x(t) + udt, t + dt)$ using the first-order Taylor expansion, it is easy to get

$$\nabla I(x(t), t)^T u + I_t(x(t), t) = 0.$$

The objective function is obtained by integrating over an image patch:

$$E(u) = \sum_{w(x,y)} ||\nabla I^T u + I_t||^2.$$

The goal is to find the u for each (x, y) that minimizes the objective function. Here, the general optimization approach is taken, by setting $\nabla E(u) = 0$. The equation $\nabla E(u) = 0$ can be written in the form $Gu + b = 0$, as follows:

$$\begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix} u + \begin{bmatrix} \sum I_x I_t \\ \sum I_y I_t \end{bmatrix} = 0.$$

It is easy to see that optimal solution exists only if $\text{rank}(G) = 2$. In real-world applications, this will be true in most circumstances due to noise. Consequently,

similar to the Harris corner detector, the u will be calculated only if the eigenvalues λ_1 and λ_2 of $G(x, y)$ are both sufficiently large.

The problem of the feature tracking algorithm is its assumptions. The Lambertian assumption yields the constraint that the projection of the same 3D point should have the same illuminant, which is similar to performing feature matching with an illuminant-variant descriptor. Additionally, the translation model assumes that the baseline between the two frames is very small. In conclusion, the feature tracking works well only for small motions with constant illuminant.

Feature matching: the goal is to find correspondences between two sets of descriptors. Depended on the search strategy, exhaustive or constrained searching is conducted. By calculating the dissimilarity score between each pair of descriptors, the score matrix is constructed. For each feature point in frame $k-1$, the best match in frame k is chosen. Finally, based on the matching strategy, some of the "best" matches might be discarded due to ambiguity or consistency. In the following section, the search strategy, dissimilarity score, and matching strategy will be discussed in detail.

5.1.1 Exhaustive and Constrained Search

As the name exhaustive search suggests, for each point in frame $k-1$, all the points in frame k are searched and the corresponding dissimilarity scores are evaluated. The time complexity is quadratic in the number of feature points, which can become impractical when the number of features is large. Actually, in addition to its time complexity, exhaustive search generates many more bad correspondences, especially for indoor environments with many repetitive patterns.

A better approach is to use constrained search. For each point $p(x, y)$ in frame $k-1$, only the point $p'(x'+u, y'+v)$ in frame k will be searched and evaluated, where (x', y') is the expected position of (x, y) in frame k. If a rectangular search

region is used, we should have $u \in [-\frac{w}{2}, \frac{w}{2}]$ and $v \in [-\frac{h}{2}, \frac{h}{2}]$, where w and h is the width and height of the rectangle. Alternatively, if a circular search region is used, we should have $\sqrt{u^2 + v^2} \leq r$, where r is the radius of the circle.

In our system, we use a circular search region. To determined the center of the circle, a constant-velocity model is used. For a feature point p_{2D} in $I_{l,k-1}$, a triangulation is performed to find its 3D coordinates p_{3D}. Based on the constant-velocity model, the reprojection of p_{3D} in frame $I_{l,k}$ is $p'_{2D} = KT_{k-1}p_{3D}$, where K is the intrinsic matrix and T_{k-1} is camera motion between $I_{l,k-2}$ and $I_{l,k-1}$. Here, p'_{2D} is used as the center of the search circle. The radius of the circle is pre-defined as $\frac{\min(n,m)}{10}$, where n and m is the number of rows and columns of the input image. Actually, we can determine the radius of the circle according to the probability $P(T_k = T_{k-1})$. In this way, the error propagation is also taken into account. During our testing, however, we found that the predefined radius works very well and is much more time-efficient compared with the probability approach.

In order to accelerate the searching process with the fixed region constraint, an indexing structure can be used, such as a multidimensional search tree (e.g., a k-d tree) or a hash table. In our system, a two dimensional k-d tree is used to conduct rapid searching within a circle.

5.1.2 Dissimilarity Score

In the process of building the correspondence matching system, different definitions of dissimilarity scores have been tested for different descriptors. In this section, the most commonly used dissimilarity scores are discussed and analysed.

5.1.2.1 SSD and NCC

For the standard corner detectors (e.g., Harris and FAST), an image patch around the corner is used as a descriptor. Between two image patches, the Sum of Squared

Differences (SSD) or Normalized Cross Correlation (NCC) measures are often used to calculate the dissimilarity score.

In SSD, the intensity differences are squared and aggregated within a square window:

$$\sum_{(i,j)\in W} (I_1(x+i, y+j) - I_2(x'+i, y'+j))^2.$$

Another very similar approach is the Sum of Absolute Differences (SAD). Instead of aggregating the square differences, SAD aggregates the absolute differences:

$$\sum_{(i,j)\in W} |I_1(x+i, y+j) - I_2(x'+i, y'+j)|.$$

SSD has higher computational complexity compared to SAD, as it involves numerous multiplication operations. However, SSD gives a larger penalty to large intensity differences, which generates fewer bad correspondences than SAD.

NCC is even more complex than both the SAD and SSD algorithms as it involves numerous multiplication, division, and square root operations:

$$\frac{\sum_{(i,j)} I_1(x+i, y+j) I_2(x'+i, y'+j)}{\sqrt{\sum_{(i,j)} I_1^2(x+i, y+j) \sum_{(i,j)} I_2^2(x'+i, y'+j)}}.$$

During the test, the NCC is actually more robust and accurate compared with SSD. Additionally, there exists much relevant literature investigating different approaches to implementing fast NCC calculations.

5.1.2.2 L^1-distance and L^2-distance

For the SIFT and SURF descriptors, L^1-distance and L^2-distance are used to evaluate the dissimilarity score. The use of L^1-distance versus L^2-distance has been heavily debated by the computer vision community, especially in the optimization perspective. A SIFT or SURF descriptor is actually a high-dimensional vector

with 128 elements, where each element represents the histogram bin value of the derivative. We will now briefly compare these two distance metrics:

- Similar to the SSD, the L^2-distance squares the error. In most circumstances, the resulting distance is much larger than the L^1-distance. It is easy to see that the L^2-distance is more sensitive to noise. In our tests, it generates fewer false positives and more true negatives.
- Similar to the SAD, the L^1-distance takes the absolute value of the error. It is less sensitive to noise and outliers. In our tests, it generates fewer true negatives and more false positives.

For the next step, a perspective-n-point algorithm with RANSAC is performed to generate an initial estimation of the camera motion. RANSAC is an outlier rejection algorithm, not an inlier selection algorithm, so it is necessary to keep the false positives low in order to achieve efficiency. In conclusion, we prefer the L^2-distance in our system.

5.1.2.3 Hamming Distance

Besides image patches, SIFT, and SURF descriptors, the ORB is also evaluated in our correspondence matching system. Similar to the BRIEF descriptor, the ORB descriptor is a vector with n elements, where each element stores the result of a binary test. Theoretically, there is no problem to use the L^1-distance or the L^2-distance to calculate the dissimilarity score between two ORB descriptors. However, the unnecessary square and absolute numeric operations tremendously increase the processing time. It is more appropriate to use the Hamming distance to calculate the dissimilarity score. The Hamming distance was originally developed in information theory to measure the minimum number of substitutions required to change one string into another. Here, we define the Hamming distance between two vectors of equal length as the number of positions at which the

(a) Harris detector with SSD

(b) SURF descriptor with L^2-distance

(c) ORB descriptor with Hamming distance

Figure 5.1: Correspondence matching results for different descriptors and distance metrics.

corresponding values are different. In our implementation, we construct the ORB descriptor as a fixed-length bit string. By using Exclusive-Or (XOR) combined with other bit operations, a rapid dissimilarity score calculation is performed.

Figure 5.1 shows the correspondence matching results for different descriptors and distance metrics. As can be observed from the comparison, unsurprisingly, the ORB and SURF outperform the Harris corner. In addition, the ORB and SURF both generates decent results. Considering its efficiency, the ORB with Hamming distance is used to perform correspondence matching in our system.

5.1.3 Matching Strategy

According to the search strategy and dissimilarity score discussed above, a dissimilarity score matrix D_k is constructed between frames $I_{k-1,l}$ and $I_{k,l}$, where $D_k(i, j)$

is the score between descriptor i in $I_{k-1,l}$ and the descriptor j in $I_{k,l}$. If one is not in the searching region of the other, $D(i,j) = \text{MAX\_VALUE}$. Based on the score matrix, for each descriptor in $I_{k-1,l}$, we find its best match in $I_{k,l}$. However, among those "best" matchs, there is large number of outliers. The matching strategy discussed in this section is used to eliminate the bad correspondences with low dissimilarity scores. In the literatures, there are two approaches that have been heavily discussed and compared—the ratio test and mutual consistency checking.

5.1.3.1 Ratio Test

As Lowe (2004) proposed in his landmark paper, the ratio test is used to reject ambiguous matches. For descriptor i in $I_{k-1,l}$, the best match p and the second best match q in $I_{k,l}$ are found. As the name suggests, the pair i and p is accepted as a correspondence only if $\frac{D(i,p)}{D(i,q)} < T$. For descriptor i in $I_{k-1,l}$, the algorithm involves a full scan of row i. Additionally, the ratio test for each descriptor is both task- and memory-independent. It is perfect to use a GPU thread to perform the ratio test for each descriptor in order to accelerate the matching process.

5.1.3.2 Mutual Consistency Check

To decide which "best" match to accept, an old but powerful trick, namely mutual consistency check, is usually involved. Given the score matrix D_k, for descriptor i in $I_{k-1,l}$, the best match is descriptor j in $I_{k,l}$, where $D_k(i,j)$ is the minimum in row i. Similarly, we check whether the best match of descriptor j in $I_{k,l}$ is descriptor i in $I_{k-1,l}$. The pair i and j is accepted as a correspondence only if both conditions are true. For descriptor i in $I_{k-1,l}$, the mutual consistency check involves two full scans for both row i and column j. Because the elements storing column is not in the same memory block, the mutual consistency check is at least 2 times slower than the ratio test. During our evaluations, we found that the

mutual consistency check generates similar results as the ratio test. Hence, the ratio test is implemented in our system to reject bad correspondences with low dissimilarity scores.

5.2 Initial Pose Estimation

In our system, the camera motion T_k between frame $k-1$ and frame k is computed from the 3D-to-2D correspondences X_{k-1} and p_k. The set of 3D feature points X_{k-1} is obtained by triangulating between the stereo pair $I_{k-1,l}$ and $I_{k-1,r}$. The set of 2D feature points p_k is detected in $I_{k,l}$. The 3D-to-2D correspondences are generated through the 2D correspondence matching between $I_{k-1,l}$ and $I_{k,l}$. As previously pointed out, motion estimation from 3D-to-2D correspondences is more accurate than from 3D-to-3D correspondences. Unlike the 3D-to-3D method, the camera motion is calculated by minimizing the reprojection error in the 3D-to-2D method. Given the fact that the baseline between two continuous frames is relatively small, the uncertainty in depth does not cause much change in the reprojected image position. Compared with the 3D-to-3D method, the effect of error propagation is actually reduced. Additionally, compared with 2D-to-2D method, there is no scale ambiguity problem for the 3D-to-2D method.

5.2.1 Perspective-N-Point Algorithm

The general formulation in the 3D-to-2D method is to find the T_k that minimizes the reprojection error

$$\arg\min_{T_k} \sum_i ||p_k^i - \hat{p}_{k-1}^i||^2,$$

where $\hat{p}_{k-1}^i$ is the reprojection of the 3D point X_{k-1}^i into the image I_k with the transformation T_k. This problem is known as Perspective from n Points (PnP). Basically, it calculates the extrinsic camera according to the 3D-to-2D correspon-

dences. There are many different solutions to it in the literature.

In order to derive the coordinates of the reprojection point, we assume that (u, v) is the reprojection of (x, y, z) according transformation T, where

$$u = \frac{T_{11}x + T_{12}y + T_{13}z + T_{14}}{T_{31}x + T_{32}y + T_{33}z + T_{34}},$$
$$v = \frac{T_{21}x + T_{22}y + T_{23}z + T_{24}}{T_{31}x + T_{32}y + T_{33}z + T_{34}}.$$

Note that each 3D-to-2D correspondence provides 2 equations. Inside T, there are 6 degrees of freedom (DOFs), with the rotation matrix and translation vector providing 3 DOFs each. As the result, the minimal-case solution involves three 3D-to-2D correspondences, which is known as the P3P algorithm (Kneip et al., 2011). The P3P algorithm involves the law of cosines and a large number of elimination steps, which we will not discuss in this thesis. Instead, a simple and straightforward P6P solution will be introduced.

For simplicity, we assume that the 2D coordinates are already normalized by K^{-1}. By stacking the constraints of six point-correspondences, a linear system of equation in the form of $AP = 0$ is constructed:

$$\begin{bmatrix} 0 & 0 & 0 & 0 & -x_1 & -y_1 & -z_1 & -1 & x_1v_1 & y_1v_1 & z_1v_1 & v_1 \\ x_1 & y_1 & z_1 & 1 & 0 & 0 & 0 & 0 & -x_1u_1 & -y_1u_1 & -z_1u_1 & -y_1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & -x_6 & -y_6 & -z_6 & -1 & x_6v_6 & y_6v_6 & z_6v_6 & v_6 \\ x_6 & y_6 & z_6 & 1 & 0 & 0 & 0 & 0 & -x_6u_6 & -y_6u_6 & -z_6u_6 & -y_6 \end{bmatrix} \begin{bmatrix} T_{11} \\ T_{12} \\ T_{13} \\ T_{14} \\ T_{21} \\ T_{22} \\ T_{23} \\ T_{24} \\ T_{31} \\ T_{32} \\ T_{33} \\ T_{34} \end{bmatrix} = 0.$$

The entries of T can be computed from the nullvector of A, which can be ob-

tained using SVD. The rotation and translation parts can be easily extracted from $P = [R|t]$. The rotation matrix obtained from the last step is not necessarily orthonormal. A Gram-Schmidt process can be applied to convert R to an orthonormal matrix. In the end of a general approach, both R and t can be refined by nonlinear optimization of the reprojection error.

The 3D-to-2D motion estimation assumes that the 2D points only come from one camera, the left camera in our implementation. Obviously, it is desirable to make use of feature points from both cameras. However, by conducting correspondence matching for both images and applying the PnP algorithm to two sets of correspondences, the precision does not improve as we expected. The processing time for motion estimation is almost doubled. Hence, we only use the left camera to generate 3D-to-2D correspondences in our system.

5.2.2 RANSAC

After applying the constrained searching strategy and ratio test matching strategy, the resulting matched points still contain a considerable number of outliers, which are mismatched points. There are many possible reasons that could cause this situation; e.g., image noise, occlusions, blur, and changes in viewpoint. For example, the feature transformation implemented in our system is the ORB detector and descriptor. In principle it is rotation-invariant and scale-invariant, but both rotation and scale is approximated in discretized space. Additionally, it is well-known that changes of viewpoint are actually a perspective transformation of the generated image. Here, the correspondence matching technique assumes a simple transformation with pure rotation, scaling, and translation, which is just a mathematical model that approximates the more complex reality. Moreover, throughout the most relevant literature, outlier rejection is the most delicate task in a VO system. Next, we briefly discuss the Random Sample Consensus (RANSAC) method (Fischler and Bolles, 1981), which we use in our system.

RANSAC is an iterative method to estimate the parameters of a mathematical model from a set of observed data that are contaminated by outliers. The basic idea is to compute the model hypothesis from a set of randomly sampled data and then all other data are tested against the generated hypothesis. The hypothesis that has the highest consensus is selected as the solution. It is similar to the voting scheme used in elections and decision making. The implementation of this voting scheme is based on two assumptions:

1. the noisy features or outliers will not vote consistently for any single model;
2. there are enough inliers to agree on a good model.

As mentioned in the Section 5.1, the RANSAC is an outlier rejection algorithm, not an inlier selection algorithm. It is necessary to have a sufficient number of inliers to make it work. That is the main reason that we conduct the constrained search and ratio tests to initially reduce the percentage of false positives among the matched points.

The outline of RANSAC method implemented in our system is as follows:

Input: A is the set of n correspondences, d is reprojection threshold, k is the maximum number of iterations.

1.1) Randomly select a sample of s points from set A, where s is the minimal number of points to estimate camera motion with respect to a specific algorithm; e.g., $s = 6$ for P6P and $s = 3$ for P3P.

1.2) Compute the $[R|t]$ based the perspective-n-point algorithm.

1.3) According to $[R|t]$, calculate the reprojection error for all other points in set A. For a point p, if its reprojection error $\epsilon(p) < d$, then p is added into the inlier set B. Store the inlier set B.

1.4) Repeat from **1.1** until the maximum number of iterations k is reached.

2) The set B' with maximum number of inlier is chosen as a solution.

3) The model is further improved by optimization against all inliers in set B'.

From the above outline, it is easy to see that RANSAC is a probabilistic algorithm, which produces a "good" result only with a certain probability. The probability increases as more iterations are conducted. Next, we discribe the method for choosing k to ensure the a reasonable result is obtained.

We set ω as the probability of choosing an inlier each time a single point is selected, which can be formulated as

$$\omega = \frac{\#\ inliers\ in\ A}{\#\ points\ in\ A}.$$

Given s as the minimum number of points needed to estimate a model, ω^s is the probability that all s points are inliers and $1-\omega^s$ is the probability that at least one of s points is an outlier. If one or more points in the s selected points are outliers, a bad model should be generated by the motion estimation algorithm. Let p be the probability that a good model is generated by the RANSAC algorithm. In other words, among the k iterations, there is at least one iteration that s selected points are all inliers. The probability that the algorithm never selects a set of s inliers must be same as $1-p$, which means $1-p = (1-\omega^s)^k$. Taking the logarithm of both sides of the equation will lead to

$$k = \frac{\log(1-p)}{\log(1-\omega^s)}.$$

As can be observed from the above equation, for a given probability p, parameters ω and s are the two main parameters that might affect the maximum number of iterations k. As ω decreases or s increases, k will be tremendously increased. In order to keep ω as high as possible, ORB is used to perform feature transformation, and the constrained search and ratio test are conducted in correspondence

matching. With respect to s, the minimum number of points needed to conduct motion estimation is 3. As the result, we use the P3P method to obtain $[R|t]$ based on three 3D-to-2D correspondences. In our implementation, by setting $d = 1.2$, we found that it is unnecessary to perform optimization for all inliers, which is a time-consuming process.

5.2.3 Motion Threshold

Our tests revealed that none of the triangulation methods are perfect. For all the triangulation methods tested in our system, the average reprojection error is about 0.9 pixels. In other words, for a stereo correspondence pair x_l and x_r, its corresponding 3D position X can be generated through triangulation. Given the fact that X is in the same coordinate system as the left camera, if we try to reproject it into the original left image, the new position is

$$x'_l = K[I|0]X.$$

Ideally, we should have $x_l = x'_l$. However, in reality, we have

$$||x_l - x'_l|| \approx 0.9$$

Actually, the reprojection error is relatively small, but it would lead to unfavourable drifting over time. If the relative distance between correspondences has the same scale as the reprojection error, the motion generated by the PnP algorithm can be biased. This situation happens frequently when the camera motion is small and the frame rate is high. Because there are no assumptions regarding the camera motion, a solution must be developed to deal with this "fake" motion. There are two possible solutions:

1. Optimize the position of the 3D point cloud. There are two sources of

error, the imprecise triangulation method and imprecise 2D coordinates. To improve the precision of the 2D coordinates, sub-pixel optimization is applied. To improve the precision of the triangulation method, for each pair x_l and X, an non-linear optimization is conducted with respect to the reprojection error.

2. Perform the motion estimation only for the key frames. Between two continuous key frames, the average distance $\bar{d}$ between the 2D correspondences and the average reprojection error ϵ must satisfy the condition $\frac{\bar{d}}{\epsilon} > T$, where T is the threshold value.

In our tests, we found that the first approach does not work well, where only about 7%–13% reprojection error can be reduced in this process. By analyzing our system, we found that there are two main reasons:

1. The baseline of our stereo camera is 14 cm, which is relatively large. Small changes in 2D position brought by sub-pixel optimization do not cause much optimization in the 3D coordinate for the large baseline.

2. The 3D coordinate obtained by linear triangulation is already optimal in the least squares sense. It is difficult to make large improvements by the subsequent non-linear optimization.

Additionally, for each frame, the optimization process takes about 5 ms. Surprisingly, the second approach works perfectly. By setting $T = 6$, most of the drifting can be eliminated. Moreover, when the camera motion is very small (e.g., hand shaking or normal vibration), instead of performing motion estimation, our system claims that the camera is unmoved. This saves much computation and generates relatively smooth camera trajectories.

CHAPTER 6

Motion Optimization

In a general VO system, individual transformations T_k are concatenated to form the current camera pose P_k. Each camera motion T_k is independently generated by conducting the PnP algorithm for a set of 3D-to-2D correspondences between I_{k-1} and I_k. As a result, each of these transformations T_k has certain range of uncertainty, and the uncertainty of the camera pose P_k depends on the uncertainty of P_{k-1} and T_k. In the following, the propagation of uncertainty is briefly introduced.

Before discussing this problem, it is beneficial to first discuss how uncertainty propagates in general linear and non-linear systems:

- Given a linear system $y = Ax$, let us assume C_x is the covariance matrix of x, where

  $$C_x(i,j) = \text{cov}(x_i, x_j).$$

  It is easy to derive that

  $$\begin{aligned} C_{Ax}(i,j) &= \text{cov}(\sum_{q=1}^{n} A_{iq}x_q, \sum_{p=1}^{n} A_{jp}x_p) \\ &= \sum_{q=1}^{n}\sum_{p=1}^{n} A_{iq}A_{jp}\text{cov}(x_q, x_p). \end{aligned}$$

  As the result, $C_y = AC_xA^T$ is obtained.

- Given a non-linear system $y = f(x)$, the first-order Taylor approximation is

applied to convert it into a linear system

$$y = f(x_0) + J_{f(x_0)}(x - x_0),$$

where $J_{f(x_0)}$ is the Jacobian matrix of $f(x)$ at x_0. According to the formula derived for the linear system, we have $C_y = J_{f(x_0)} C_x J_{f(x_0)}^T$.

In the VO system, each camera pose P_k and transformation T_k can be represented by a six-element vector containing the translation (x, y, z) and rotation (ϕ, θ, ψ) in Euler angles. The current camera pose P_k is written as a function of P_{k-1} and T_k, where

$$P_k = f(P_{k-1}, T_k).$$

Let us assume that P_{k-1} has covariance matrix $C_{P_{k-1}}$ and Jacobian matrix $J_{P_{k-1}}$, and T_k has covariance matrix C_{T_k} and Jacobian matrix J_{T_k}. Therefore, the covariance matrix for P_k can be derived as follows:

$$\begin{aligned} C_{P_k} &= \begin{bmatrix} J_{P_{k-1}} & J_{T_k} \end{bmatrix} \begin{bmatrix} C_{P_{k-1}} & 0 \\ 0 & C_{T_k} \end{bmatrix} \begin{bmatrix} J_{P_{k-1}} \\ J_{T_k} \end{bmatrix} \\ &= J_{P_{k-1}} C_{P_{k-1}} J_{P_{k-1}} + J_{T_k} C_{T_k} J_{T_k}. \end{aligned}$$

As can be observed from this equation, the uncertainty of the camera pose is always increasing when transformations are concatenated. The continuous increase of uncertainty is called "drifting".

There are two standard techniques to reduce drifting over time—pose graph optimization and bundle adjustment. In order to construct a real-time VO system with less drifting, a windowed bundle adjustment is implemented in our system. To perform bundle adjustment, a consistent world coordinate must be constructed. The entire 3D point cloud is added into this world coordinate and linked with the corresponding 2D feature points in different key frames. Additionally, with

respect to the camera pose and camera motion, a pose graph is also necessary to accomplish bundle adjustment. In this graph, the camera poses are the nodes and the camera motion obtained from motion estimation are the edges between nodes. In the following sections, we discuss world coordinate construction and windowed bundle adjustment in detail.

6.1 Correspondence Tracking

In this section, we illustrate the method used to track and maintain the world map. The world map consists of a set of feature points located in a world coordinate frame W. For each point x_W in the world map, a 2D descriptor, a 3D position, and a set of corresponding 2D positions in different frames are maintained. In order to construct a consistent map, the descriptors generated in I_k must be compared with all points in the map, which are the descriptors generated in $\{I_0, \ldots, I_{k-1}\}$. For a point x_i^k detected in I_k, if it is matched with any point x_W in the map, we add frame index k and point index i into x_W; otherwise, it is created as a new world point and added into the map.

The above method is a general way to perform tracking and mapping. However, during our implementation, we find that comparing each new generated descriptor against all points in the map is time-consuming. As the result, our system only supports feature-point tracking between two continuous key frames.

The tracking algorithm uses a similar constrained searching strategy, as illustrated in Section 5.1. Instead of performing a forward matching, however, our system conducts a backward matching. For each 2D point x_i^k detected in I_k, its corresponding 3D point X_i^k is reprojected back into I_{k-1} according to the inverse of T_k, where

$$\hat{x}_i^k = KT_k^{-1}X_i^k.$$

Next, the tracking system performs a constrained search within a fixed circle. In

our implementation, given the fact that the motion estimation with RANSAC is relatively accurate, the radius of the searching circle is about 10 pixels, and the center is $\hat{x}_i^k$. For each point x_i^k, it matches with all the points detected in I_{k-1} within the circle. The point x_p^{k-1} with best score $s_{x_p^{k-1}}$ and the point x_q^{k-1} with second best score $s_{x_q^{k-1}}$ are recorded. Then, x_i^k and x_p^{k-1} are accepted as the same point if they pass the ratio test, namely

$$\frac{s_{x_p^{k-1}}}{s_{x_q^{k-1}}} < T,$$

where T is the threshold value. If x_i^k and x_p^{k-1} are accepted as the same point, the world point x_W for x_p^{k-1} is found and the frame index k and point index i are added into x_W; otherwise, a new world point x'_W is created for x_i^k.

The world map constructed above is able to accomplish the global bundle adjustment. However, for the windowed bundle adjustment, all the points in the world coordinate have to be searched to find the points inside the window. To accelerate this process, instead of constructing a global world map, a local world map is maintained. There is one more operation that must be added into the pipeline. Given the window size m, after finishing the backtracking for frame k, the old points which exist only in frame $k-m$ should be deleted. In the end of the general pipeline for the current frame k, all the points in the local world map are searched. For each world point, if frame index $k-m$ exists, we delete that index. If there are no other frame indexes after deletion, we claim that this world point appears only in or before frame $k-m$. Hence, we delete such world points.

6.2 Windowed Bundle Adjustment

Throughout the literatures of SLAM or SFM system, an off-line global bundle adjustment is conducted in the end of the processing pipeline. Given a set of 3D

points and camera poses, the bundle adjustment can be defined as the problem of simultaneously refining the 3D point cloud describing the scene geometry and the parameters of the camera poses, by minimizing the reprojection error between the observed and reprojected point positions. The reprojection error can be expressed as the sum of squares of a large number of non-linear, real-valued functions. More formally, assume that n 3D points are seen in m views and x_{ij} is the position of point i observed in view j. Let v_{ij} denote the binary variable, where $v_{ij} = 1$ if point i is visible in view j and 0 otherwise. In addition, assume that camera pose j is parametrized by a 6-element vector $P_j(u, v, w, \phi, \theta, \psi)$ and 3D world point i is represented as a 3-element vector $X_i(x, y, z)$. The reprojection error with respect to all 3D points and camera poses can be formulated as

$$\sum_{i=1}^{n}\sum_{j=1}^{m} v_{ij}||f(X_i, P_j), x_{ij}||^2,$$

where $f(X_i, P_j)$ is the reprojection position of point i in view j.

During our implementation and evaluation, we identified three main problems:

1. At each frame, optimizing all the 3D points and camera poses generated in previous stages—i.e., performing global bundle adjustment—is too time consuming to implement in real-time.

2. By taking both the 3D point cloud and camera poses into account, the dimensionality of the optimization is enormous adn the optimizer takes too many iterations to reach convergence.

3. By simply optimizing the total reprojection error, the result might be unfavourable, especially considering that the continuous tracking algorithm generates a considerable number of the outliers, namely mis-tracked points.

There are two standard methods to solve Problem 1. It is possible to perform only a global optimization at some key frames or in the end, or conduct a win-

dowed bundle adjustment. In our tests, the first approach did not work well. By performing only a global optimization at some key frames, a large range of uncertainty might already be accumulated, which becomes almost impossible to reduce to zero. Also, our VO system is designed for real-world applications, such as for use in wearable devices, robots, and automotive applications. As a result, real-time feedback is necessary. However, using the first approach, at the frame in which optimization is performed, the delay is unacceptably long.

Surprisingly, the second approach works well. At each frame k, a windowed bundle adjustment is conducted, where only the 3D points and camera poses generated within the window are considered. It is easy to observe that the windowed bundle adjustment at frame k does not cause considerable change of T_k and it is relatively independent of the motion estimation of frame $k+1$. Hence, we use two different threads to deal with the bundle adjustment at frame k and motion estimation at frame $k+1$, since there are no sequential dependencies between those two processes. In this way, both the time consumed by bundle adjustment and the delay of real-time feedback is shortened.

In order to solve Problem 2, an approach similar to pose-graph optimization is applied. In the windowed bundle adjustment we adapted, only the camera poses within the window are optimized and the 3D positions are viewed as constants. As mentioned in Section 4.2.1, much effort has been devoted to performing an unbiased stereo camera calibration. Given unbiased camera parameters, the 3D positions obtained from the triangulation should have enough precision. By only optimizing the camera poses, the dimensionality of the objective function is largely reduced. Using the Levenberg-Marquardt methods, the reprojection error normally converges after 20 iterations for window size $m = 6$.

A general optimization approach is used to deal with Problem 3, which applies a loss function to reduce the influence of the points with high reprojection error, usually the one corresponding to outliers. To associate a loss function with the

reprojection error, the loss function takes the reprojection error as input and the output of the loss function is treated as the ultimate error. The new objective function is constructed as

$$\sum_{i=1}^{n}\sum_{j=1}^{m} v_{ij}\rho(||f(X_i, P_j), x_{ij}||^2),$$

where $\rho(s)$ is the loss function. The loss functions tested in our system were the Huber and Cauchy loss functions. Before describing the details of our implementation, a general introduction of loss functions is discussed.

There is considerable literature investigating the effectiveness and robustness of different loss functions. Here, we list several basic but valuable properties that most reasonable loss functions $\rho(s)$ should have:

- $\rho(0) = 0$ and $\rho'(0) = 1$.
- $\rho'(s) < 1$ and $\rho''(s) < 0$ in the outlier region.

The constraint of 0 error guarantee that the loss function generates similar value as the original error measure for small value of residues. Moreover, clearly a good loss function should suppress the error in the outlier region to reduce its influence. The Huber function tested in our system is a classical loss function which has all the properties discussed above and is formulated as

$$\rho(s) = \begin{cases} s & s \leq 1; \\ 2\sqrt{s} - 1 & s > 1. \end{cases}$$

Note from this equation that the error s remains the same when it is smaller than or equal to 1; otherwise, the error s gets suppressed. The original Huber function sets the boundary between inlier and outlier region as 1. However, it is possible

to change the boundary value to a by adding a scaling factor $a > 0$, where

$$\rho(s, a) = a\rho(\frac{s}{a}).$$

The square root provides less suppression for large residues. To further increase the suppression, the Cauchy loss function

$$\rho(x) = \log(1 + s)$$

was also tested in our system. Like the Huber function, the boundary value between inlier and outlier region can be easily changed using the scale factor. During our implementation, we found that two loss functions perform similarly, but the Huber function slightly outperforms the Cauchy function. By plotting both functions, we find that the Cauchy function not only suppresses the error within the outlier region but also slightly reduces the error within the inlier region, which is not what we want. Consequently, we use the Huber function in our system to perform outlier region suppression. The scale factor used in our system is intuitively set to

$$a = \epsilon^2 * m,$$

where m is the window size for bundle adjustment and ϵ is the average reprojection error between two successive frames. The reason for ϵ^2 is that loss function takes the square of reprojection error as input.

CHAPTER 7

Experimental Evaluation

To evaluate the robustness and accuracy of our system, two sets of data were tested.

The first set mainly focuses on indoor environments, which are captured through a Fujifilm stereo digital camera as shown in Figure 7.1(a). With two separate 10-megapixel sensors, the camera is able to capture high quality stereo video with 1280×720-pixel resolution and 30 frames per second. In addition, the synchronized control of the two sensors and lenses produces a perfectly matched pair of video sequences.

The second dataset mainly focuses on outdoor environments, which are provided by the KITTI vision benchmark suite (Geiger et al., 2012). This odometry benchmark comprises 11 sequences with ground truth trajectories. The ground truth is constructed by a Velodyne laser scanner and a GPS localization system as

(a) (b)

Figure 7.1: Fijifilm stereo camera and KITTI experimental vehicle.

shown in Figure 7.1(b). For evaluation with ground truth, we calculate the translation and rotation error for all possible subsequences of length $(100, \cdots, 800)$ meters. During our test, we found that the ground truth error for very short subsequences is relatively large, which might bias the evaluation results. Therefore, short subsequences, such as 5 or 10 meters, are not evaluated in our system. Moreover, the errors are measured in percentages for translation and in degrees per meter for rotation.

In the following sections, we present the results of our experiments with both indoor and outdoor environments, and bundle adjustment with different window sizes.

7.1 Indoor and Outdoor Environments

Most VO systems reported in the literature are specially designed for only indoor or only outdoor environments. By taking advantage of week ambient illumination, most indoor VO systems are constructed based on depth cameras. With help from structured light, accurate depth maps can be obtained through semi-global matching algorithms. However, in outdoor scenarios, the structured light sources have to compete with sunlight, whose power is normally 2 to 5 orders of magnitude larger than the projected light. Naturally, the strong ambient illumination severely degrades the performance of structured-light-based techniques. In order to construct a VO system for both indoor and outdoor environments, we build our system with a stereo camera. However, without structured light, tracking in indoor environments is quite challenging. Compared with the outdoor environment, there are fewer distinguished feature points to track. Furthermore, repetitive patterns in the indoor environment can cause many problems for the correspondence matching system.

To demonstrate the robustness and accuracy of our system, evaluations were

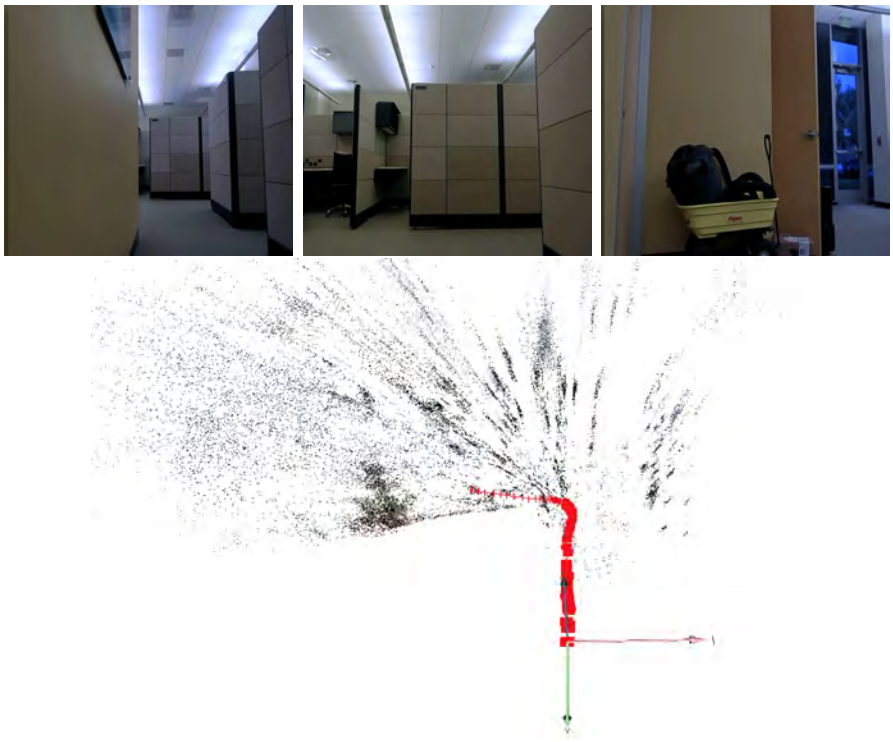


Figure 7.2: Indoor test scenarios, and resulted 3D trajectory and reconstructed

conducted for both the indoor and outdoor scenarios.

For the evaluation of indoor environment, the videos were captured using the Fujifilm stereo rigs. Without help from the GPS localization system, it is almost impossible to obtain accurate ground truth data for each frame.

Instead of evaluating the accuracy of the camera trajectory, the robustness is mainly evaluated in indoor scenarios. As shown in Figure 7.2, most of our tested videos are recorded in the environment with a few traceable features and massive repetitive patterns. As can be observed from the resulting 3D camera trajectory and scene reconstruction, our system is robust enough to handle this indoor environment.

To evaluate the performance of our VO system on outdoor environments, we employ the KITTI vision benchmark suite. With a Velodyne laser scanner and a GPS localization system, the error of the available ground truth is limited to 11

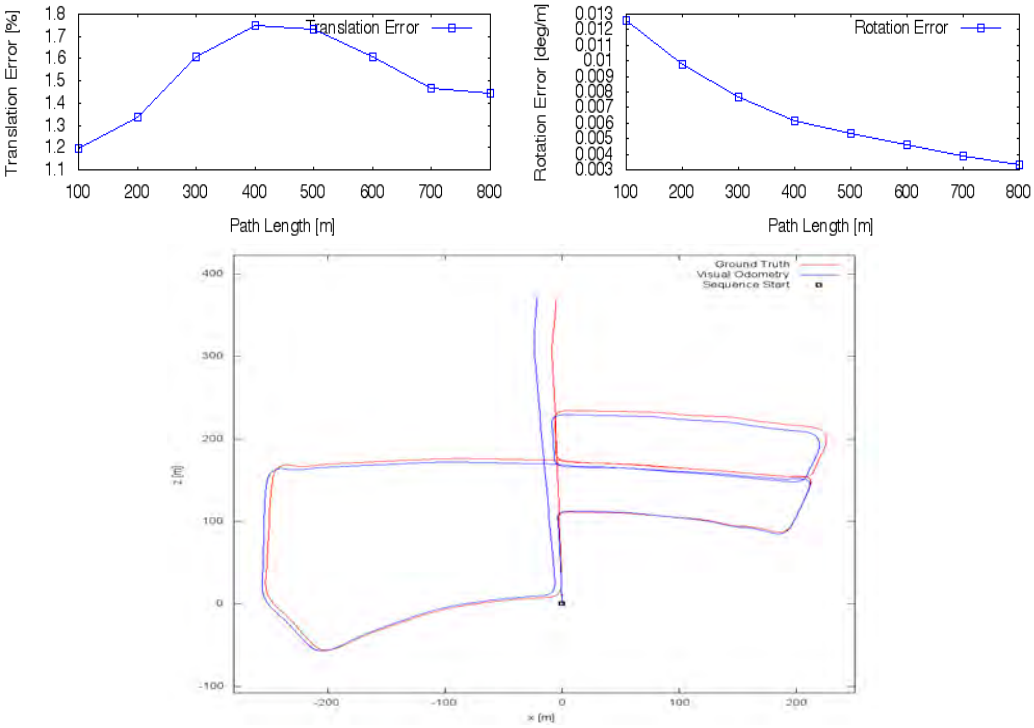


Figure 7.3: The evaluation plots for the 5th KITTI dataset.

centimeters. In addition, the datasets are captured by driving around the mid-size city of Karlsrushe, which includes diverse environments, e.g., rural area, highway, and forest.

For most of the 11 test datasets, our system is able to achieve surprisingly high accuracy, especially for videos involving small and medium travel lengths. As shown in Figure 7.3, for subsequences of length 100–800 meters, the range of translation errors is $[1.2\%, 1.7\%]$ and the range of rotation errors is $[0.03\,deg/m, 0.013\,deg/m]$.

In addition, our system exhibits decent performance for a test video involving a large travel length. As shown in Figure 7.4, with similar measurement, the range of translation errors is $[1.95\%, 2.35\%]$ and the range of rotation errors is within $[0.005\,deg/m, 0.014\,deg/m]$. Comparing to the performance for medium travel length, both translation and rotation errors are relatively worse. The three possible reasons are as follows:

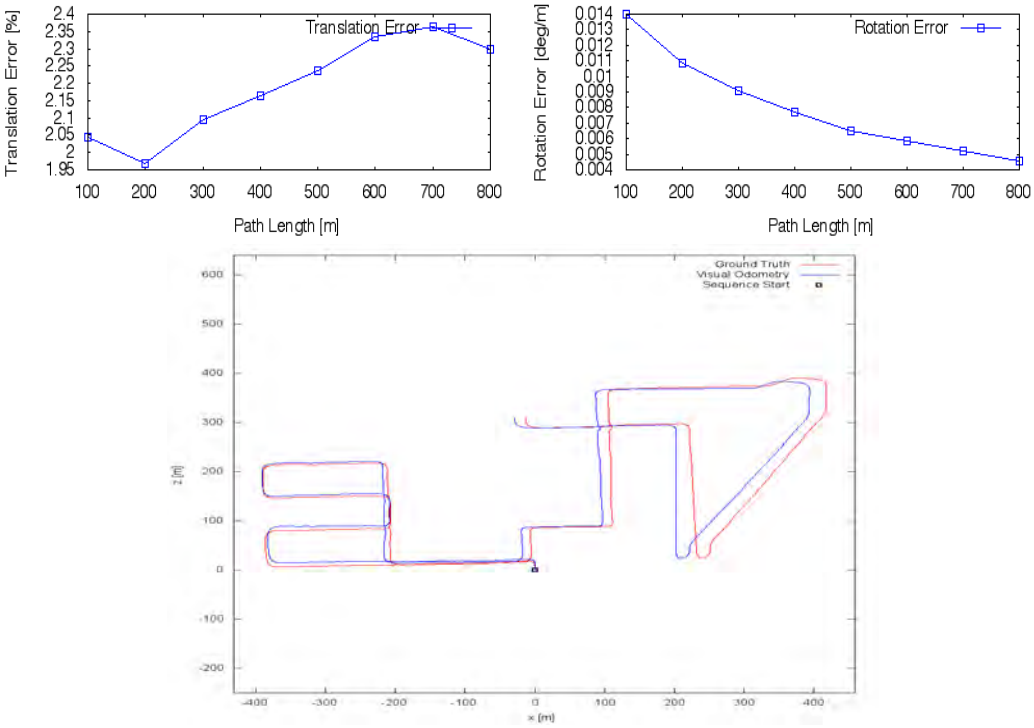


Figure 7.4: The evaluation plots for 8th KITTI dataset.

- As discussed previously, the uncertainty of the camera pose always increases when transformations are concatenated.
- For all the evaluations, the camera motions are optimized using bundle adjustment with a window size $m = 6$. This is only able to largely reduce the drifting within the local region of the trajectory.
- In order to increase efficiency, our system supports feature point tracking only between two successive key frames. Due to occlusion and change of viewpoint, for the same 3D points that appear in discontinuous key frames, bundle adjustment might not be able to reduce the drifting under such circumstances.

The further optimization with respect to large-scale environments will be discussed in the next chapter.

In summary, our VO system is able to achieve high accuracy and robustness

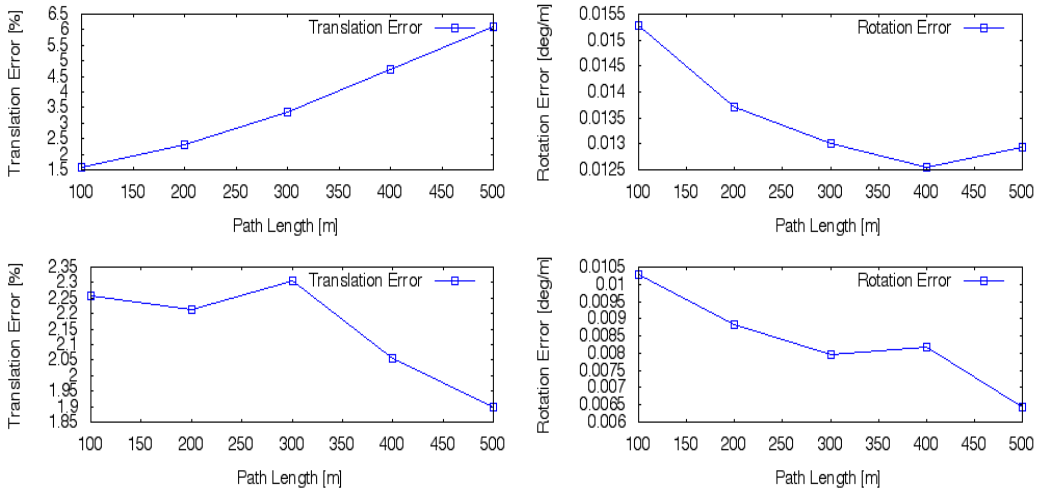


Figure 7.5: The evaluation plots for 3rd KITTI dataset with and without bundle adjustment.

for both indoor and outdoor environments. Among the 11 test cases provided by the KITTI vision benchmark, the average translation error is 2.09% and the average rotation error is 0.0067deg/m per 800 meters of travel.

7.2 Window Sizes of Bundle Adjustment

In our experiments, we find that the window size has a tremendous effect on bundle adjustment performance. However, throughout the recent literatures, there are only a few discussions about how to choose an appropriate or optimal window size. In this section, the performances of bundle adjustment with different window sizes is presented. First, it is beneficial to compare the errors with and without bundle adjustment.

In Figure 7.5, the first row of plots shows the translation and rotation errors without bundle adjustment. As can be observed from the plots, the translation drift keeps increasing as the length of measurement increases. Because the rotation error is measured in degrees per meter, it is compromised as the length of the subsequence increases. Actually, the rotation drift is quite obvious. The second

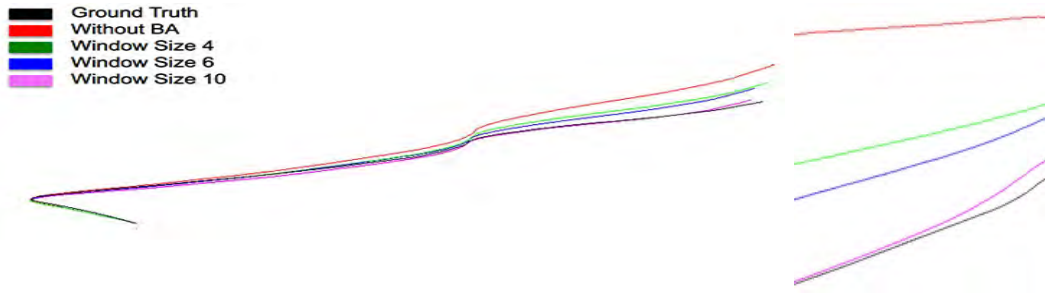


Figure 7.6: The camera trajectory for 4th KITTI dataset with the zoom-in details.

Figure 7.7: The camera trajectory for 4th KITTI dataset with the zoom-in details.

row of plots shows our evaluation results with bundle adjustment, where both the translation and rotation errors are highly reduced, especially for long subsequence length.

It is obvious that the bundle adjustment is able to mitigate the uncertainty propagation. In addition, it is valuable to investigate the performance of different window sizes.

Due to the constraint of real-time performance, three different window sizes, 4, 6, and 10, were mainly evaluated in our system. For most of the test cases, the drifting of both translation and rotation errors decreases as we increase the window size. As can be observed in Figure 7.6, comparing the one without bundle adjustment and with bundle adjustment of window size 4, the accuracy of the VO is greatly improved.

As expected, the improvement becomes less obvious as we continue increasing

the window size. Also, for some cases shown in the Figure 7.7, the performance of bundle adjustment with window size 10 is worse than the one with window size 4. For the continuous tracking algorithm, as the window size increases, the number of outliers involved in the optimization also increases. If there are a considerable number of outliers, even with a robust loss function, the resulted camera pose will be biased. Consequently, a bundle adjustment with window size 6 was implemented in our system.

CHAPTER 8

Conclusion

We will now conclude the thesis by reviewing our contributions and discussing interesting directions for future work.

8.1 Thesis Contributions

We have developed a functioning stereo VO system for precise position estimation and 3D scene reconstruction. Using novel scale-space features, an efficient feature matching algorithm, and windowed bundle adjustment, we obtain an extremely precise positioning system for use in both indoor and outdoor environments. By identifying the relatively independent relations between the different tasks, parallel computing techniques were used to achieve real-time performance for 30-frame-per-second video sequences with 1280×720-pixel resolution.

In addition to implementing a real-time and robust stereo VO system, we developed new techniques to tackle several classical problems. The "fake motion" problem caused by imperfect triangulation methods is solved by key frame-selection with a motion threshold. A Huber loss function with a scale factor is designed to reduce the influence of outliers during continuous tracking, which may generate biased optimization results for bundle adjustment.

Moreover, we discussed in depth the core components of our system. The efficiency and robustness of five corner and blob detectors and descriptors were analyzed. For the correspondence matching algorithm, the advantages and disad-

vantages of feature tracking and feature matching was discussed in detail. Several strategies, such as constrained search and the ratio test, were proposed to reduce the percentage of false positives in the correspondence matching. With respect to motion estimation, we compared the efficiency and accuracy of the classical 3D methods, namely, 2D-to-2D, 3D-to-2D, and 3D-to-3D.

8.2 Future Work

In our current system, the only input data are stereo video sequences. However, many VO applications involve multiple sensors. One area of improvement would be to integrate into our system the inputs from other sensors, e.g., IMUs and wheel odometry sensors. In addition, the camera motions generated through two successive stereo pairs are actually independent of each other, which could result in unrealistic and sudden changes. The Kalman filter can be integrated into our system to impose the constraint of continuous motion. A similar constratin can be imposed by performing sensor fusion with the data collected by IMUs.

Moreover, our systems supports feature tracking only between two successive key frames. In order to support feature tracking in discontinuous key frames, an exhaustive search of all world points must be done for new detected points. To accelerate the search process, an structured indexing of the world map, such as using a k-d tree or an octree, can be incorporated into our system.

Finally, bundle adjustment is too time consuming for reducing the drifting with large window sizes. It might be interesting to test a pose-graph optimization method, such as g2o (Kummerle et al., 2011).

Bibliography

Arun, K. S., Huang, T. S., and Blostein, S. D. (1987). Least-squares fitting of two 3-d point sets. In *IEEE Trans. Pattern Anal. Machine Intell.*, pages 698–700.

Bay, H., Tuytelaars, T., and Gool, L. V. (2006). Surf: Speeded up robust features. In *ECCV*, pages 404–417.

Berthold, K. P. H. and Brian, G. S. (1981). Determining optical flow. *ARTIFICAL INTELLIGENCE*, pages 185–203.

Corke, P., Strelow, D., and Singh, S. (2004). Omnidirectional visual odometry for a planetary rover. In *Prof. IEEE Int. Conf. Intelligent Robots and Systems*, pages 4007–4012.

Engel, J., Schöps, T., and Cremers, D. (2014). Lsd-slam: Large-scale direct monocular slam. In *ECCV*, pages 834–849. Springer.

Engel, J., Sturm, J., and Cremers, D. (2013). Semi-dense visual odometry for a monocular camera. In *Proc. IEEE Int. Conf. Computer Vision*, pages 1449–1456.

Fischler, M. A. and Bolles, R. C. (1981). Ransac sample consensus: A paradigm for model fitting with applications to image analysis and auto- mated cartography. In *Commun. ACM*, pages 381–395.

Fraundorfer, F. and Scaramuzza, D. (2011). Visual odometry: Part i - the first 30 years and fundamentals. *IEEE Robotics and Automation Magazine*.

Geiger, A., Lenz, P., and Urtasun, R. (2012). Are we ready for autonomous driving? the kitti vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition*.

Harris, C. and Pike, J. (1987). 3d positional integration from image sequences. In *Proc. Alvey Vision Conf.*, pages 233–236.

Hartley, R. and Zisserman, A. (2003). *Multiple view geometry in computer vision*. Cambridge university press.

Howard, A. (2008). Real-time stereo visual odometry for autonomous ground vehicles. In *Intelligent Robots and Systems*, pages 3946–3952. IEEE.

Kerl, C., Sturm, J., and Cremers, D. (2013). Robust odometry estimation for rgb-d cameras. In *Prof. IEEE Int. Conf. Intelligent Robots and Systems*, pages 3748–3754.

Kitt, B., Geiger, A., and Lategahn, H. (2010). Visual odometry based on stereo image sequences with ransac-based outlier rejection scheme. In *Intelligent Vehicles Symposium*, pages 486–492. IEEE.

Kneip, L., Scaramuzza, D., and Siegwart, R. (2011). A novel parametrization of the perspective-three-point problem for a direct computation of absolute camera position and orientation. In *Proc. Computer Vision and Pattern Recognition*, pages 2969–2976.

Konolige, K., Agrawal, M., and Sola, J. (2011). Large-scale visual odometry for rough terrain. In *Robotics Research*, pages 201–212. Springer.

Kummerle, R., Grisetti, G., Strasdat, H., Konolige, K., and Burgard, W. (2011). g 2 o: A general framework for graph optimization. In *Prof. IEEE Int. Conf. Robotics and Automation*, pages 3607–3613.

Longuet-Higgins, H. (1981). A computer algorithm for reconstructing a scene from two projections. *Nature*, pages 133–135.

Lowe, D. (2004). Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, pages 91–110.

Milella, A. and Siegwart, R. (2006). Stereo-based ego-motion estimation using pixel tracking and iterative closest point. In *Proc. IEEE Int. Conf. Computer Vision*, pages 21–21.

Nister, D. (2003). An efficient solution to the five-point relative pose problem. In *Proc. Computer Vision and Pattern Recognition*, pages 195–202.

Nister, D., Naroditsky, O., and Bergen, J. (2004). Visual odometry. In *Proc. Computer Vision and Pattern Recognition*, pages 652–659.

Rosten, E. and Drummond, T. (2006). Machine learning for high-speed corner detection. In *In European Conference on Computer Vision*, pages 430–443.

Rublee, E., Rabaud, V., Konolige, K., and Bradski, G. (2011). Orb: An efficient alternative to sift or surf. In *Proc. IEEE Int. Conf. Computer Vision*, pages 2564–2571.

Triggs, B., McLauchlan, P., Hartley, R., and A., F. (2000). Bundle adjustment a modern synthesis. In *Proc. IEEE Int. Conf. Computer Vision*, pages 298–372.

Yamaguchi, K., Kato, T., and Ninomiya, Y. (2006). Vehicle egomotion estimation and moving object detection using a monocular camera. In *Proc. IEEE Int. Conf. Pattern Recognition*, pages 610–613.