UNIVERSITY OF CALIFORNIA SAN DIEGO

Analysis of Geometry and Deep Learning-based Methods for Visual Odometry

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

in

Electrical Engineering (Signal and Image Processing)

by

You-Yi Jau

Committee in charge:

Professor Manmohan Krish Chandraker, Chair
Professor Nikolay A. Atanasov, Co-Chair
Professor Hao Su
Professor Nuno M. Vasconcelos

2020

The Thesis of You-Yi Jau is approved, and it is acceptable in quality and form for publication on microfilm and electronically:

Co-Chair

Chair

University of California San Diego

2020

DEDICATION

For my parents and my brother supporting my study and pursuit of knowledge.

EPIGRAPH

I think, therefore I am.

René Descartes

I'm a greater believer in luck, and I find the harder I work the more I have of it.

Thomas Jefferson

TABLE OF CONTENTS

Signature Page iii

Dedication iv

Epigraph v

Table of Contents vi

List of Figures ix

List of Tables x

Acknowledgements xi

Vita xiv

Abstract of the Thesis xv

Chapter 1 Introduction 1
1.1 Overview 1
1.2 Camera Pose Estimation 2
1.3 Visual Odometry (VO) 2
1.4 Simultaneous Localization and Mapping (SLAM) 3
1.5 Thesis Outline 3
1.6 Keywords 4
1.7 Terminology 4

Chapter 2 Background and Related Work 5
2.1 Overview on Visual Odometry 5
2.1.1 History 5
2.1.2 Problem Formulation 6
2.1.3 Pose Representation 8
2.1.4 Camera Model and Calibration 9
2.1.5 Feature Extraction and Matching 10
2.1.6 Motion Estimation 11
2.1.7 Outlier Rejection 14
2.1.8 Triangulation and Keyframe Selection 14
2.1.9 Bundle Adjustment (BA) 15
2.2 Geometry-Based Method 16
2.3 Deep Learning-Based method 17
2.4 Conclusion 18

Chapter 3 Geometry-Based Method for Visual Odometry 20
3.1 Overview to ORB-SLAM 20

3.1.1 Tracking . . . 22
3.1.2 Mapping . . . 22
3.1.3 Loop Closure . . . 23
3.2 Experiments . . . 23
3.2.1 Datasets . . . 24
3.2.2 Evaluation Metrics . . . 24
3.3 Discussion . . . 27
3.3.1 Initialization and Relocalization . . . 27
3.3.2 Outlier Rejection . . . 27
3.3.3 Keyframe-Based System . . . 27
3.3.4 Long Feature Tracking . . . 28
3.3.5 Robust Bundle Adjustment . . . 28
3.4 Conclusion . . . 29

Chapter 4 Deep Keypoint-Based Camera Pose Estimation with Geometric Constraints 31
4.1 Introduction . . . 31
4.2 Method . . . 33
4.2.1 Overview . . . 33
4.2.2 Feature Extraction (FE) . . . 34
4.2.3 Pose Estimation (PE) . . . 36
4.2.4 Training Process . . . 38
4.2.5 Network Structure . . . 39
4.2.6 Parameter Setting . . . 40
4.3 Experiments . . . 40
4.3.1 Datasets . . . 41
4.3.2 Relative Pose Estimation . . . 42
4.3.3 SuperPoint Correspondence Estimation . . . 45
4.4 Conclusion . . . 47

Chapter 5 Deep Learning-based Method for Visual Odometry . . . 49
5.1 Overview to SC-SfMLearner . . . 49
5.1.1 Pipeline . . . 49
5.1.2 Network design . . . 50
5.1.3 Loss functions . . . 52
5.2 Experiments of SC-SfMLearner on Various Datasets . . . 53
5.2.1 Implementation Details . . . 54
5.2.2 Domain Gap . . . 54
5.2.3 Overfitting Issues . . . 55
5.2.4 Relative Pose Prediction . . . 55
5.3 Future work . . . 56
5.3.1 Optimization . . . 58
5.3.2 Keyframe-Based System . . . 58
5.3.3 Long Feature Tracking . . . 58
5.4 Conclusion . . . 59

Chapter 6 Conclusion 63
6.1 Future Direction 64

Bibliography 65

LIST OF FIGURES

Figure 2.1. Camera projection model. 7

Figure 2.2. Basic pipeline of visual odometry. 8

Figure 2.3. Geometric bundle adjustment. 16

Figure 3.1. Overview of ORB-SLAM. 21

Figure 3.2. Qualitative VO results on KITTI dataset using ORB-SLAM2-VO. 26

Figure 3.3. Qualitative VO results on EuRoC dataset using ORB-SLAM2-VO. 30

Figure 4.1. Overview of the system 32

Figure 4.2. Network structure of our feature extraction (FE) and pose estimation(PE) modules. 34

Figure 4.3. Pose estimation comparison. 41

Figure 4.4. Failure cases of pose estimation. 46

Figure 4.5. Change of keypoint distribution after end-to-end training. 46

Figure 5.1. SC-SfMLearner pipeline 50

Figure 5.2. SC-SfMLearner prediction on KITTI sequence 09 51

Figure 5.3. Comparison of qualitative RPE results on the EuRoC sequence using geometry-based or deep learning-based methods. 56

Figure 5.4. Qualitative VO results for KITTI model on KITTI dataset for SC-SfM Learner. 57

Figure 5.5. Qualitative VO results for KITTI model on EuRoC dataset for SC-SfM Learner 60

Figure 5.6. Qualitative VO results for EuRoC model on KITTI dataset for SC-SfM Learner. 61

Figure 5.7. Qualitative VO results for EuRoC model on EuRoC dataset for SC-SfM Learner. 62

LIST OF TABLES

Table 3.1. Quantitative result of ORB-SLAM-VO on KITTI dataset. 25

Table 3.2. Quantitative result of ORB-SLAM-VO on EuRoc dataset. 25

Table 4.1. The reference table for modules and losses trained for experiments. 36

Table 4.2. Parameters for clamping pose-loss. 40

Table 4.3. Comparison of pose estimation for learning-based KITTI models on KITTI dataset 43

Table 4.4. Comparison of pose estimation for SIFT-based KITTI models on KITTI dataset. 44

Table 4.5. Comparison of pose estimation for learning-based KITTI models on Apollo dataset. 44

Table 4.6. Comparison of pose estimation for SIFT-based KITTI models on Apollo dataset. 45

Table 4.7. Superpoint evaluation. 47

Table 5.1. PoseNet architecture. 50

Table 5.2. DepthNet architecture. 52

Table 5.3. Quantitative result of SC-SfMLearner on KITTI dataset. 56

Table 5.4. Quantitative result of SC-SfMLearner on EuRoC dataset. 56

ACKNOWLEDGEMENTS

I would like to acknowledge Professor Manmohan Chandraker for his support as the chair of my committee. He advised and inspired me on my research, which led to the publication and this thesis. The meetings and encouragement are invaluable for my master's study.

I would like to acknowledge Professor Hao Su for his support of research. He shared his knowledge and thoughts and pushed me toward the deep understanding of knowledge. The enlightenment from him and the discussion with the lab mates led to the thesis.

I would like to acknowledge Professor Nikolay Atanasov for his support as the co-chair of my committee. He advised me during my service as a TA in his class. The cooperation was wonderful and the knowledge supported me as the foundation for further research.

I would like to acknowledge Professor Nuno Vasconcelos for his support as a committee member. His lecture in my first quarter inspired me of the theoretical perspective of my research.

I would like to acknowledge Professor Mohan M. Trivedi for his support for TAship. The experiences during the two classes were wonderful and precious.

I would like to acknowledge Professor Shao-Yi Chien and Dr. Po-Chen Wu for the inspiration of my interest in the field.

I would like to acknowledge Dr. Wei-Chao Chen and Dr. Trista Chen for the advice during my internship in 2018 and afterwards. The experience had a great impact on my mindset on research.

I would like to acknowledge Rui Zhu for working together on the project. The cooperation and discussion were essential to my research and engineering capabilities. The project contributes to the key part of this thesis.

I would like to acknowledge Bowen Zhang for our cooperation as TAs. It was fantastic working with him. His attitude and accountability inspired my way of work.

I would like to acknowledge Giayuan Gu for his thoughts and suggestions on my research. He shared his knowledge and codes, which helped on my mindset and deliverables.

I would like to acknowledge Ishit Mehta for his insightful opinions. The discussion about

the technical details was critical for my work. The discussion about the world is motivating.

I would like to acknowledge Stephanie Mathew for logistical support. She supported my search for TAship, degree planning, and thesis planning in a detailed-oriented way.

I would like to acknowledge Li-An (Leon) Yang for his encouragement. The motivation came from tennis retreat, pot luck, or hangouts, which helped me destress during busy seasons.

I would like to acknowledge Fred Lin for his generosity during my early time in San Diego. I would like to acknowledge Joseph Li-Yuan Chiang and Vanessa Chang for their support when I just came here. The grocery shopping rides were essential for my living.

I would like to acknowledge Keng-chi Chang and Norton Cheng for their support. The potlucks, jam sessions (with Sheng-Yong), board games, and grocery shopping made my life wonderful.

I would like to acknowledge Sheng-Yong Niu for his inspiration on my vision. The discussion on past experience, worldwide issues, and future plan is fruitful for fostering my mind.

I would like to acknowledge Shuang Liu for his support as my neighbor in the lab. The swimming retreats were refreshing and memorable.

I would like to acknowledge Hsuan-Lin (Charlene) Her for her companionship. Her encouragement on my life, research, and career made me become better. The time jogging, working, or hanging out together was fantastic.

I would like to acknowledge my family, my mother, father and brother, who supported my Master's study, both financially and mentally. The support allowed me to pursue my dream and enjoy my life.

In Ch. 2, in part, has been submitted for publication of the material as it may appear in Conference on Intelligent Robots and Systems (IROS), 2020. You-Yi Jau, Rui Zhu, Hao Su, Manmohan Chandraker. The thesis author was the primary investigator and author of this paper.

In Ch. 4, in full, has been submitted for publication of the material as it may appear in Conference on Intelligent Robots and Systems (IROS), 2020. You-Yi Jau, Rui Zhu, Hao Su,

Manmohan Chandraker. The thesis author was the primary investigator and author of this paper.

In Ch. 5, in part is currently being prepared for submission for publication of the material. You-Yi Jau, Manmohan Chandraker. The thesis author was the primary investigator and author of this paper.

VITA

2018	Bachelor of Science, National Taiwan University (NTU), Taiwan
2019	Research Assistant, Department of Computer Science and Engineering
2019–2020	Teaching Assistant, Department of Electrical and Computer Engineering
2020	Teaching Assistant, Department of Computer Science and Engineering
2020	Master of Science, University of California San Diego

PUBLICATIONS

You-Yi Jau*, Rui Zhu*, Hao Su, Manmohan Chandraker. ”Deep Keypoint-based Camera Pose Estimation with Geometric Constraints”. In Submitted to Conference on Intelligent Robots and Systems (IROS), 2020. Under review.

FIELDS OF STUDY

Major Field: Electrical Engineering (Signal and Image Processing)

Studies in Computer Science and Engineering
Professors Manmohan Chandraker and Professor Hao Su

ABSTRACT OF THE THESIS

Analysis of Geometry and Deep Learning-based Methods for Visual Odometry

by

You-Yi Jau

Master of Science in Electrical Engineering (Signal and Image Processing)

University of California San Diego, 2020

Professor Manmohan Krish Chandraker, Chair
Professor Nikolay A. Atanasov, Co-Chair

In the fields of VR, AR, and autonomous driving, it is critical to track the accurate location of an agent using cameras. This thesis dives into the problem of using ordered image sequences for localization, known as visual odometry. The lines of research can be categorized into two main group, geometry-based methods and deep learning-based methods. Geometry-based methods have been explored for over a decade, which yield robust real-time prediction in both outdoor and indoor environments. In recent years, deep learning-based methods show the potential to outperform geometry-based methods in localization. However, they are yet to be proved as accurate in variety of scenes. In this thesis, we first dive into a complete geometry-

based pipeline and point out the key factors for a robust system. Second, we design a deep learning-based camera pose estimation pipeline with geometric constraints, which generalizes better than the learning-based baselines under two datasets. In the end, we explore the possibility of enhancing deep learning prediction based on geometric optimization. The thesis plots a road for combining both methods by thorough comparison. By leveraging the advantages of geometry-based and learning-based methods, the future of a robust visual odometry system can be anticipated.

Chapter 1

Introduction

1.1 Overview

Visual odometry has been widely researched. The problem is defined as estimating the camera pose from ordered images. Applications, including virtual reality (VR), augmented reality (AR) devices, robots, and self-driving vehicles, usually require real-time localization. Traditionally, this problem is solved by geometry-based methods, where the geometry between multiple views of the same object is expressed into mathematical forms, and further enforced through linear algebra or optimization. However, geometry-based methods have been challenged by emerging deep learning-based approaches. Deep learning-based methods usually consist of three parts, models, datasets and loss functions. The prediction from the model is regressed by the loss functions, where the labels can come from the dataset. With carefully designed labels or constraints, the model can learn the patterns from data.

The deep learning-based methods have several advantages. First, the model can learn complex information from data. This has been proven from experiments, *e.g.* AlphaGo, where the machines defeat humans using a large amount of data and calculation. Second, the model can be optimized end-to-end, so as to achieve better performance. Deep learning models are versatile to predict from high-level semantics, *e.g.* objects, to low-level information, *e.g.* keypoints. Models can be shared or concatenated for end-to-end optimization, as long as the gradients can flow back to update the models.

The geometry-based methods also edge over deep learning-based methods in multiple ways. First of all, geometry-based methods can be interpreted through mathematical formulation. This property helps humans understand the capabilities and risks under the methods, whereas deep networks are mostly black boxes. Secondly, the concept for geometric constraints is incorporated into the methods, which leads to better generalization ability across different environments. For example, the same algorithm can work indoors for VR devices and outdoors for self-driving cars.

With the advantages of both methods, this thesis aims to analyze the pros and cons of geometry-based and deep learning-based methods across different scenes. To evaluate the algorithms, the predicted localization for each frame can be accumulated into a trajectory, which is compared with the ground truth from the dataset. The thorough qualitative results and quantitative results are shown in the comparison.

To understand the topics, we introduce the concept and applications of camera pose estimation, visual odometry and SLAM in the following sections. The three topics look at a problem from different perspectives.

1.2 Camera Pose Estimation

Camera pose estimation is a building block for the general Structure From Motion (SFM) problems. With the input of ordered or unordered images, the pipeline solves for the camera poses as well as the 3D structures [6]. In this thesis, we focus on the problem with the input of ordered images, namely frames from videos.

1.3 Visual Odometry (VO)

Visual odometry applies the idea of camera pose estimation to a video sequence. The typical output is a camera trajectory w.r.t. the world coordinate, which shows how the camera moves from frame to frame. This is the key technique for VR and AR devices because the

animation should be projected based on the position of the user. In the field of self-driving cars, visual odometry has been utilized to track the real-time positions, and further been incorporated with the motion planning algorithms.

In the applications, accuracy and response time are important factors. One obvious question is the availability of other sensors, *i.e.* GPS or IMUs. GPS can give the position w.r.t. the world coordinate, but the error is up to meters and is unresponsive in indoor environment [1]. IMU measures the relative rotation and translation from acceleration and angular velocity. However, due to drifting issues, it is not suitable for long-term prediction. On the other hand, visual cues are useful for short-term and long-term prediction, where a camera for a robot is the analogy of an eye for a human being. The details of the visual odometry pipeline are introduced in Ch. 2.

1.4 Simultaneous Localization and Mapping (SLAM)

SLAM systems demonstrate a complete pipeline for both localization and mapping [20, 47]. Localization and mapping are a chicken-and-egg problem, which is usually optimized iteratively. The differences between visual odometry and SLAM systems include mapping and loop closure. Mapping can transform the 2D points into 3D points, and add the points into a global map. By emphasizing the maintenance of a global map, the drifting problem can be reduced. In loop closure, previous frames are recorded in order to detect a loop in the trajectory. When the loop is detected, the trajectory within the loop can be optimized all together.

1.5 Thesis Outline

The chapters are organized as follows. Ch. 2 introduces the basic pipeline for visual odometry and recent work for geometry-based and deep learning-based methods. Ch. 3 describes the analysis of geometry-based methods across datasets, where ORB-SLAM2 [47, 48] is tested on KITTI [26] and EuRoC [13] datasets. In Ch. 4, we introduce an end-to-end method for camera

pose estimation with the advantages of geometric constraints, which is evaluated on KITTI [26] and ApolloScape [34] datasets. In Ch. 5, we identify the limitation of deep learning-based method for visual odometry and the potential to increase the robustness. Ch. 6 summarizes the work and points out future directions.

1.6 Keywords

Computer vision, deep learning, robotics, camera pose estimation, Structure From Motion (SFM), Simultaneous Localization and Mapping (SLAM), Visual Odometry (VO), multi-view geometry, optimization.

1.7 Terminology

Structure from Motion (SfM), Simultaneous Localization and Mapping (SLAM), Visual Odometry (VO), virtual reality (VR), augmented reality (AR), Global Positioning System (GPS), Inertial Measurement Unit (IMU), 2-dimension (2D), 3-dimension (3D), singular value decomposition (SVD), Root Mean Square Error (RMSE), Micro Aerial Vehicle (MAV), RANdom SAmple Consensus (RANSAC), Levenberg–Marquardt algorithm (LM algorithm), Bundle Adjustment (BA), Degrees of freedom (DoF), Structural Similarity Index (SSIM)

Chapter 2

Background and Related Work

This chapter provides an overview on visual odometry in Ch. 2.1. The related work for geometry-based methods and deep learning-based methods are introduced in Ch. 2.2 and Ch. 2.3.

2.1 Overview on Visual Odometry

In this section, I will discuss the history and problem formulation of visual odometry, and the building blocks of a standard pipeline. The summary is inspired by the Visual odometry tutorials [24, 59] , Wu's dissertation [74] and Multiple View Geometry in Computer Vision [33].

2.1.1 History

Deriving 3-dimensional (3D) camera motion purely from images was explored in the early 1980s. The work from Moravec [46] was about the Mars rover from NASA. The 3D camera motion from images was estimated in order to perform remote control or navigation. It showed the early work for feature detection in gray images. Moravec was performing odometry under a stereo setting. A camera was slided in a 3×3 window with known baselines (distance between cameras) at each time stamp. With sparse features on 9 images, normalized cross correlation was performed to match keypoints. Outlier rejection was done by checking the consistency between point locations on the 9 images. Survived points were triangulated to find the depth. Least squares of distances between two sets of 3D points were solved to find the relative transformation.

With the development of the algorithms, software and hardware, Nister [51] proposed the first monocular visual odometry with feature detection, matching and pose estimation. Features were detected with Harris corner detector [31] and matched with normalized cross correlation. To initialize the first set of 3D points, relative poses were estimated through the 5-point algorithm [50] in a 2D-2D correspondence setting, which became popular afterwards. Then, 2D correspondences can be triangulated into 3D points with known relative poses. When a new frame is added, 2D-3D correspondences could be formed from known 3D points. The correspondences were utilized for pose estimation using 3-point algorithm, or PnP [30]. RANSAC [22] was applied for outlier rejection on pose estimation stages.

The method from Nister [51] has served as the basic pipeline for visual odometry. The following sections introduce the methods in details.

2.1.2 Problem Formulation

Notation. We refer to the pair of images as $\mathbf{I}, \mathbf{I}' \in \mathbb{R}^{H \times W}$, the transformation matrix from frame i to j as $\mathbf{T}_{i,j} = [\mathbf{R}|\mathbf{t}] \in SE(3)$, where $\mathbf{R} \in SO(3)$ in $\mathbb{R}^{3\times3}$ is the rotation matrix and $\mathbf{t} \in \mathbb{R}^{3\times1}$ is the translation vector.

Given a set of images $\mathbf{I}_{0:n} = \{\mathbf{I}_0, ..., \mathbf{I}_n\}$, the camera trajectory w.r.t. the world coordinate is solved as $\mathbf{T}_{0,w:n,w} = \{\mathbf{T}_{0,w}, ..., \mathbf{T}_{n,w}\}$. Assume a homogeneous 3D point, $\mathbf{X}_i = [x_i, y_i, z_i, 1]$, in the world coordinate is projected to a homogeneous 2D point, $\mathbf{p}_i = [u_i, v_i, 1]$. The intrinsic matrix $\mathbf{K}$ can be found by camera calibration (Ch. 2.1.4). The transformation matrix $\mathbf{T} = [\mathbf{R}|\mathbf{t}]$ can link the world coordinate to the camera coordinate through the following equation,

$$\begin{bmatrix} h * u_i \\ h * v_i \\ h \end{bmatrix} = \mathbf{K} \begin{bmatrix} \mathbf{R}|\mathbf{t} \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ z_i \\ 1 \end{bmatrix}. \tag{2.1}$$

$\mathbf{p}_i$ and $\mathbf{X}_i$ are represented in homogeneous coordinate, and h is the arbitrary scaling factor in the

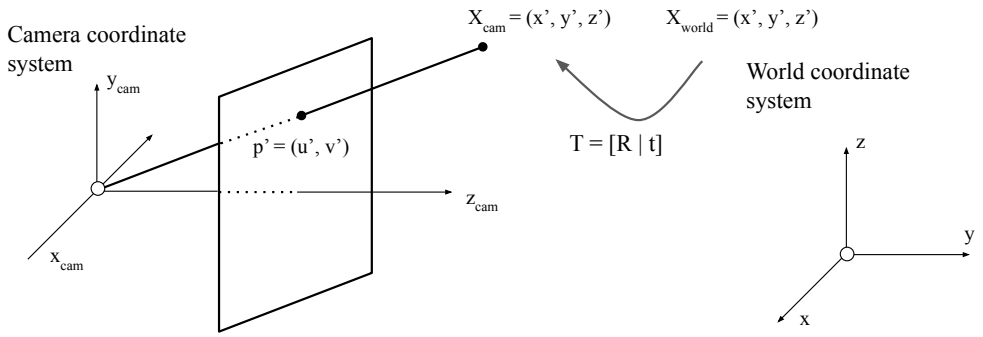


Figure 2.1. Camera projection model. The figure is inspired by the figure in [33].

homogeneous coordinate [33], which could be normalized to find the coordinate $\mathbf{p}' = [u', v']$. To be noted, we adopt left multiplying convention for the transformation matrix in the thesis. The projection is shown in Fig. 2.1.

Absolute pose estimation

The final output of visual odometry is the camera trajectory, which represents the camera poses w.r.t. the world coordinate, defined as $\mathbf{T}_{0,w:n,w}$. The pose, $\mathbf{T}_{i,w} = [\mathbf{R}_{i,w}|\mathbf{t}_{i,w}]$, is the absolute pose of the camera at time i. It describes the camera pose, *i.e.* the rotation and translation, from the perspective of the world coordinate. Eq. (2.1) describes the relation between corresponding 2D points on the image $\mathbf{I}_i$ and 3D points in the world coordinate, and can be applied to solve for $\mathbf{T}_{i,w}$ from the correspondences. The world coordinate is pre-defined and should be consistent during the estimation.

Relative pose estimation

Instead of estimating the absolute pose, we can estimate the relative pose from frame i to j as $\mathbf{T}_{i,j} = [\mathbf{R}_{i,j}|\mathbf{t}_{i,j}]$. The transformation from the camera to the world coordinate should be obtained to fulfill Eq. (2.1). Given the camera pose $\mathbf{T}_{i,w}$ at time i and relative pose $\mathbf{T}_{i,j}$, the pose $\mathbf{T}_{j,w}$ at time j can be inferred as

$$\tilde{\mathbf{T}}_{j,w} = (\tilde{\mathbf{T}}_{i,j})^{-1} * \tilde{\mathbf{T}}_{i,w}, \tag{2.2}$$

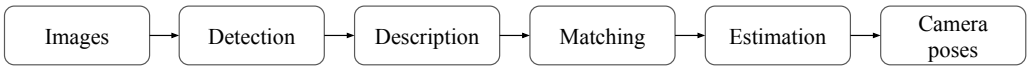


Figure 2.2. Basic pipeline of visual odometry. Given input images, cameras poses are estimated.

where $\tilde{\mathbf{T}}_{i,j}$ is the augmented transformation matrix. In the matrix form, transformation matrix $\mathbf{T}_{i,j} \in \mathbb{R}^{3\times4}$ is augmented as a 4×4 matrix with an additional row $[0,0,0,1]$, where

$$\tilde{\mathbf{T}} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix}. \tag{2.3}$$

We apply the left multiplying convention, denoted as $*$. This forms a linear system, where composition is achieved by matrix multiplication.

Therefore, the camera trajectory can be obtained from relative poses and a reference absolute pose. In evaluation, the trajectory is aligned with the ground truth trajectory before comparison. The modules in the following sections can be connected to estimate the camera trajectory, as shown in Fig. 2.2. Given images as input, camera poses are predicted.Ch. 2.1.3 and Ch. 2.1.4 describe the properties of poses and camera model. Feature detection, description and matching are described in Ch. 2.1.5, and estimation is described in Ch. 2.1.6 and Ch. 2.1.7.

2.1.3 Pose Representation

Pose inversion

From Eq. (2.3), the inverse of an augmented pose could be found by simply inversion of the matrix, where $\tilde{\mathbf{T}}_{i,j} = \tilde{\mathbf{T}}_{j,i}^{-1}$. With the known constraint for rotation matrix $\mathbf{R} * \mathbf{R}^T = \mathbf{I}_{3\times3}$, we can infer

$$\tilde{\mathbf{T}}_{i,j} * \tilde{\mathbf{T}}_{j,i} = \tilde{\mathbf{T}}_{i,j} * \tilde{\mathbf{T}}_{i,j}^{-1} = \begin{bmatrix} \mathbf{R}_{i,j} & \mathbf{t}_{i,j} \\ \mathbf{0} & 1 \end{bmatrix} * \begin{bmatrix} \mathbf{R}_{i,j}^T & -\mathbf{R}_{i,j}^T \mathbf{t}_{i,j} \\ \mathbf{0} & 1 \end{bmatrix} = \mathbf{I}_{3\times3}, \tag{2.4}$$

which easily leads to the inversion.

Euler angle

The rotation matrix can be represented as the euler angle in $\mathbb{R}^3$, which is the the composition of rotation w.r.t. 3 axes. The rotation w.r.t. x, y, z-axis is represented as $R_x(\theta_x), R_y(\theta_y), R_z(\theta_z)$. For example, Given the rotation angle $(\theta_x, \theta_y, \theta_z)$, the rotation matrix can be found as $R = R_x(\theta_x)R_y(\theta_y)R_z(\theta_z)$, with the x-y-z ordering. To be noted, orders of multiplication should be fixed to have consistent results. More details can be found in [3].

Euler angle is convenient because it is the minimum number of representation and easy to understand. When using this representation, the total transformation can be formulated as $(\theta_x, \theta_y, \theta_z, t_x, t_y, t_z)$. However, there's singularity problem known as gimbal lock.

Quaternion

Another form to represent rotation is the quaternion vector, denoted as a 4D unit vector $a + b\mathbf{i} + c\mathbf{j} + d\mathbf{k}$, where a, b, c, d are real numbers, and $\mathbf{i}, \mathbf{j}, \mathbf{k}$ are the axes defined in the quaternion space. Quaternion is robust without the singularity issue in euler angles, but is hard to comprehend. More details can be found in [4]. When using this representation, the total transformation is in the form of 7D vector, with quaternion and translation vectors.

Rodrigues' rotation formula

Rogridues' rotation formula in $\mathbb{R}^3$ is related to axis-angle representation. It can be understood as rotating an angle θ w.r.t. an axis $\mathbf{k}$ in $\mathbb{R}^3$. The axis-angle representation is also be known as the log and exponential map of the rotation matrix. More details can be found in [2, 5, 18].

2.1.4 Camera Model and Calibration

Among the camera models, pinhole camera model is popular and used in the article. Projection from 3D points $\mathbf{X}$ to the image can be described as the light rays passes the camera center $\mathbf{c}_0 = (u_0, v_0)$. A light ray appears to be a pixel $\mathbf{p}$ on an image at the focal length $\mathbf{f}_0 = (f_u, f_v)$,

where numerous rays form a whole image. The mapping from 3D to 2D can be expressed as

$$h\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \mathbf{K} * \mathbf{X} = \begin{bmatrix} f_u & 0 & u_0 \\ 0 & f_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}. \tag{2.5}$$

To be noted, Eq. (2.5) shares the same meaning as Eq. (2.1). Matrix $\mathbf{K}$ is called intrinsic matrix with intrinsic parameters. Image coordinates can be normalized with intrinsic matrix. Let $\tilde{\mathbf{p}} = \mathbf{K}^{-1} * \mathbf{p}$ as normalized image coordinates, where $\tilde{\mathbf{p}} = [\tilde{u}, \tilde{v}, 1]$.

When capturing images, distortion may occur and make straight lines curved. The effect could not be expressed from the intrinsic matrix, but can be corrected on the image. In radial distortion, the image grid is distorted from the radial center, which can be modeled as high order polynomial, as described in [33], p.191.

The intrinsic and distortion parameters can be found through calibration, which is available in Matlab and C code. In the thesis, we assume the parameters are known.

2.1.5 Feature Extraction and Matching

As we can see from the examples in Ch. 2.1.1, matching, or correspondences are the key to camera pose estimation. The points used in correspondences are usually called keypoints or features, in the form of $\mathbf{p}_i$. Given a new frame j, the matching with the previous frame i can be found to form 2D-2D correspondences, which could be solved for relative pose $\mathbf{T}_{i,j}$. If the 3D locations of the matched keypoints on the previous frame are known, we can form 2D-3D correspondences. The absolute camera pose $\mathbf{T}_{j,w}$ can be solved.

To form a set of matching points, there are two streams of work. The first one is to track the keypoints by checking the matching with the neighboring pixels in the next frame. As in [46] and [51], normalized cross-correlation of a patch is applied to find correspondences in a search region. However, the approach is based on the assumption of small baselines between two images, which is not true for SfM systems or large rotation motions. The second line of research

is to perform matching on the whole image. The correspondences are found through matching the descriptors. Feature detection, description and matching is to be discussed.

There are many feature detectors proposed since Moravec [46], which can be categorised into corner and blob detectors. The quality of a detector can be evaluated through localization accuracy (image coordinates), repeatability (a keypoint can be discovered across different images), distinctiveness (a keypoint can be matched correctly), and so on. A corner detector discovers the keypoints from the intersection of lines, *e.g.* Harris [31] and Fast [66] detectors. On the other hand, a blob detector focuses on the difference between neighboring pixels, which can be found through gradients, *e.g.* SIFT [42], and SURF [9]. Detectors have different properties. For example, SIFT detector is known for scale and rotation invariance with subpixel accuracy. SURF detector is not as rotation invariant, but more efficient than SIFT.

With a set of keypoints from each image, descriptors are the key to form matching. Early descriptors take a patch around the keypoint and calculate similarity through normalized cross correlation (NCC) or sum of squared differences (SSD). However, the method is not scale nor rotation invariant and prone to errors. SIFT [42] has been proven to be robust for extracting distinctive features. Through scale-space pyramid, SIFT takes the keypoint at the most dominant scale. When constructing the descriptor, patches adapted by scale and orientation are sampled into a 4×4 grid. In each grid, histograms of gradient magnitude are formed in 8 discretized directions, which create an overall $4 \times 4 \times 8$, 128 dimension vector for each descriptor.

With two sets of keypoints and descriptors, correspondences can be found through nearest neighbor matching. Two-way nearest neighbor matching ensures the descriptor is the closest to each other. Moreover, ratio test, which rules out the matching whose second closest feature is closer than a threshold, can increase the robustness to repetitive patterns.

2.1.6 Motion Estimation

Motion estimation can be categorized into two groups, epipolar geometry and optimization. Here we briefly introduce the basic idea.

Epipolar geometry

To solve for camera pose using epipolar geometry, we require correspondences as input. Given 2D-3D correspondences, PnP [38] is applied to find absolute camera pose w.r.t. the world coordinate. Only a minimum of 3 points are required to estimate the pose up to scale. This is easier and faster to apply RANSAC compared to other models needing more points. Due to the robustness of the algorithm and small number of required matching, PnP algorithm is adopted by many visual odometry pipelines, including ORB-SLAM [47, 48].

However, 2D-3D correspondences are sometimes not feasible if 3D points are not available, such as the initialization of the system. With only 2D-2D correspondences, fundamental matrix or essential matrix can be estimated to solve for the camera pose. Here we briefly introduce the equations, which can be found in [33] for more details. The relation between a pair of corresponding points, $\mathbf{p} = [u, v, 1]$ and $\mathbf{p}' = [u', v', 1]$ can be related as,

$$\mathbf{p}'^T \mathbf{F} \mathbf{p} = 0 \tag{2.6}$$

where $\mathbf{F} \in \mathbb{R}^{3\times3}$ is the fundamental matrix. $\mathbf{F}$ incorporates the projection matrix and the relative pose up to scale, which has 7 degrees of freedom with rank 2. It is useful when the camera intrinsic matrix is unknown. The equation can be interpreted as the coordinate $\mathbf{p}'$ is on the epipolar line $\mathbf{F} * \mathbf{p} = 0$. By normalizing the image coordinates, we can have,

$$\tilde{\mathbf{p}}'^T \mathbf{E} \tilde{\mathbf{p}} = (\mathbf{K}'^{-1}\mathbf{p}')^T \mathbf{E} (\mathbf{K}^{-1}\mathbf{p}) = \mathbf{p}'^T * \mathbf{K}^{-T}\mathbf{E}\mathbf{K}^{-1} * \mathbf{p} = 0, \tag{2.7}$$

where $\mathbf{E}$ is called essential matrix, with 5 degrees of freedom. $\mathbf{E}$ incorporates the relative pose without scale. $\mathbf{F}$ can be converted into $\mathbf{E}$ through the equation

$$\mathbf{E} = \mathbf{K}'^T \mathbf{F} \mathbf{K}. \tag{2.8}$$

8-point algorithm [32] can be applied to solve for the fundamental matrix from correspondences. When camera parameters are known, 5-point algorithm [50] is applied to solve for $\mathbf{E}$.

An essential matrix contains the information of rotation and translation.

$$\mathbf{E} = [\mathbf{t}]_\times \mathbf{R} = \mathbf{R}[\mathbf{R}^T\mathbf{t}]_\times, \tag{2.9}$$

where $[.]_\times \in \mathbb{R}^{3\times3}$ is the cross-product matrix.

Using singular value decomposition (SVD), we can solve for the rotation and translation from the essential matrix. However, we get 4 solutions with the combinations of 2 rotation matrices and 2 translation vectors. Although all solutions satisfy Eq. (2.9), the only solution can be found by checking if all points are in front of the camera.

Optimization method

Optimization method, on the other hand, requires no detection and description steps when solving for the camera poses, while coarse correspondences are built through initial depth and pose. Given the parameters, and initial estimate, the solution is found by minimizing the energy function. Newton, Gauss-Newton, gradient descent, and Levenberg–Marquardt algorithm (LM algorithm) are common options [33, 52].

The idea to estimate the relative camera pose is to enforce the photometric consistency across two frames. An example of energy function is shown as follows,

$$E(\mathbf{T}_{r,t}) = \sum_i (\mathbf{I}_r(\mathbf{p}_i) - \mathbf{I}_t(\omega(\mathbf{p}_i, \mathbf{D}_r(\mathbf{p}_i), \mathbf{T}_{r,t})))^2, \tag{2.10}$$

where the subscript r is for reference frame, t is for target frame, and i is for image points [20]. The parameters are the camera pose from reference to target frame, $\mathbf{T}_{r,t}$. The function $\omega(.)$ is the warping function using 2D points $\mathbf{p}$, depth $\mathbf{D}$, and pose $\mathbf{T}$. The function $\mathbf{I}(.)$ gets the intensity value from the image coordinate $\mathbf{p}$. The warping forms the correspondences between $\mathbf{I}_r$ and $\mathbf{I}_t$.

2.1.7 Outlier Rejection

Outliers lie in mostly every sampling-based method, and may deteriorate the prediction. It is crucial to have outlier rejection mechanisms for robust estimation, *i.e.* RANdom SAmple Consensus (RANSAC) [22, 65]. The typical steps in RANSAC are as follows, summarized from the Algorithm 4.4 in [33].

Given a dataset S containing outliers,

1. Randomly sample a subset s to fit the model.
2. Given a metric and a threshold t, find the inlier set S_i of the current estimate.
3. If the size of S_i, denoted as $N(S_i)$, is larger than before, save the estimate and $N(S_i)$.
4. If $N(S_i)$ is larger than some threshold T, or N steps are reached, terminate the loop. Use the largest inlier set S_i to generate final estimate.

The accuracy is affected by the threshold t, T, and number of steps N. To guarantee the accuracy, N could be decided by the number of samples in minimum subset s to fit the model, and the percentage of outliers. The larger subset s is or the higher percentage of outliers, more iterations are needed. The rule of thumb is to use the outlier rejection mechanism whenever outliers exist.

2.1.8 Triangulation and Keyframe Selection

With the relative poses and correspondences between two frames, we can do triangulation to find the depth (or disparity) of the point, which lifts the 2D points to 3D points. The basic idea of using triangulation is to utilize the relation $\mathbf{p} = T * \mathbf{X}$ and $\mathbf{p}' = T' * \mathbf{X}$, where $\mathbf{p}$ and $\mathbf{p}'$ define the correspondences on the images, $\mathbf{X}$ is the shared 3D points, and $\mathbf{T}$, $\mathbf{T}'$ define the projection from the 3D points to the image planes. $\mathbf{T}$ and $\mathbf{T}'$ are the absolute poses. We can define $\tilde{\mathbf{T}} = \mathbf{I}_4$, so $\tilde{\mathbf{T}}' = \tilde{\mathbf{T}}_{rel} * \tilde{\mathbf{T}}$. The 3D point can be lifted based on the camera coordinate system in the first image. The relation can be formulated as a linear problem, and solved by the Direct Linear Transformation (DLT) algorithm. More details can be found in Ch. 4.1 and Ch. 12.2 in [33].

When doing triangulation, images with large baselines could reduce the error, where keyframes are usually introduced for the condition. When the motion between two cameras is large, the triangle of the 3D point and the correspondences can be formed correctly. In a keyframe-based system, we can do triangulation only on the selected keyframes. The keyframes are the representative frames, or at least have some distance between each other. Everytime the keyframe is selected and inserted, the 2D correspondences between the inserted keyframe and previous keyframes can be triangulated to 3D points, and added to the global map.

2.1.9 Bundle Adjustment (BA)

With the pipeline estimating the camera motion using correspondences, the estimation can be further optimized using bundle adjustment. The key idea is to optimize the estimate across multiple frames using the consistency between the poses and points. The bundle adjustment in a keyframe-based system can have the benefit of optimizing across more distinctive frames. The trajectory coverage of one optimization is also larger than the system without keyframes, which reduces the drifts effectively. Bundle adjustment can be categorized into geometric bundle adjustment and photometric bundle adjustment.

Assume we have N frames in a window of optimization. In geometric bundle adjustment, we have a global map with 3D points and the corresponding 2D projected points scattered on the N frames. To be noted, not all the points are observed by every frame. We also have the absolute poses of the N frames. The energy function is defined as the reprojection error of the 3D points.

$$\min_{\hat{\mathbf{T}}^i, \hat{\mathbf{X}}_j} \sum_{ij} d(\mathbf{K}\hat{\mathbf{T}}^i\hat{\mathbf{X}}_j, \mathbf{p}^i_j)^2, \tag{2.11}$$

where $\hat{\mathbf{T}}^i$ are the poses in N frames, $\hat{\mathbf{X}}_j$ are the 3D points, and $\mathbf{p}^i_j$ is the projected 3D point j on frame i. Assume the intrinsic matrix $\mathbf{K}$ is known. $d(\mathbf{p}, \mathbf{p}')$ is the distance between the points $\mathbf{p}$ and $\mathbf{p}'$ on the 2D image plane. The optimization of 3D points and camera poses can be done through the method described in Ch. 2.1.6, as shown in Fig. 2.3. More details can be referred to

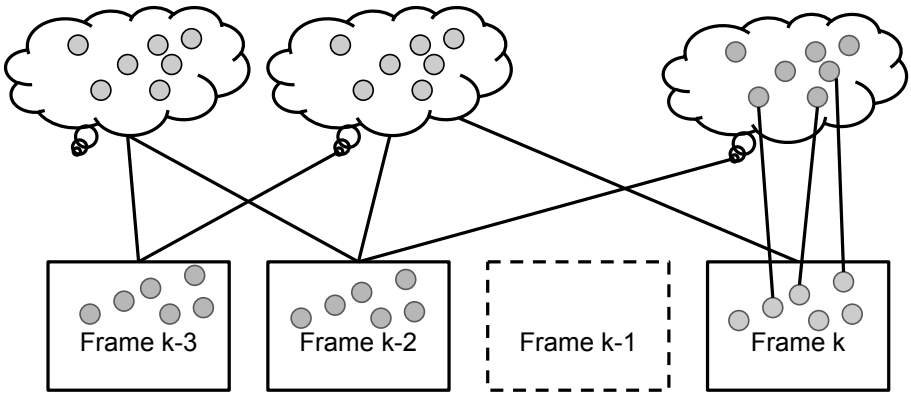


Figure 2.3. Geometric bundle adjustment. The clouds host 3D points and have matching with 2D points across frames. The optimization is applied on the 2D-3D correspondences across multiple frames. Non keyframes are skipped during the optimization.

Ch. 18.1 in [33].

For photometric bundle adjustment, we have the same parameters to optimize but different energy function. The idea is similar to Eq. (2.10), but is extended to N frames. The parameters are optimized using the photometric consistency. It usually requires more computational cost than geometric bundle adjustment with known 2D-3D correspondence. The method is introduced in LSD-SLAM [20] and DSO [19].

2.2 Geometry-Based Method

Visual odometry (VO) is a well-established field [21, 23, 49], which estimates camera motion between image frames. This line of research can be separated into two main groups, feature-based methods and direct methods. For feature-based methods, *e.g.* [27, 47], sparse keypoints for images are detected and described in order to form a set of correspondences. The correspondences are then used for pose estimation using 8-point algorithm [32], PnP [38], or optimized jointly with pose using bundle adjustment [67]. Due to the presence of localization noise and outliers, RANSAC [22] is a popular choice for outlier rejection. However, the method

struggles in case of textureless or repetitive patterns, where keypoints are noisy and difficult to match, as it only uses sparse features across the image and and strives to find a good subset of them. This issue motivated the direct methods [20, 49], which maximizes photometric consistency over all pixel pairs. However, the method suffers in dynamic scenes or challenging lighting environments. Methods combining sparse feature-based methods and direct methods are proposed in recent years, *e.g.* [19, 21, 23], and in addition, loop closure and bundle adjustment (BA) have been applied to extend VO to simultaneous localization and mapping (SLAM).

2.3 Deep Learning-Based method

Learning-based visual odometry

Deep learning for VO has developed rapidly in recent years, *e.g.* [8, 10, 36, 39, 40, 62, 64, 70, 84], taking advantages of convolutional neural networks (CNN) for better adaptation to specific domains. For the monocular camera setting, CNN-SLAM [64] claims that learned depth prediction helps in textureless regions and corrects the scale. PoseNet [36] utilizes CNN to predict a 6 DoF global pose and claims the ability to adapt to a new sequence with fine-tuning. To take advantages of temporal information, DeepVO [70, 71] proposes to use recurrent neural networks (RNN) to predict poses along the sequence. The most recent work is [75] which brings learnable memory and refinement modules into the framework. For the sparse feature-based category, some works have been done using learning-based methods, *e.g.* [16, 17, 35, 82]. In [35], feature descriptors of a two-layer shallow networks is combined with the SLAM pipeline. In addition, SuperPoint [16, 17], which is a learned feature extractor, is combined with BA to update the stability score for each point. However, learned feature extractor is employed in an off-the-shelf manner, which may not be optimal from an end-to-end perspective.

Learning-based feature extraction and matching

Feature extraction, which consists of keypoint detection and description, has been utilized in a variety of vision problems. Traditionally, detectors [42, 58, 66] and descriptors [42, 58]

are mostly designed by heuristics. SIFT [42], which utilizes the Gaussian feature pyramid and descriptor histograms, has achieved success over the past decade. In recent years, deep learning has been utilized to build up feature extractors, *e.g.* [17, 53, 77]. To our knowledge, LIFT [77] is the first end-to-end pipeline, which consists of a SIFT-like procedure with sliding window detection and is trained on ground truth generated from SIFT and SfM [73]. LF-net [53] optimizes keypoints and correspondences with gradients using ground truth camera poses and depth. SuperPoint [17] proposes a self-supervised pipeline to train detector and descriptor at the same time and beats SIFT in HPatches [7] evaluation, with some follow-up works [15, 63]. However, all of the feature extractors are not optimized in together with the overall VO system, leading to suboptimal performance. Also, their evaluation metrics, *i.e.* matching score, does not necessarily reflect the performance of pose estimation in a VO task.

Learning-based camera pose estimation

Learning-based methods for camera pose estimation have been gaining attention in recent years. Following direct methods, the works [28, 39, 57, 69, 78, 79, 85] take advantages of geometric constraints of 3D structure, and jointly estimate depth and pose in an unsupervised manner using photometric consistency. Poursaeed *et al.* [54] uses Siamese networks [37] to regress the fundamental matrix between left and right views through the sequence. For feature-based pipeline, DSAC [11] makes a differentiable sampling-based version of RANSAC, while others [45, 56] utilize PointNet-like architecture [55] to weigh each input correspondence and subsequently solve for the camera pose. These methods retain the mathematical and geometric constraints from traditional methods, and therefore can be more generalizable than direct prediction from image appearance.

2.4 Conclusion

In this chapter, we introduce the basic pipeline of visual odometry using epipolar geometry or optimization. The building blocks are critical for a robust visual odometry system. We

also introduce the related work for geometry-based and deep learning-based methods. The field is actively explored using deep learning. This leads to the following analysis and development for both methods.

In Ch. 2, in part, has been submitted for publication of the material as it may appear in Conference on Intelligent Robots and Systems (IROS), 2020. You-Yi Jau, Rui Zhu, Hao Su, Manmohan Chandraker. The thesis author was the primary investigator and author of this paper.

Chapter 3

Geometry-Based Method for Visual Odometry

We choose the sparse keypoint-based method from the geometry-based categories due to clear success of the line of research, compared to direct or dense methods. ORB-SLAM [47, 48] is selected as our framework for geometry-based method because it is a monocular system that works well across different scenes. It serves as a strong baseline against later approaches.

In the following sections, we give an overview of ORB-SLAM pipeline in Ch. 3.1. Experiments on two different datasets are presented in Ch. 3.2. The analysis and discussion is in Ch. 3.3, and the conclusion is in Ch. 3.4.

3.1 Overview to ORB-SLAM

From Fig. 3.1, we can see the pipeline of ORB-SLAM with image frames as input. Different modules, including tracking, mapping, and loop closing, are designed to enhance the accuracy and robustness. Keyframes and global map are critical for long-term prediction as well as the balance of efficiency. In our experiments, we disable loop closing module, and evaluate the visual odometry. We call the system as ORB-SLAM-VO.

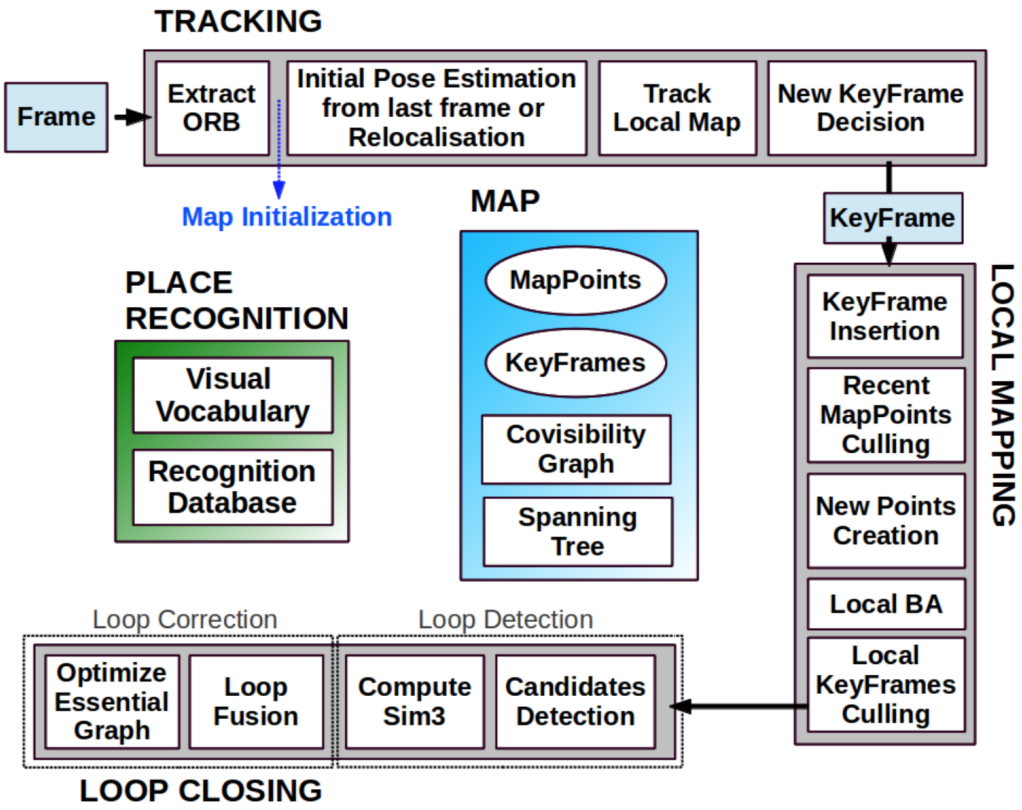


Figure 3.1. Overview of ORB-SLAM. The figure is from the original paper [47].

3.1.1 Tracking

The tracking module follows the general pipeline as in Ch. 2.1, from feature extraction, matching to pose estimation. For feature detection and description, ORB-SLAM uses ORB features [58], which consists of multi-scale FAST corners [66] and 256 bit descriptors. With the ORB features, we can initialize the first pose from 2D-2D correspondences. Then, triangulation and bundle adjustment (Ch. 2.1.9) are done for map initialization. To be noted, heuristic design is made to enhance robustness.

Once the initialization is done, current pose is predicted from the previous pose and optimized through 2D-3D correspondences. We use constant velocity model to generate initial pose [80]. 2D-2D correspondences are found through guided search with the initial pose. The search range is increased to be wider if few correspondences were found. The 2D-2D matching can be extended to 2D-3D with the known 3D coordinates from the previous frame. Also, the 2D-3D correspondence set are further expanded through a local map with a set of keyframes. The optimization is performed in the similar way in Ch. 2.1.9.

Keyframes play a role to incorporate temporal consistency and reduce drift. Instead of performing optimization across every frame, we select keyframes as representatives. Keyframe selection is based on the number of overlapping keypoints or matching with the reference keyframe. Once a new keyframe is inserted, triangulation and optimization are performed afterwards. The optimization updates the absolute poses of keyframes, where the regular frames in between are represented as relative poses w.r.t. the nearby keyframe.

3.1.2 Mapping

Different from tracking module, which deals with every frame, mapping module only takes care of keyframes. It also manages a global map consisted of 3D map points, keyframes, and the covisibility graph. For every keyframe insertion, map points and keyframes are examined for maintenance. Redundant points, which are erroneous or not trackable, are deleted during

point culling. New points are created through triangulation. Covisibility graph shows the relation of the keyframes, where nodes represent the keyframes and edges indicate that the two keyframes share overlapping views. Bundle adjustment for poses and points is done for optimization (Ch. 2.1.9). Redundant keyframes, where most points are covered by other keyframes, are removed at this stage. The iterative insertion and culling maintain the efficiency of the system in long term prediction.

3.1.3 Loop Closure

In ORB-SLAM, loop closure is a mechanism to detect and correct long term drifts. The idea is to compare the current keyframe with previous keyframes to find the loop in the trajectory. The visual cues are described and compared through bag of words. Once the loop is detected, optimization is done to correct the pose and points within the loop. However, this module is disabled in our visual odometry setting.

3.2 Experiments

We perform ORB-SLAM without loop closure, named ORB-SLAM-VO, on KITTI [26] and EuRoC [13] datasets. The implementation is based on open-sourced ORB-SLAM [48] [1], with python binding[2]. The evaluation is performed using the script from [82] [3] and python evo package [29]. The quantitative results are shown in Tab. 3.1 and Tab. 3.2. The qualitative results are shown in Fig. 3.2 and Fig. 3.3. To be noted, due to the different definition of axes in [82] and [29], the axes in Fig. 3.2 are referred to the x, y-axis in the world coordinate, where the axes in Fig. 3.3 are already in the world coordinate.

[1] https://github.com/raulmur/ORB_SLAM2
[2] https://github.com/jskinn/ORB_SLAM2-PythonBindings
[3] https://github.com/Huangying-Zhan/kitti-odom-eval

3.2.1 Datasets

KITTI dataset [26] was published in 2013 and has become a standard visual odometry benchmark. The dataset captures outdoor scenes from a vehicle. The benchmark contains 22 sequences in total, where sequences 00-10 have ground truth poses released. The ground truth poses are obtained from GPS and IMU. We run ORB-SLAM-VO on sequences 00-10.

EuRoC dataset [13] contains visual-inertial data collected by a micro aerial vehicle (MAV). It presents 11 scenes from easy to difficult in indoor environments with 6D pose ground truth. We run ORB-SLAM-VO on 10 sequences (except MH_03_medium).

3.2.2 Evaluation Metrics

Average translational/ rotational RMSE

The metric is specific for KITTI dataset, estimating the average error per meter. The error is averaged over the root mean square error (RMSE) of all possible sequences in the length of $100, 200, ..., 800$ meters. The unit for average translational RMSE is percentage $[\%]$, and the unit for average rotational RMSE is degree per meter $[deg/m]$. The error is represented as (t_{err}, r_{err}) in Tab. 3.1.

Absolute Trajectory Error (ATE)

ATE, also called absolute pose error (APE), is a metric used in TUM dataset [61]. It estimates the error for absolute poses, which reflects the global accuracy. Since the estimated trajectory is usually using a different world coordinate from the ground truth, we need to align the two trajectories. In our case, since the real scale is unknown, we apply Sim(3) Umeyama alignment [68] with 7 degrees of freedom to align the rotation, translation and scale. The trajectories are visualized in Tab. 3.1 and Tab. 3.2.

Table 3.1. Quantitative result of ORB-SLAM-VO on KITTI dataset. We show the results for odometry sequences 00-10. The metrics are described in Ch. 3.2.2.

Method	Metric	00	01	02	03	04	05	06	07	08	09	10	Avg. Err.
ORB-SLAM2 (w/o LC)	t_{err}	21.001	109.677	10.278	1.822	1.510	17.586	25.004	12.436	26.908	18.934	6.000	22.832
	r_{err}	0.278	1.260	0.236	0.337	0.175	0.573	0.605	0.886	0.452	0.468	0.535	0.528
	ATE	75.169	483.244	48.063	0.824	1.143	53.379	74.207	20.812	119.772	70.164	20.929	87.973
	RPE (m)	0.280	3.184	0.173	0.030	0.047	0.274	0.386	0.142	0.432	0.345	0.109	0.491
	RPE (°)	0.068	0.119	0.061	0.052	0.051	0.051	0.045	0.054	0.059	0.059	0.060	0.062

Table 3.2. Quantitative result of ORB-SLAM-VO on EuRoc dataset. The metrics are described in Ch. 3.2.2.

Method	Metric	MH01	MH02	MH04	V101	V102	V103	MH05	V201	V202	V203	Avg. Err.
ORB-SLAM2-VO	ATE	0.037	0.030	0.101	0.088	0.085	0.431	0.093	0.066	0.066	0.535	0.153
	RPE (m)	0.050	0.056	0.107	0.040	0.092	0.080	0.106	0.040	0.090	0.140	0.080
ORB-SLAM2 (w/ LC)	ATE	0.039	0.031	0.064	0.087	0.063	0.305	0.825	0.060	0.055	0.244	0.177
	RPE (m)	0.050	0.055	0.107	0.039	0.094	0.120	0.129	0.040	0.090	0.139	0.086

Relative Pose Error (RPE)

The metric, proposed in TUM dataset [61], is applied to evaluate the relative poses. Assume we have ground truth pose from frame i to j, $\mathbf{T}_{i,j}^{gt}$ and estimated pose, $\mathbf{T}_{i,j}^{est}$. The error term can be computed through the inverse composition of augmented transformation matrix as

$$\tilde{\mathbf{T}}_{i,j}^{err} = (\tilde{\mathbf{T}}_{i,j}^{gt})^{-1} * \tilde{\mathbf{T}}_{i,j}^{est}. \tag{3.1}$$

Translation error is computed through RMSE of the translation vector, in the unit of meters (m). Rotation error is the magnitude of angle in the axis-angle representation, in the unit of degrees (deg). The quantitative results of ATE and RPE are shown in Tab. 3.1 and Tab. 3.2 for each dataset.

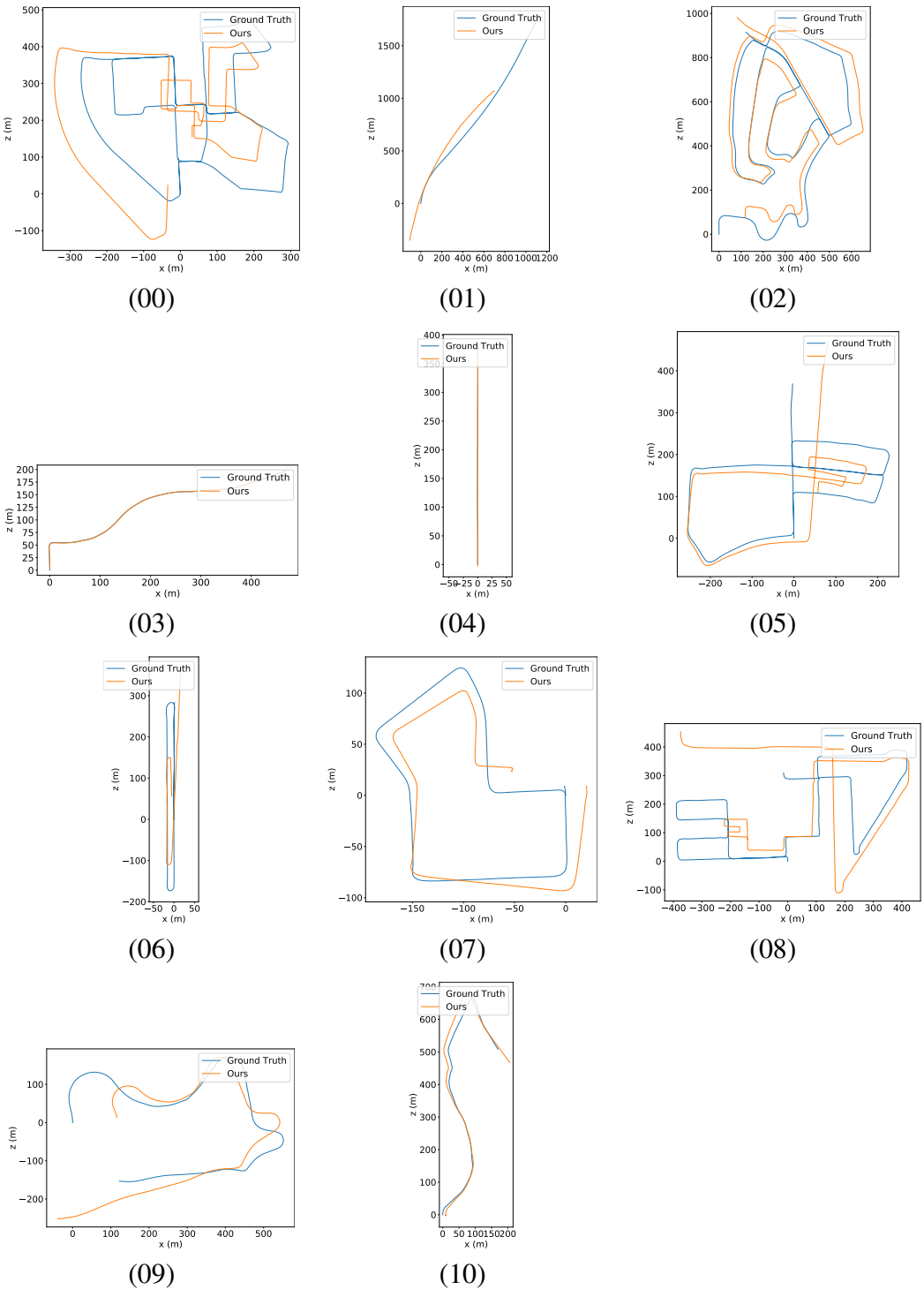


Figure 3.2. Qualitative VO results on KITTI dataset using ORB-SLAM2-VO. Ground truth trajectories are plotted in blue, while estimated ones are in orange. The trajectories are aligned in 7 DoF.

3.3 Discussion

Observing the success of ORB-SLAM in KITTI and EuRoC datasets, we analyze the pipeline and identify several key factors that lead to the success in the system.

3.3.1 Initialization and Relocalization

When 3D-2D correspondences can not be found, 2D-2D matching should be performed to estimated relative pose. It happens in the phase of initialization and relocalization. In initialization, global map and 3D points have not yet created. Instead, 2D-2D correspondences should be found. Initial relative poses are important because they affect the creation of map and 3D points. In ORB-SLAM, we consider the cases of planar or structured scenes in the image. If the correspondences lie on a planar surface, homography matrix is estimated. For a non-planar or structured scene, essential matrix is found using the method in Ch. 2.1.6. We heuristically pick the estimation with better model fitting.

When the tracking is lost, relocalization is performed. The same technique is applied to find the relative pose w.r.t. nearby keyframes using 2D-2D correspondences. This phase is critical to recover the estimation when challenging scenes make the system fail.

3.3.2 Outlier Rejection

Outliers appear in the set of 2D-2D, or 2D-3D correspondences, where mechanism is needed to enhance robustness. During the estimation, RANSAC is applied for outlier rejection, as described in Ch. 2.1.7.

3.3.3 Keyframe-Based System

ORB-SLAM is a keyframe-based system, which makes the estimation and optimization efficient. As described in Ch. 2.1.8, the decision of keyframe insertion is made for every input frame. To be more specific, the conditions for keyframe insertion are as follow.

- When more than 20 frames are observed since last keyframe, the keyframe is inserted to ensures good localization.
- When more than 50 points are tracked by this frame, the keyframe is inserted to ensures good tracking.
- When less than 90% of the points are overlapped as the reference keyframe, the frame is inserted without redundancy.

When one of the above conditions is satisfied, the current frame is identified as a keyframe. To reduce redundancy, the system deletes a keyframe when the 3D map points on the keyframe are overlapped more than 90% by 3 other keyframes. The maintenance of keyframes not only makes the system efficient, but also leads to robust estimation by taking the advantages of 3D geometry.

3.3.4 Long Feature Tracking

2D-2D correspondences, sometimes across multiple frames, are the key for the estimation and optimization in ORB-SLAM. Therefore, the feature extractor should be robust to reduce the outlier rate. ORB-SLAM adopts the ORB features [58], which is faster than SIFT [42] for up to 2 orders of magnitude. The feature extractor is rotation invariant and robust to noise using a binary descriptor. The feature detector adopts the design of FAST corner detector [66] with additional orientation. In general, the sparse keypoints can better survive the change of lightning, which happens during tracking. The discussion for feature extractor can be found in Ch. 2.1.5.

3.3.5 Robust Bundle Adjustment

The bundle adjustment is performed in two ways, local bundle adjustment and pose graph optimization. Local bundle adjustment is triggered everytime a new keyframe is inserted. The poses and 3D points on nearby keyframes are optimized to minimize the reprojection error, as shown in Ch. 2.1.9.

However, the complexity grows linearly when the number of frames increase. It is not feasible to do bundle adjustment for all the keyframes. Instead, pose graph optimization can perform efficiently to correct the estimated poses [60]. Given the covisibility graph representing the connectivity between keyframes and the poses between keyframes, the similarity constraint is put for optimization over 7 DoF. The 6 DoF poses and scales are corrected to reduce drift issues. The final results can be observed in Fig. 3.2 and Fig. 3.3.

3.4 Conclusion

In this chapter, we pick a monocular SLAM system, ORB-SLAM [47,48] for our analysis of a robust visual odometry system. We test ORB-SLAM-VO in outdoor and indoor environments and show the results in Ch. 3.2. Based on the experiments, key factors of the system are analyzed in Ch. 3.3. This system inspires us of designing a system leveraging geometry-based and deep learning-based methods. In Ch. 4, we propose a deep learning-based pipeline with geometric constraints.

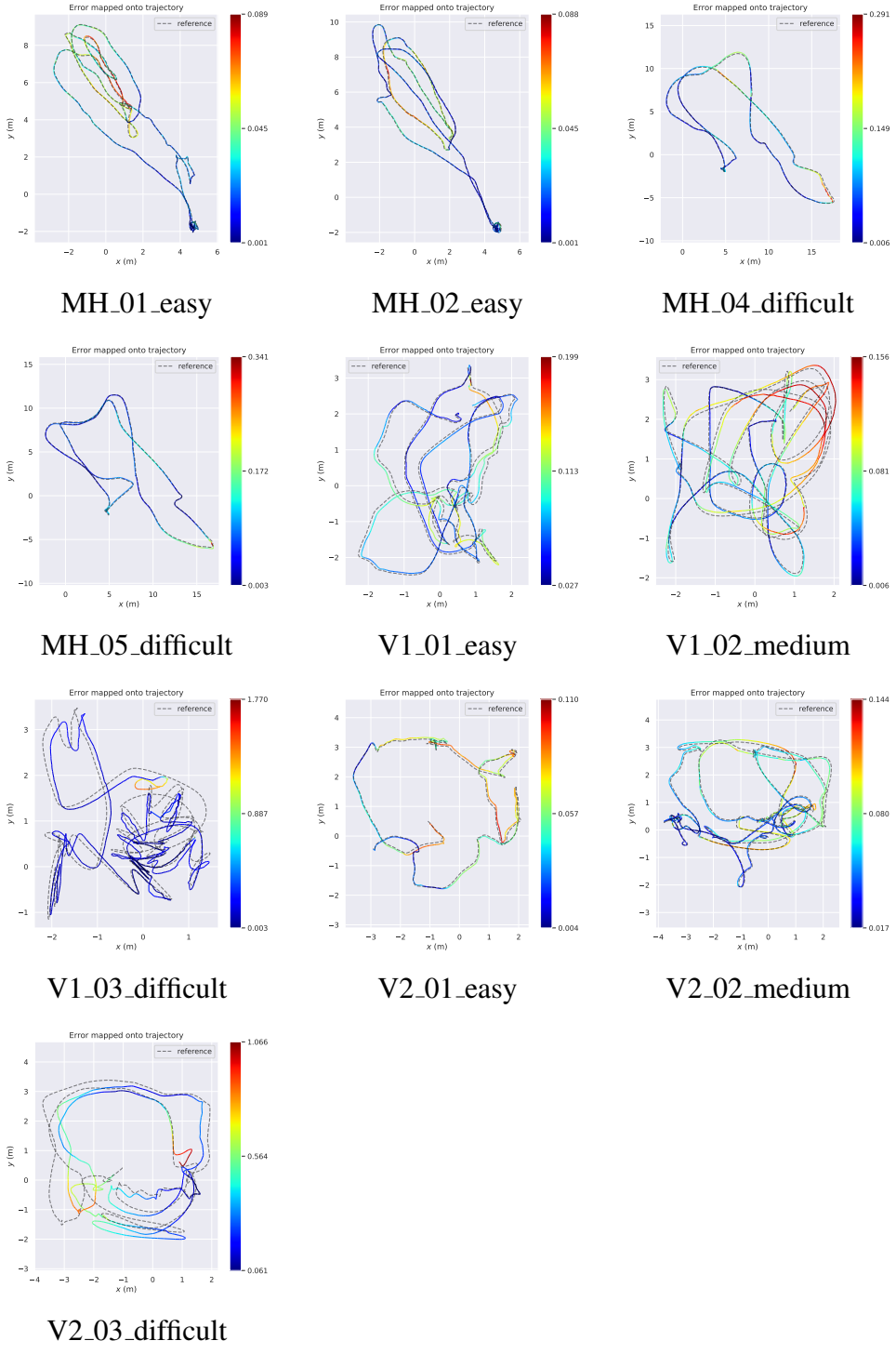


Figure 3.3. Qualitative VO results on EuRoC dataset using ORB-SLAM2-VO. The ground truth trajectories are plotted in dotted lines, whereas the estimated ones are mapped in colors with error. The trajectories are aligned in 7 DoF.

Chapter 4

Deep Keypoint-Based Camera Pose Estimation with Geometric Constraints

4.1 Introduction

Camera pose estimation has been the key to simultaneous localization and mapping (SLAM) systems. To this end, multiple methods have been designed to estimate camera poses from input image sequences, or in a simplified setting, to get the relative camera pose from two consecutive frames. Traditionally a robust keypoint detector and feature extractor, *e.g.* SIFT [42], coupled with an outlier rejection framework, *e.g.* RANSAC [22], has dominated the design of camera pose estimation pipeline for decades.

Recently there have been efforts to bring deep networks to each step of the pipeline, specifically keypoint detection [17, 53, 77], feature extraction [17, 53] and matching [11, 17, 53], as well as outlier rejection [11, 12, 56]. The potential benefit is being able to handle challenges such as textureless regions by incorporating data-driven priors. However, when combining such components to replace the classic counterparts, conventional SIFT-based camera pose estimation still significantly outperforms them by a considerable margin. This could be attributed to three basic challenges for learning-based systems. First, these learning-based methods have been individually developed for their own purposes, but never been trained and optimized end-to-end for the ultimate purpose of getting better camera poses. Geometric constraints and the final pose estimation objective are not sufficiently incorporated in the pipeline. Second, learning-based

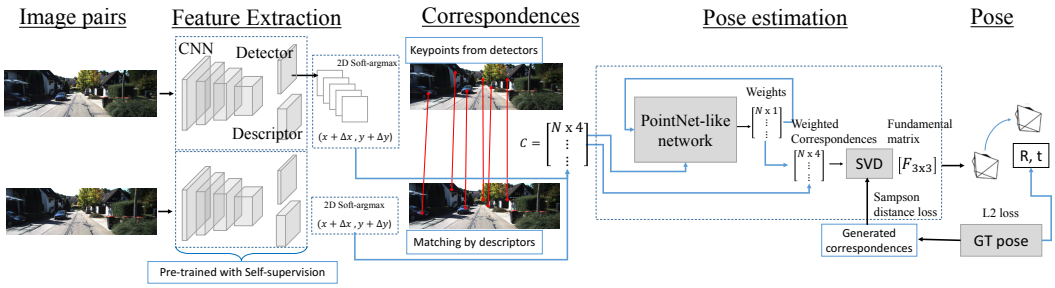


Figure 4.1. Overview of the system. A pair of images is fed into the pipeline to predict the relative camera pose. Feature extraction predicts detection heatmaps and descriptors for finding sparse correspondences. Local 2D `Softargmax` is used as a bridge to get subpixel prediction with gradients. Matrix $\mathbf{C}$ of size $N \times 4$ is formed from correspondences. $\mathbf{C}$ is the input for pose estimation, where the PointNet-like network predicts weights for all correspondences. Weighted correspondences are passed through SVD to find fundamental matrix F, which is further decomposed into poses. Ground truth poses (GT poses) are used to compute L2 loss between rotation and translation (**pose-loss**). Correspondences generated from GT poses are used to compute fundamental matrix loss (**F-loss**). See more details in Sec. 4.2.

methods have over-fitting nature to the domains they are trained on. When the model is applied to a different dataset, the performance is often inconsistent across various datasets compared to SIFT and RANSAC methods. Third, our evaluation shows that existing learning-based feature detectors, which serve at the very beginning of the entire pipeline, are significantly weaker than the hand-crafted feature detectors (*e.g.* SIFT detector). This is because obtaining training samples with accurate keypoints and correspondences, at the level to surpass or just match the subpixel accuracy of SIFT, is tremendously difficult.

In face of these issues when naively putting existing learning-based methods together, we propose the end-to-end trained framework for relative camera pose estimation between two consecutive frames (Fig. 4.1). Our framework integrates learnable modules for keypoint detection, description and outlier rejection inspired by the geometry-based classic pipeline. The whole framework is trained in an end-to-end fashion with supervision from ground truth camera pose, which is the ultimate goal for pose estimation. Particularly, in facing the third challenge of requiring accurate keypoints for feature detector training, we introduce a `Softargmax` detector head in the pipeline, so that the final pose estimation error could be back-propagated to provide

subpixel level supervision.

Experiments show that the end-to-end learning can drastically improve the performance of existing learning-based feature detectors, as well as the entire pose estimation system. We show that our method outperforms existing learning-based pipelines by a large margin, and performs on par with the state-of-the-art SIFT-based methods on various datasets. We also demonstrate the significant benefit of generalizability to unseen datasets compared to learning-based baseline methods. We evaluate our model on KITTI [26] and ApolloScape [34] datasets and demonstrate not only quantitatively but also qualitatively. That is, by training end-to-end, we are able to obtain relatively balanced keypoint distribution corresponding to appearance and motion patterns in the image pair.

To summarize, our contributions include:

- We propose the keypoint-based camera pose estimation pipeline, which is end-to-end trainable with better robustness and generalizability than the learning-based baselines.
- The pipeline is connected with the novel `Softargmax` bridge, and optimized with geometry-based objective obtained from correspondences.
- The thorough study on cross-dataset setting is done to evaluate generalization ability, which is critical but not much discussed in the existing works.

We describe our pipeline in detail in Sec. 4.2 with the design of the loss functions and training process. We show the quantitative results and qualitative results of pose estimation in Sec. 4.3.

4.2 Method

4.2.1 Overview

We propose a deep feature-based camera pose estimation pipeline called `DeepFEPE` (Deep learning-based Feature Extraction and Pose Estimation), which takes two frames as input

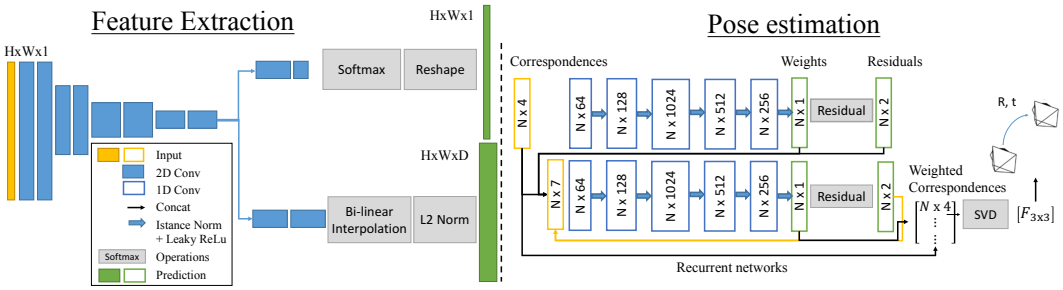


Figure 4.2. Network structure of our feature extraction (FE) and pose estimation(PE) modules. FE module [17] has VGG-like structure, with gray images as input, and detection and description heatmaps as output. PE module has N correspondences ($\mathbf{C}$) as input, with several layers of 1D convolution to initialize weights and compute residuals. The weights, residuals and correspondences are fed into the RNN with the same structure for D iterations (D=5). From the final correspondences with weights, the fundamental matrix is estimated.

and estimates the relative camera pose. The pipeline mainly consists of two learning-based modules, for feature extraction and pose estimation respectively, as shown in Fig. 4.1.

Instead of naive concatenation of the modules, careful designs are made for training `DeepFEPE` end-to-end, which includes `Softargmax` [14] detector head and geometry-embedded loss function. The `Softargmax` detector head equips the feature detection with sub-pixel accuracy, and enables gradients from pose estimation to flow back through the point coordinates. For loss function, we not only regress a fundamental matrix, but also directly constrain the decomposed poses. We enforce a geometry inspired L2 loss on the estimated rotation and translation, which leads to better prediction and generalization ability, as shown in Sec. 4.3. We include more details for `DeepFEPE` and the network structures in Fig. 4.2.

Notation We refer to the pair of images as $I, I' \in \mathbb{R}^{H \times W}$, the transformation matrix from frame i to j as $T_{ij} = [R|t]$, where $R \in \mathbb{R}^{3\times3}$ is the rotation matrix and $t \in \mathbb{R}^{3\times1}$ is the translation vector. We refer to a point in 2D image coordinates as $p \in \mathbb{R}^2$, where $p = [u, v]$.

4.2.2 Feature Extraction (FE)

We use learning-based feature extraction (FE), namely SuperPoint [17], in our pipeline. SuperPoint is chosen as our base component because it is trained with self-supervision and

demonstrated top performance for homography estimation in HPatches dataset [7]. Similar to traditional feature extractors, *e.g.* SIFT, SuperPoint serves as both the detector and descriptor, with the input gray image $I \in \mathbb{R}^{H\times W\times 1}$, and output keypoint heatmap $H_{det} \in \mathbb{R}^{H\times W\times 1}$ and descriptor $H_{desc} \in \mathbb{R}^{H\times W\times D}$. SuperPoint consists of a fully-convolutional neural network with a shared encoder and two decoder heads as the detector and descriptor respectively, as shown in Fig. 4.2.

`Softargmax` Detector Head

To overcome the challenge of training end-to-end, we propose detector head with 2D `Softargmax`. In the original Superpoint, non-maximum suppression (NMS) is applied to the output of keypoint decoder H_{det} to get sparse keypoints. However, the output from NMS only has pixel-wise accuracy and is non-differentiable. Inspired by LF-Net [53], we apply `Softargmax` on the 5×5 patches extracted from the neighbors of each keypoint. The final coordinate of each keypoint can be expressed as

$$(u', v') = (u_0, v_0) + (\delta u, \delta v), \tag{4.1}$$

where in a given 2D patch,

$$\delta u = \frac{\sum_j \sum_i e^{f(u_i, v_j)} i}{\sum_j \sum_i e^{f(u_i, v_j)}}, \delta v = \frac{\sum_j \sum_i e^{f(u_i, v_j)} j}{\sum_j \sum_i e^{f(u_i, v_j)}}. \tag{4.2}$$

$f(u,v)$ denotes the pixel value of the heatmap at position (u,v), and i, j denotes the relative directions in x, y-axis with respect to the center pixel (u_0, v_0). The integer-level keypoint (u_0, v_0) is therefore updated to (u', v') with subpixel accuracy.

The output of the `Softargmax` enables flow of gradients from the latter module to the front, in order to refine the coordinates for subpixel accuracy. To pre-train the FE module with `Softargmax`, we convolve the ground truth 2D detection heatmap with a Gaussian kernel σ_{fe}. The label of each keypoint is represented as a discrete Gaussian distribution on a 2D image.

Table 4.1. The reference table for modules and losses trained for experiments. The table lists all the baselines used in Sec. 4.3.2. Baselines and our models are referred by symbols, as they consist of different FE and PE modules trained using different losses.

Model References		Feature extraction		Pose estimation		Loss		Training
Catogories	Symbols	Sift (Si)	Superpoint (Sp)	RANSAC (Ran)	DeepF (Df)	F-loss (f)	Pose-loss (p)	End-to-end (end)
SIFT + RANSAC (Si-base)	Si-Ran	✓		✓				
Superpoint + RANSAC (Sp-base)	Sp-Ran		✓	✓				
Baseline with Sift + DeepF (Si-models)	Si-Df-f	✓			✓	✓		
	Si-Df-p	✓			✓		✓	
	Si-Df-fp	✓			✓	✓	✓	
Ours - no end-to-end training (Sp-models)	Sp-Df-f		✓		✓	✓		
	Sp-Df-p		✓		✓		✓	
Ours - with end-to-end training `(DeepFEPE)`	Sp-Df-f-end		✓		✓	✓		✓
	Sp-Df-p-end		✓		✓		✓	✓
	Sp-Df-fp-end		✓		✓	✓	✓	✓

Descriptor Sparse Loss

To pre-train an efficient FE, we adopt sparse descriptor loss instead of dense loss. Original dense loss [17] collects loss from all possible correspondences between two sets of descriptors in low resolution output, which creates a total of $(H_c \times W_c)^2$ of positive and negative pairs. Instead, we sparsely sample N positive pairs, and M negative pairs collected from each positive pair, forming $M \times N$ pairs of sampled correspondences. The loss function is the mean contrastive loss as described in [17].

Output of Feature Extractor

We obtain correspondences for pose estimation from the sparse keypoints and their descriptors. To get the keypoints, we apply non-maximum suppression (NMS) and a threshold on the heatmap to filter out redundant candidates. The descriptors are sampled from H_{desc} using bi-linear interpolation. With two sets of keypoints and descriptors, 2-way nearest neighbor matching is applied to form N correspondences, an $N \times 4$ matrix, as input for pose estimation.

4.2.3 Pose Estimation (PE)

Pose estimation takes correspondences as input to solve for the fundamental matrix. Instead of using a fully connected layer to regress fundamental matrix or pose directly as

in [8, 54, 85], we embed geometric constraints, *i.e.* sparse correspondences, into camera pose estimation. To create a differentiable pipeline in replacement of RANSAC for pose estimation from noisy correspondences, we build upon the Deep Fundamental Matrix Estimation (DeepF) [56], and propose a geometry-based loss to train `DeepFEPE`.

Existing Objective for Learning Fundamental Matrix

DeepF [56] formulates fundamental matrix estimation as a weighted least squares problem. The weights on the correspondences indicate the confidence of matching pairs, and are predicted using a neural network model with the PointNet-like structure. Then, weights and points are applied to solve for the fundamental matrix. Residuals of the prediction, as defined in [56], are obtained from the mean Sampson distance [43] of the input correspondences. The correspondences, weights, and residuals are fed into the model recurrently to refine the weights. To be more specific, the residuals $r(\mathbf{p_i}, \mathbf{F})$ are calculated as following:

$$r(\mathbf{p_i}, \mathbf{F}) = |\mathbf{p_i}^T \mathbf{F} \mathbf{p_i}'|(\frac{1}{\|\mathbf{F}^T \mathbf{p_i}\|_2} + \frac{1}{\|\mathbf{F}\mathbf{p_i}'\|_2}), \tag{4.3}$$

where $\mathbf{p_i} = (u, v, 1)$ and $\mathbf{p_i}' = (u', v', 1)$ denote a pair of correspondences in the homogeneous coordinates.

Following [56], the loss is defined as epipolar distances from *virtual* point on a grid, to their corresponding epipolar lines, which are generated from ground truth fundamental matrix. It is abbreviated as **F-loss** in the following sections.

Geometry-based Pose Loss

Due to the fact that a good estimation in epipolar space does not guarantee better pose estimation, we propose a geometry-based loss function by enforcing a loss between estimated poses and ground truth poses. The estimated fundamental matrix is converted into the essential matrix using calibration matrix and further decomposed into 2 sets of rotation and 2 translation matrices. By picking the one camera pose where all points are in front of both cameras (which

gives lowest error among all 4 possible combinations of poses), we obtain the rotation in quaternions [83] and translation vectors, and compute L2 loss as our geometry-based objective. Then, loss terms L_{rot} and L_{trans} are collected from the pose with minimum L2 loss.

$$L_{rot} = min(\|q(\mathbf{R}_{est\_i}) - q(\mathbf{R}_{gt})\|_2), i = [1,2], \tag{4.4}$$

$$L_{trans} = min(\|\mathbf{t}_{est\_i} - \mathbf{t}_{gt}\|_2), i = [1,2], \tag{4.5}$$

where R,t are decomposed from the essential matrix, and $q(.)$ converts the rotation matrix into quaternion vector.

The final loss is followed,

$$\begin{aligned} L = & min(L_{rot}(\mathbf{R}_{est}, \mathbf{R}_{gt}), c_r) + \\ & \lambda_{rt} * min(L_{trans}(\mathbf{t}_{est}, \mathbf{t}_{gt}), c_t), \end{aligned} \tag{4.6}$$

where c_r and c_t are clamping constants for losses to prevent gradient explosion. The geometry-based loss is abbreviated as **pose-loss** in the following sections.

4.2.4 Training Process

After initializing both FE and PE modules respectively, we train the pipeline end-to-end. Our FE module is trained using self-supervised method [17]. The keypoint detector is initialized by synthetic data, which can be used to generate pseudo ground truth for detectors on any dataset with single image homography adaptation (HA). Homography warping pairs are generated on-the-fly for descriptor training [17]. We put a Gaussian filter on the ground truth heatmap to enable prediction with `Softargmax`, where $\sigma_{fe} = 0.2$. For descriptor sparse loss, we have $H_c = H/8$, $W_c = W/8$, $N = 600$, and $M = 100$. NMS window size is set to be $w = 4$. The model is trained with 200k iterations on synthetic datasets, and 50k iterations on real images.

With the correspondences from the pre-trained FE, we initialize the PE module using **F-loss**. The training converges at around 20k iterations. For training with the **pose-loss**, We set

initial $c_r = 0.1$, $c_t = 0.5$ and $\lambda_{rt} = 0.1$.

When connecting the entire pipeline, the gradients from **pose-loss** flow back through the Pose Estimation(PE) module to update the prediction of weights, as well as the Feature Extraction(FE) module to update the locations of keypoints. The pipeline and supervision is shown in Fig. 4.1.

4.2.5 Network Structure

Feature extraction

The architecture of feature extraction module follows SuperPoint model [17], which is a VGG-like structure. Fig. 4.2 depicts the structure of networks, which consists of a shared encoder with two decoder heads. The encoder has eight 3×3 CNN layers with size 64-64-64-64-128-128-128-128. Max-pooling is applied every two layers except the last layer. For decoders, 1 layer of 3×3 CNN and 1 layer of 1×1 CNN is applied, with size 256-65 for the detector and size 256-256 for the descriptor. The output for the detector is further processed by $Softmax$, dustbin cleaning, and *reshape* to the output $H_{det} \in \mathbb{R}^{H \times W \times 1}$. For the descriptor output, bi-linear interpolation and L2 Norm along the depth channel is applied to get the heatmap $H_{desc} \in \mathbb{R}^{H \times W \times D}$. $BatchNorm$ nomalization and *ReLu* activation function are applied to the output of every layer except the final layer, where there's only $BatchNorm$ applied.

Pose estimation

The architecture of pose estimation module follows DeepF model [56], which is a PointNet [55]-like structure. As shown in Fig. 4.2, the module is first used to predict initial weights, and then being used recurrently for D iterations to refine weights based on previous prediction and residuals. The module consists of 5 layers of $1D$ CNN with size 64-128-1024-512-256, where the kernel size and stride size are equal to 1. For each layer except the last one, $InstanceNorm1d$ normalization and $Leaky-ReLu$ activation are applied. For the recurrent network, we keep the design $D = 5$, which is the final choice made by DeepF [56]. For all pose

estimation models, we use 1000 correspondences as input.

4.2.6 Parameter Setting

For feature extraction (FE), we mainly follow the choice of parameters of the original SuperPoint implementation [17]. We set the non-maximum suppression (NMS) distance as 4 pixels, and set the threshold to filter out H_{det} after NMS to be 0.015 ($\approx 1/65$), which corresponds to the depth of detection map. For feature matching, we calculate the cosine distance d_m between two descriptors, which ranges from -1 to 1. We compute $d = 2 - 2 * d_m$ and set the threshold to be 1.0.

For pose estimation, we set the clamping of f-loss at 0.02. For pose-loss, we set clamping to the L2 loss of rotation and translation separately, and decrease the thresholds at 3k and 6k iterations. The clamping parameters are shown in Tab. 4.2.

Table 4.2. Parameters for clamping pose-loss.

Iterations	Rotation	Translation clamping
0-3k	0.1	0.5
3k-6k	0.01	0.3
6k-	0.001	0.1

For SIFT matching, we use $FlannBasedMatcher$ and $knnMatch$ with $k = 2$ in $OpenCV$. Then, we use ratio test with threshold 0.8 to extract matching with high quality. For RANSAC, we use threshold 0.1 to find inliers. For training `DeepFEPE`, we train with batch size 4 and learning rate 10^{-4}.

4.3 Experiments

We evaluate `DeepFEPE` using pose estimation error, and compare with previous approaches. Acronyms and symbols for the approaches are defined in Tab. 4.1. Different methods are evaluated on KITTI dataset [25], and further on ApolloScape dataset [34] to show the generalization ability to unseen data. To be noted, we evaluate for relative pose estimation with

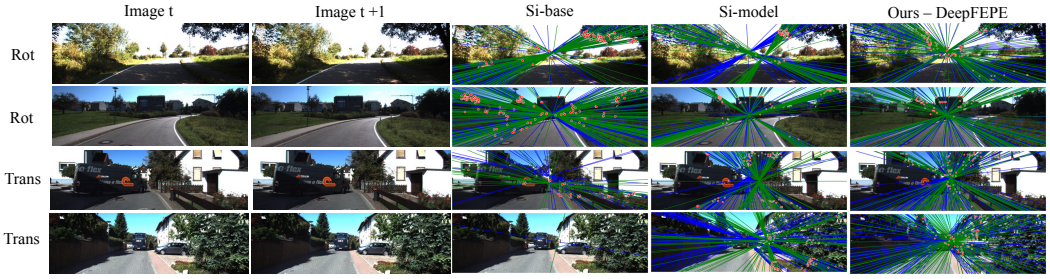


Figure 4.3. Pose estimation comparison. The first two columns show the image pairs. The last three columns compare Si-base, Si-model and `DeepFEPE`. We show 2 examples for rotation and 2 for translation dominated pairs. The blue lines are plotted from ground truth fundamental matrix F_{gt}, as the green lines are from estimated F_{est}. Red dots are the keypoints from correspondences with high weights. For Si-base, the correspondences are selected by RANSAC, where keypoints around vanishing points are usually rejected. However, Si-model utilizes all correspondences to solve for fundamental matrix, which leads to better quantitative results after training. The distribution of points in `DeepFEPE` is more balanced than for others, leading to more accurate pose estimation.

existing baselines in Sec. 4.3.2, but not for visual odometry before extending to multi-frame setting. We demonstrate significant improvement quantitatively for learning-based methods with end-to-end training against baselines, as shown in Tab. 4.3 and Tab. 4.5. To give an insight into the improvement of optimizing SuperPoint from **pose-loss**, we evaluate the epipolar error of correspondences quantitatively in Tab. 4.7 and visualize the change of keypoint distribution during training in Fig. 4.5.

4.3.1 Datasets

We extract all pairs of consecutive frames, *i.e.* with time difference 1, for training and testing.

KITTI dataset We train and evaluate our pipeline using KITTI odometry sequences, with ground truth 6 DoF poses obtained from IMU/GPS. There are 11 sequences in total, where sequences 00-08 are used for training (16k samples) and 09-10 are used for testing (2,710 samples).

ApolloScape dataset The dataset is collected in driving scenarios, with ground truth 6 DoF poses collected from GPS/IMU. It includes different view angles of the camera and lighting

variations from KITTI dataset, and is used for generalization testing. We use the training split from Road 12, 14, 15, 16, 17 for training (22k samples), and testing split in Road 11 for testing (5.8k samples).

4.3.2 Relative Pose Estimation

We evaluate the performance of `DeepFEPE` from the estimated rotation and translation, as in [56]. With the transformation matrix, we calculate the error by composing the inverse of ground truth matrix with our estimation. Then, we extract the angle from the composed rotation matrix as the error term.

$$\mathbf{R}_{rel} = \mathbf{R}_{est} * \mathbf{R}_{gt}^{T}, \tag{4.7}$$

$$\delta\theta = \|Rog(\mathbf{R}_{rel})\|_2, \tag{4.8}$$

where $Rog(.) \in \mathbb{R}^{3\times3} \rightarrow \mathbb{R}^{3\times1}$ converts a rotation matrix to a Rodrigues' rotation vector. The length of the resulting vector represents the error in angle. We measure translation error by angular error due to scale ambiguity between the estimated translation and the ground truth vector.

$$\delta\mathbf{t} = cos^{-1}(\frac{\mathbf{t}_{est} \cdot \mathbf{t}_{gt}}{\|\mathbf{t}_{est}\|\|\mathbf{t}_{gt}\|}) \tag{4.9}$$

The equation computes the angle between the estimated translation vector and ground truth vector. With the rotation and translation error for each pair of images throughout the sequence, we compute the inlier ratio, from 0% to 100%, under different thresholds. Mean and median of the error are also computed in degrees.

We compare different models as follows. **(1) Si-base** (classic baseline models): Correspondences from SIFT are fed into RANSAC for pose estimation. **(2) Si-models** (SIFT and DeepF models): SIFT correspondences are used to estimate pose using deep fundamental matrix [56], which is the current state-of-the-art relative pose estimation pipeline. **(3) Sp-base** (SuperPoint with RANSAC): SuperPoint is pre-trained and then connected to RANSAC for

pose estimation. **(4) Sp-models** (SuperPoint and DeepF models): SuperPoint is pre-trained on the given dataset and then frozen to train DeepF models. **(5)** `DeepFEPE` (Our method with end-to-end training): feature extraction (FE) and pose estimation (PE) are trained jointly using **F-loss** or **pose-loss**. The reference table of the models above are shown in Tab. 4.1, with symbols representing different training combinations. The models are trained on KITTI and evaluated on both KITTI and ApolloScape datasets.

Tab. 4.3 compares the learning-based baseline (Sp-base) with our `DeepFEPE` model, which shows significant improvement w.r.t. rotation and translation error. Looking into the rotation error, the pre-trained SuperPoint [17] performs poorly with RANSAC pose estimation (0.217 degrees median error), whereas the DeepF [56] module improves that to 0.078 degrees. Our `DeepFEPE` further improves the rotation median error to 0.041 degrees, with translation median error from 2.1 (Sp-base) to 0.5 degrees.

Table 4.3. Comparison of pose estimation for learning-based KITTI models on KITTI dataset. The set of models are trained on KITTI with learning-based feature extraction (FE). It shows significant improvement from RANSAC to DeepF pose estimation, and from separate models to end-to-end trained models. (Refer to Tab. 4.1 for acronyms.)

KITTI Models	KITTI dataset - error(deg.) inlier ratio↑, mean↓, median↓					
	Rotation (deg.)			Translation (deg.)		
	0.1↑	Mean.↓	Med.↓	2.0↑	Mean.↓	Med.↓
Base(Sp-Ran)	0.189	0.641	0.217	0.481	5.798	2.103
Sp-Df-f	0.633	0.100	0.078	0.830	1.476	0.846
Sp-Df-p	0.875	0.130	0.047	0.887	1.719	0.539
Ours(Sp-Df-f-end)	0.915	0.053	0.042	0.905	1.662	**0.489**
Ours(Sp-Df-p-end)	**0.932**	**0.050**	**0.041**	0.905	1.600	0.503
Ours(Sp-Df-fp-end)	0.910	0.054	0.048	**0.917**	**1.062**	0.504

In terms of other baselines, we compare `DeepFEPE` with Si-models and Si-base in Tab. 4.4. `DeepFEPE` achieves better mean translation and rotation error compared to Si-base, and comparable performance with Si-models. The table demonstrates that the `DeepFEPE` model sets up the new state-of-the-art for learning-based relative pose estimation against DeepF. The qualitative results are shown in Fig. 4.3 and Fig. 4.4, comparing Si-base, Si-model and `DeepFEPE`. Pose estimation is visualized by comparing the epipolar lines projected from ground

Table 4.4. Comparison of pose estimation for SIFT-based KITTI models on KITTI dataset. The table compares our DeepFEPE model with Si-base and Si-models for pose estimation. Our model works better than Si-base, and comparable with the state-of-the-art Si-models. (Refer to Tab. 4.1 for acronyms.)

KITTI Models	KITTI dataset - error(deg.) inlier ratio↑, mean↓, median↓					
	Rotation (deg.)			Translation (deg.)		
	0.1↑	Mean.↓	Med.↓	2.0↑	Mean.↓	Med.↓
Base(Si-Ran)	0.818	0.391	0.056	0.899	1.895	0.639
Si-Df-f	0.938	**0.051**	0.041	0.914	1.699	**0.484**
Si-Df-p	0.901	0.059	0.044	0.903	1.472	0.513
Si-Df-fp	**0.947**	0.111	**0.038**	0.916	1.741	0.484
Ours(Sp-Df-fp-end)	0.910	0.054	0.048	**0.917**	**1.062**	0.504

Table 4.5. Comparison of pose estimation for learning-based KITTI models on Apollo dataset. The table compares the learning-based approaches in a cross-dataset setting. The table shows that our end-to-end DeepFEPE performs the best.

KITTI Models	Apollo dataset - error(deg.) inlier ratio↑, mean↓, median↓					
	Rotation (deg.)			Translation (deg.)		
	0.1↑	Mean.↓	Med.↓	2.0↑	Mean.↓	Med.↓
Base(Sp-Ran)	0.407	0.205	0.118	0.583	5.645	1.670
Sp-Df-f	0.725	0.126	0.068	0.754	2.074	1.155
Sp-Df-p	0.730	0.124	0.067	0.827	1.905	0.974
Ours(Sp-Df-f-end)	0.841	0.100	0.051	0.910	**1.122**	**0.589**
Ours(Sp-Df-p-end)	0.686	0.152	0.071	0.747	2.652	1.068
Ours(Sp-Df-fp-end)	**0.864**	**0.092**	**0.051**	**0.924**	1.275	0.659

truth and estimated fundamental matrices. If the estimated one is close to ground truth, the vanishing point should match that of ground truth. Keypoints with high weights predicted by Pose Estimation (PE) are also plotted for reasoning the relation of point distribution and pose estimation.

Due to the fact that learning-based methods are biased towards the training data, we evaluate the models trained from KITTI on ApolloScape dataset. The results demonstrate that our model retains generalization ability and is less prone to overfitting. From Tab. 4.5, we compare DeepFEPE model with Sp-base models and observe the benefit from end-to-end training with lower rotation and translation error. Without end-to-end training, the Sp-base models degrade significantly (in Tab. 4.3) and are won over by end-to-end models by a large margin. Comparing to other baselines in Tab. 4.6, we observe that the Si-base demonstrates the highest accuracy, and DeepFEPE achieves better mean rotation and translation error over Si-models.

Table 4.6. Comparison of pose estimation for SIFT-based KITTI models on Apollo dataset. The table compares our DeepFEPE with other baseline methods in a cross-dataset setting. The results show that Si-base maintains the best overall results, and DeepFEPE performs better than Si-models.

KITTI Models	Apollo dataset - error(deg.) inlier ratio↑, mean↓, median↓					
	Rotation (deg.)			Translation (deg.)		
	0.1↑	Mean.↓	Med.↓	2.0↑	Mean.↓	Med.↓
Base(Si-Ran)	**0.922**	0.157	**0.037**	**0.979**	**0.788**	0.388
Si-Df-f	0.845	0.172	0.043	0.895	2.452	0.389
Si-Df-p	0.727	0.333	0.056	0.760	4.918	0.658
Si-Df-fp	0.840	0.148	0.044	0.911	2.103	**0.369**
Ours(Sp-Df-fp-end)	0.864	**0.092**	0.051	0.924	1.275	0.659

To further examine the benefit of geometry-based loss, we can look into Tab. 4.3, with 3 models trained on either **F-loss**, **pose-loss** or both. We can observe the model trained using both losses achieves significantly better mean translation error. We believe this is because the geometric information incorporated in **pose-loss** encourages the keypoint distribution in FE to be pose-aware. The keypoints are updated to balance between good localization accuracy and matching w.r.t. the pose estimation. The change of keypoint distribution is observed from Fig. 4.5. This shows the potential of having a robust and optimized feature extractor with end-to-end training. As observed from the figure, keypoints close to the vanishing point are reduced after the end-to-end training. It is because these points are good for matching but may incur high triangulation errors when solving for camera pose, due to their little motion from frame to frame. On the other hand, points near the border of the image see a noticeable increase. These points may not be robustly matched with conventional descriptors because of large motion and in some case motion blur. On the contrary, our method is able to reveal these points which provide wider baseline for more accurate camera pose estimation.

4.3.3 SuperPoint Correspondence Estimation

To understand how Feature Extraction (FE) module is updated after training, we collect quantitative results using Sampson distance and demonstrate keypoint distribution qualitatively. For each pair of correspondences, we calculate the Sampson distance from Eq. (4.3), which

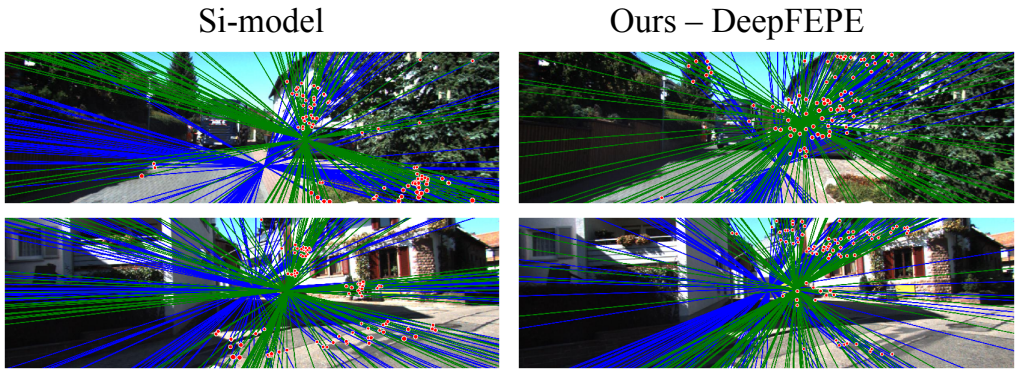


Figure 4.4. Failure cases of pose estimation. Compare Si-model with our `DeepFEPE`. The failure cases include over-exposed and textureless scenes. The challenging views result in noisy correspondences and wrong prediction. (Lines and dots are plotted as in Fig. 4.3.)

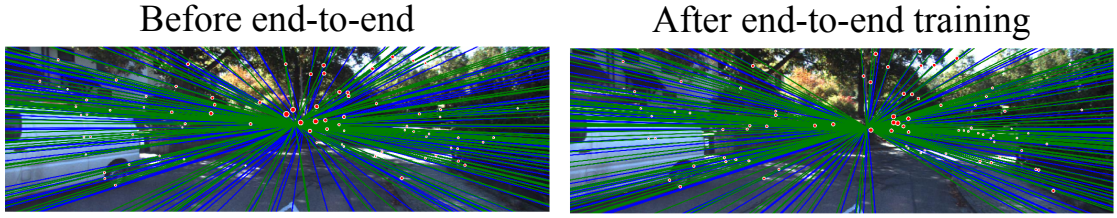


Figure 4.5. Change of keypoint distribution after end-to-end training. In our pipeline, we train feature extractor and pose estimation end-to-end. To show the qualitative results of feature extractor, we freeze pose estimation module and update only feature extractor. The results show that the change of distribution helps the pose estimation predict better pose. (Lines and dots are plotted as in Fig. 4.3.)

Table 4.7. Superpoint evaluation. When training end-to-end, we can see the increase of inlier ratio w.r.t. Sampson distance and number of correspondences.

	KITTI - epiploar dists (n pixels), num of matches: mean/ med.					
KITTI Models	0.2	0.5	1.0	2.0	Mean	Med.
Sp-Ran	0.080	0.195	0.361	0.581	541.546	533.000
Sp-Df-f-end	0.107	0.258	0.460	0.685	719.986	703.000
Sp-Df-p-end	0.096	0.232	0.421	0.643	626.343	611.000
Sp-Df-fp-end	0.105	0.254	0.453	0.677	669.170	654.000

indicates whether the pair of points lies close to each other's epipolar line. We show the inlier ratio w.r.t. different distance value (unit: pixel) from 0.2 to 2, as well as the number of correspondences in Tab. 4.7. The results show an increase of inlier ratio up to 10% with Sampson distance below 1px on KITTI dataset. The number of correspondences also increases by around 20%. The result shows that the end-to-end training improves the individual module as well. The model trained on **F-loss** has the best result under this metric, as the **F-loss** minimizes the energy in epipolar space.

4.4 Conclusion

In this chapter we propose an end-to-end trainable pipeline for estimating camera poses from input image pairs. We demonstrate that our performance is on par with classic methods, and superior generalization ability to unseen data compared with other existing baselines. Both qualitative and quantitative results are included in the chapter to support the claim. We provide further insights into the benefits that end-to-end training brings into keypoint detection, feature extraction and pose estimation. Future work of this may include sequential input or keyframes with long-term temporal cues. Experiments on other datasets with different motion patterns than the driving datasets in the chapter can also be explored.

In Ch. 4, in full, has been submitted for publication of the material as it may appear in Conference on Intelligent Robots and Systems (IROS), 2020. You-Yi Jau, Rui Zhu, Hao Su, Manmohan Chandraker. The thesis author was the primary investigator and author of this

paper.

Chapter 5

Deep Learning-based Method for Visual Odometry

In this chapter, we examine an existing deep learning-based method on outdoor and indoor dataset. The development of deep learning-based method starts from SfMLearner [85], which claims better performance than short version of ORB-SLAM. Among papers after that [10, 28, 39, 44, 78, 81], we pick SC-SfMLearner [10] as our basic framework. The paper uses similar network architectures and loss functions as SfMLearner, but with an additional loss to achieve scale consistency. We use the released code from [10] [1] as our starting point. Ch. 5.1 gives an overview to the pipeline for SC-SfMLearner. The experiments are performed on different environments in Ch. 5.2, including KITTI [26] and EuRoC [13] datasets. With quantitative and qualitative results shown in Ch. 5.2, the future work is listed in Ch. 5.3.

5.1 Overview to SC-SfMLearner

5.1.1 Pipeline

The overview to SC-SfMLearner is shown in Ch. 5.1. A pair of images, $\mathbf{I}_a$ and $\mathbf{I}_b$, are fed into PoseNet to predict the relative pose $\mathbf{P}_{ab}$. The image can be fed into DepthNet for single image depth prediction. The supervision signals come from the projection flow from $\mathbf{I}_a$ to $\mathbf{I}_b$ using the predicted depth and pose. The novelty of this work is the geometric consistency loss,

[1] https://github.com/JiawangBian/SC-SfMLearner-Release

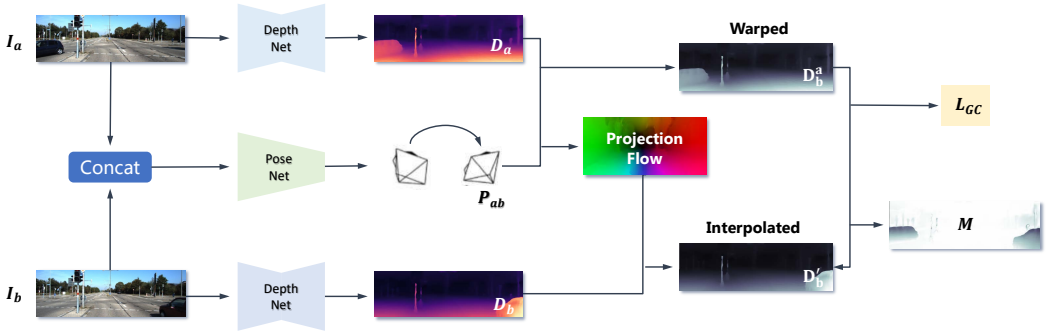


Figure 5.1. SC-SfMLearner pipeline. This figure is from the original paper [10].

$\mathbf{L}_{GC}$, calculated from the predicted depth images.

The prediction from the paper [10] in Fig. 5.2 shows competitive results against ORB-SLAM on KITTI dataset. It demonstrates low drifting in the global trajectory.

5.1.2 Network design

PoseNet

The architecture for PoseNet follows the design from SfMLearner, but without the decoder structure. The architecture is shown in Tab. 5.1. The output from `pose_pred` layer

Table 5.1. PoseNet architecture. (Refer to Ch. 5.1 for description.)

Layer name	Output size	Architecture (kernel, depth, stride)
Input	$H \times W$	
conv1	$H/2 \times W/2$	$(7 \times 7, 16, 2)$
conv2	$H/4 \times W/4$	$(5 \times 5, 32, 2)$
conv3	$H/8 \times W/8$	$(3 \times 3, 64, 2)$
conv4	$H/16 \times W/16$	$(3 \times 3, 128, 2)$
conv5	$H/32 \times W/32$	$(3 \times 3, 256, 2)$
conv6	$H/64 \times W/64$	$(3 \times 3, 256, 2)$
conv7	$H/128 \times W/128$	$(3 \times 3, 256, 2)$
pose_pred	$H/128 \times W/128$	$(1 \times 1, 6, 1)$

is pooled into pose vector of size $1 \times 1 \times 6$ through H, W channels, which is the euler angle representation of the relative pose. The vector is then multiplied by a scale factor α, which is set to 0.01.

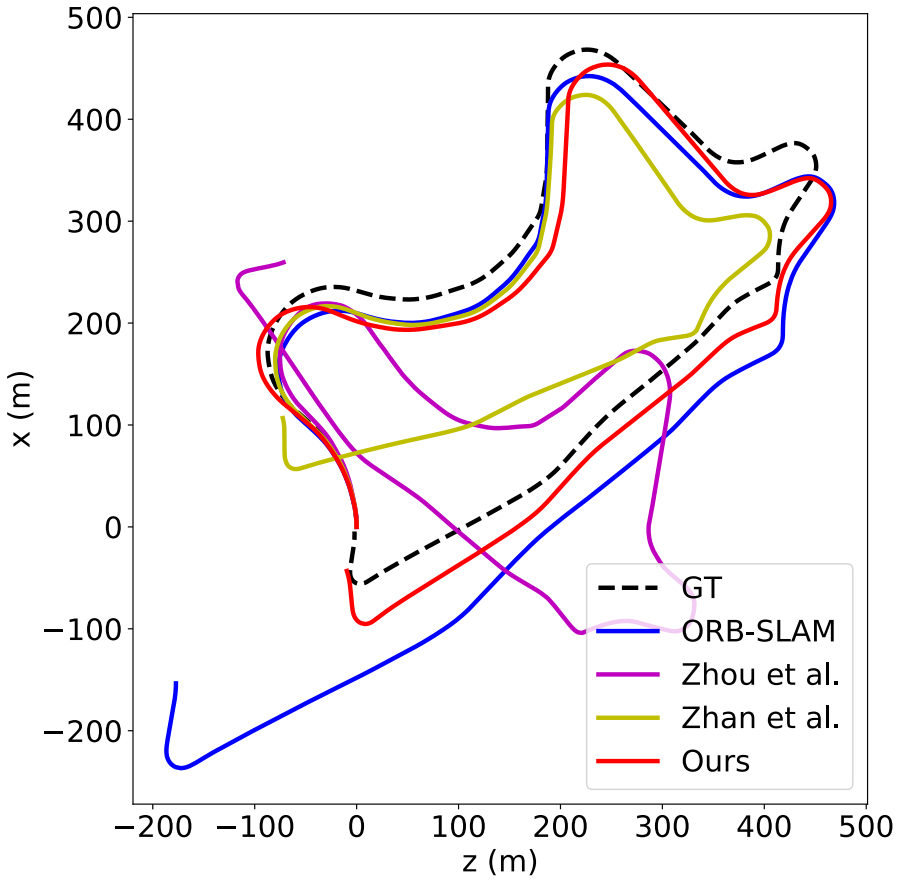


Figure 5.2. SC-SfMLearner prediction on KITTI sequence 09. This figure is from the original paper [10].

DepthNet

Among the depth or disparity networks, DispNet used in SfMLearner [10] and DispResNet in [57], we follow the choice of DispResNet as in [10]. The architecture is shown in Tab. 5.2. The convolutional layers are mostly consisted of residual blocks with internal skip connections. Upsampling is done through transpose convolution ($UpConv$). The layers with same depth are connected through *iConv*, where *conv* and $UpConv$ are concatenated and fed into *iConv*. The output layer is passed through convolutional layer with depth 1 and sigmoid activation function, and is linearly transformed by $\alpha * x + \beta$, with $\alpha = 10$ and $\beta = 0.01$ in our setting.

Table 5.2. DepthNet architecture. The network is proposed by [57]. (Refer to Ch. 5.1 for description.)

Layer name	Output size	Architecture (kernel, depth, stride)
Input	$H \times W$	
conv1	$H/2 \times W/2$	$(7 \times 7, 32, 2), (7 \times 7, 32, 1)$
conv2	$H/4 \times W/4$	Residual: $[(3 \times 3, 64, 2)] \times 2$
conv3	$H/8 \times W/8$	Residual: $[(3 \times 3, 128, 2)] \times 2$
conv4	$H/16 \times W/16$	Residual: $[(3 \times 3, 256, 2)] \times 3$
conv5	$H/32 \times W/32$	Residual: $[(3 \times 3, 512, 2)] \times 3$
conv6	$H/64 \times W/64$	Residual: $[(3 \times 3, 512, 2)] \times 3$
conv7	$H/128 \times W/128$	Residual: $[(3 \times 3, 512, 2)] \times 3$
Upconv7	$H/64 \times W/64$	$(3 \times 3, 512, 2)$
iConv7	$H/64 \times W/64$	Residual: $[(3 \times 3, 512, 1)] \times 2$
Upconv6	$H/32 \times W/32$	$(3 \times 3, 512, 2)$
iConv6	$H/32 \times W/32$	Residual: $[(3 \times 3, 512, 1)] \times 2$
Upconv5	$H/16 \times W/16$	$(3 \times 3, 256, 2)$
iConv5	$H/16 \times W/16$	Residual: $[(3 \times 3, 256, 1)] \times 2$
Upconv4	$H/8 \times W/8$	$(3 \times 3, 128, 2)$
iConv4	$H/8 \times W/8$	Residual: $[(3 \times 3, 128, 1)] \times 2$
Upconv3	$H/4 \times W/4$	$(3 \times 3, 64, 2)$
iConv3	$H/4 \times W/4$	Residual: $[(3 \times 3, 64, 1)] \times 1$
Upconv2	$H/2 \times W/2$	$(3 \times 3, 32, 2)$
iConv2	$H/2 \times W/2$	Residual: $[(3 \times 3, 32, 1)] \times 1$
Upconv1	$H \times W$	$(3 \times 3, 16, 2)$
iConv1	$H \times W$	Residual: $[(3 \times 3, 16, 1)] \times 1$
predict_disp	$H \times W$	$(3 \times 3, 1, 1), Sigmoid$

5.1.3 Loss functions

We follow the original design in SC-SfMLearner [10]. Here is the description of the loss functions. Details can be found in [10].

Photometric loss

Photometric loss is the key to train the DepthNet and PoseNet without ground truth supervision. It is based on the assumption that the corresponding points have the same intensity on the two frames. It can be formulated as follows.

$$L_p = \frac{1}{|V|} \sum_{\mathbf{p} \in V} ||\mathbf{I}_a(\mathbf{p}) - \mathbf{I}'_a(\mathbf{p})||_1, \tag{5.1}$$

where $\mathbf{I}_a$ is the target image and $\mathbf{I}'_a$ is the image warped to the target image using predicted pose and depth. V is the set for valid points $\mathbf{p}$. L1 loss enforces the consistency. The photometric loss is added by a dissimilarity of Structural Similarity (SSIM) term [72]. The term normalizes the pixel intensities, which leads to robustness of change of illumination.

Smoothness loss

The edge-aware smoothness loss is utilized to regularize the depth prediction. It can help handle regions with low texture.

Geometry consistency loss

The geometry consistency loss proposed by SC-SfMLearner can be formulated as

$$L_{GC} = \frac{1}{|V|} \sum_{\mathbf{p} \in V} D_{diff}(\mathbf{p}), \tag{5.2}$$

where $D_{diff}(\mathbf{p})$ records the relative differences between warped and interpolated depth prediction at point $\mathbf{p}$. It is a pixel-wise alignment between two depth images.

5.2 Experiments of SC-SfMLearner on Various Datasets

To identify the limitation of SC-SfMLearner, we have the models trained on KITTI [26] and EuRoC [13] dataset individually and perform cross dataset evaluation. The quantitaive and

qualitative results are shown in Tab. 5.3, Tab. 5.4, Fig. 5.4, Fig. 5.5, Fig. 5.6, and Fig. 5.7. As the experiments shown in the SC-SfMLearner paper [10], the prediction for KITTI dataset is accurate visually (Fig. 5.4) with high accuracy in ATE (Tab. 5.3) comparing with that in ORB-SLAM-VO (Tab. 3.1). However, when evaluating the model on EuRoC [13] dataset, which is in indoors, the error jumps high (Tab. 5.4) compared to ORB-SLAM-VO prediction (Tab. 3.2). The trajectories are also far from the ground truth, as shown in Fig. 5.5.

5.2.1 Implementation Details

Following the setting in SC-SfMLearner, we train the model on KITTI dataset using batch size 4 for 200 epochs. However, the same setting didn't work for EuRoC dataset. When using the same batch size, the depth model tends to predict zeros, which makes the smoothness loss goes to zero. Since the camera baselines for image pairs in EuRoC dataset are small, we increase the skipped interval in a pair of images. We increase the interval from 1 (consecutive) to 8 timestamps, and set the batch size to 8. The setting trains the depth and pose model well. After training for 100 epochs, we train 200 more epochs on the setting with consecutive frames for fine-tuning.

5.2.2 Domain Gap

The domain gap between KITTI and EuRoC datasets is large. First, the KITTI dataset is collected by a vehicle outdoors, whereas EuRoC dataset is collected by an MAV indoors. The scale of KITTI scenes is around hundred meters, compared to several meters in EuRoC sequences. The trajectory in KITTI contains most variations in x, y-axis (in world coordinate), but MAV can fly around in 3 dimensions. Second, the camera in KITTI dataset has wide view angles with RGB channels, whereas EuRoC has narrow view angles with gray images.

The EuRoC dataset originally has radial distortion, which makes straight lines to be curved in an image. We undistorted the images using given parameters and visualized in a video

[2]. In experiments, we didn't see much difference using undistorted images.

5.2.3 Overfitting Issues

The overfitting problems are particularly obvious in our cross-dataset experiments, where we have poses predicted from the PoseNet. When the model trained on KITTI dataset is tested on EuRoC, it tends to predict forward motion, as shown in Fig. 5.5. On the other hand, when the model trained on EuRoC dataset is tested on KITTI sequences, it predicts rotation motion even on straight roads, *i.e.* sequence (01) and (04) in Fig. 5.6. It is because of the extensive rotations and loops in EuRoC dataset that makes the model over-react on a pure forward motion.

5.2.4 Relative Pose Prediction

If we only look at the relative pose error (RPE) on EuRoC dataset, the error is actually lower in SC-SfMLearner than that in ORB-SLAM-VO(Tab. 3.2 vs. Tab. 5.4). When we look closer to the error map of RPE in Fig. 5.3, the prediction in ORB-SLAM-VO sometimes jumps off in the trajectory. We infer that the error is due to motion blur or textureless regions. In the long run, the trajectory can be optimized through bundle adjustment (Ch. 2.1.9) on keyframes. On the contrary, the error map for SC-SfMLearner mostly lies in the blue (low error) end. It is possible that the model predicts conservative motions, which led to the low local error.

We also observe the generalization ability of the DepthNet. We visualize the prediction of the KITTI model on KITTI and EuRoC datasets, by plotting out the depth prediction and the warped images using predicted poses and depth. The models can have reasonable depth prediction. The videos are presented for KITTI dataset[3] and EuRoC dataset[4].

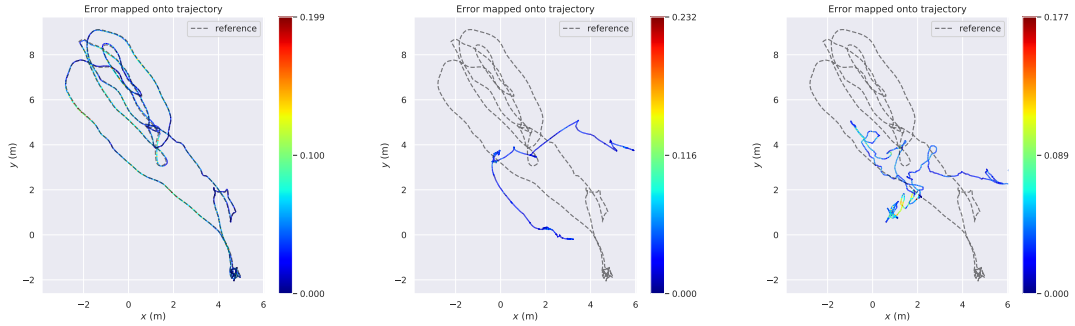
[2]https://youtu.be/8VD6Ud7IcRk
[3]https://youtu.be/8tFE6itHaeM
[4]https://youtu.be/0xP6R6Yha2c

Table 5.3. Quantitative result of SC-SfMLearner on KITTI dataset. The models are trained either on KITTI or EuRoC dataset, and tested on KITTI dataset. The metrics are described in Ch. 3.2.2.

Method	Metric	00	01	02	03	04	05	06	07	08	09	10	Avg. Err.
Trained on KITTI	t_{err}	9.490	49.059	11.127	7.669	3.336	5.445	9.293	5.092	4.774	8.304	10.178	11.252
	r_{err}	2.676	1.196	2.356	3.445	1.643	1.661	3.205	2.603	1.446	2.295	3.064	2.326
	ATE	49.123	171.636	113.621	8.018	2.393	20.333	37.556	7.035	24.953	19.850	14.486	42.637
	RPE (m)	0.133	2.276	0.171	0.097	0.148	0.087	0.110	0.052	0.085	0.130	0.106	0.309
	RPE (°)	0.121	0.095	0.111	0.089	0.074	0.078	0.079	0.083	0.083	0.115	0.127	0.096
Train on EuRoC	t_{err}	50.472	64.791	61.388	68.217	29.073	53.695	49.508	61.520	50.048	56.244	34.813	52.706
	r_{err}	25.077	13.666	28.538	42.617	24.473	32.811	26.102	51.984	25.156	27.183	24.219	29.257
	ATE	172.808	508.970	262.845	49.435	28.651	141.453	104.893	79.424	189.406	163.817	50.771	159.316
	RPE (m)	0.519	2.492	0.637	0.526	0.707	0.429	0.673	0.364	0.540	0.582	0.466	0.721
	RPE (°)	0.900	0.551	0.826	0.646	0.568	0.716	0.725	0.846	0.798	0.824	0.693	0.736

Table 5.4. Quantitative result of SC-SfMLearner on EuRoC dataset. The models are trained either on KITTI or EuRoC dataset, and tested on undistorted EuRoC sequences. The metrics are described in Ch. 3.2.2.

Method	Metric	MH01	MH02	MH04	V101	V102	V103	MH05	V201	V202	V203	MH03	Avg. Err.
Trained on KITTI	ATE	3.353	3.158	6.423	1.773	1.760	1.475	6.344	1.964	1.923	1.841	3.548	3.051
	RPE (m)	0.033	0.043	0.079	0.029	0.066	0.057	0.076	0.026	0.064	0.087	0.097	0.060
Trained on EuRoC	ATE	3.639	3.631	6.291	1.749	1.760	1.498	6.190	1.425	2.060	1.877	3.368	3.044
	RPE (m)	0.039	0.052	0.092	0.030	0.068	0.057	0.092	0.047	0.066	0.089	0.100	0.066



ORB-SLAM-VO SC-SfMLearner(KITTI) SC-SfMLearner(EuRoC)

Figure 5.3. Comparison of qualitative RPE results on the EuRoC sequence using geometry-based or deep learning-based methods.

5.3 Future work

By observing ORB-SLAM-VO and SC-SfMLearner, we identify several directions for future work in deep learning-based systems. Among the key factors for ORB-SLAM in Ch. 3.3, optimization, keyframe-based system, and long feature tracking are discussed as follows.

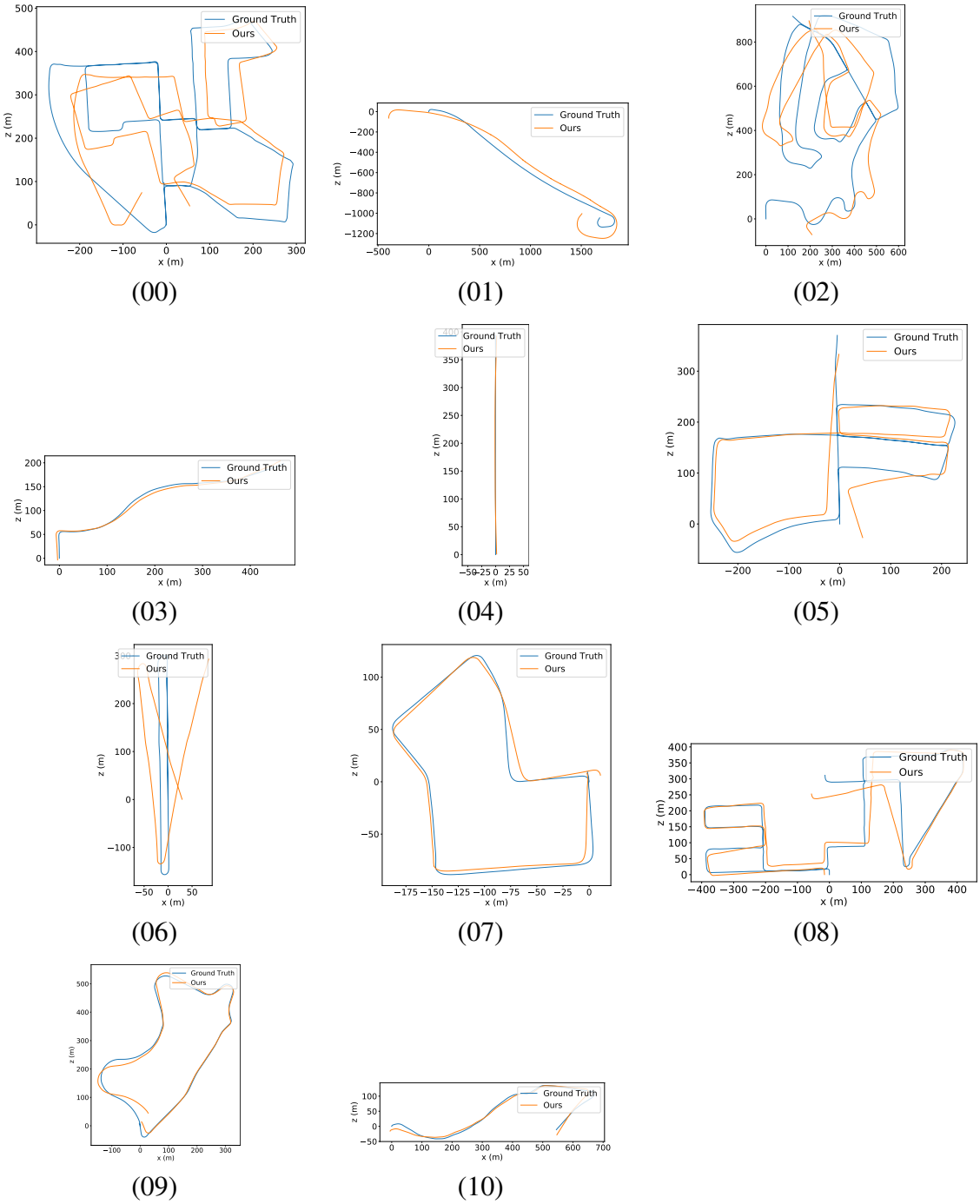


Figure 5.4. Qualitative VO results for KITTI model on KITTI dataset for SC-SfMLearner. Ground truth trajectories are plotted in blue, while estimated ones are in orange. The trajectories are aligned in 7 DoF.

5.3.1 Optimization

Given the reasonable relative pose estimation but poor global trajectory, we identify that optimization across frames can correct the drifting. In contrast of the geometric bundle adjustment in ORB-SLAM [47], the setting can be similar to DSO [19] using photometric optimization.

There are some works [41, 76] plugging in the deep learning modules in geometry-based method like DSO [19], and achieve high accuracy. However, the optimization is mainly based on the original geometry-based designs [19, 23], where the deep networks only provide a good initial pose, depth, or uncertainty. BA-Net [62] makes the bundle adjustment differentialable in the pipeline. However, BA-Net is still far from applicable for a robust visual odometry system. It can be an interesting path to explore how to incorporate the optimization tightly in the deep learning-based pipeline.

The robust bundle adjustment designed in ORB-SLAM [47] includes local BA and pose graph optimization. The combination of local and global optimization can correct the drifting accumulated during the estimation.

5.3.2 Keyframe-Based System

As keyframes are essential for robust estimation in ORB-SLAM, benefits are also observed in training SC-SfMLearner. In our experiments (Ch. 5.2), the DepthNet can be properly trained with an interval between a pair of frames with larger baselines. The setting of intervals between pairs can be decided on-the-fly instead of being fixed. It is related to the topics of curriculum learning.

5.3.3 Long Feature Tracking

As mentioned in Ch. 3.3, geometric bundle adjustment takes effect when the long feature tracking is available. The correspondences across frames are strong cues to optimize the estimated poses. In keypoint-based methods, the features are robust to illumination. However, the

illumination change can fail the basic assumption of constant intensity across frames for direct methods. In D3VO [76], the change of illumination is formulated as an affine transformation predicted by the network. In BA-Net [62], the features from the intermediate layer are used to replace the raw image, which leads to better optimization. The path is yet to explore how to enable the long feature tracking in the setting without sparse feature extraction and matching.

5.4 Conclusion

In this chapter, we pick SC-SfMLearner as the template for deep learning-based visual odometry method and analyze the strengths and weaknesses. In the experiments in outdoor and indoor environments, the deep learning models suffer from various issues. We identify the domain gap between KITTI and EuRoC datasets and the overfitting issues observed from the plots. However, from the quantitative results of relative poses, we point out the possibility of increasing accuracy with cross frame optimization. The benefits of keyframe-based system and long feature tracking in ORB-SLAM are also paths to explore.

In Ch. 5, in part is currently being prepared for submission for publication of the material. You-Yi Jau, Manmohan Chandraker. The thesis author was the primary investigator and author of this paper.

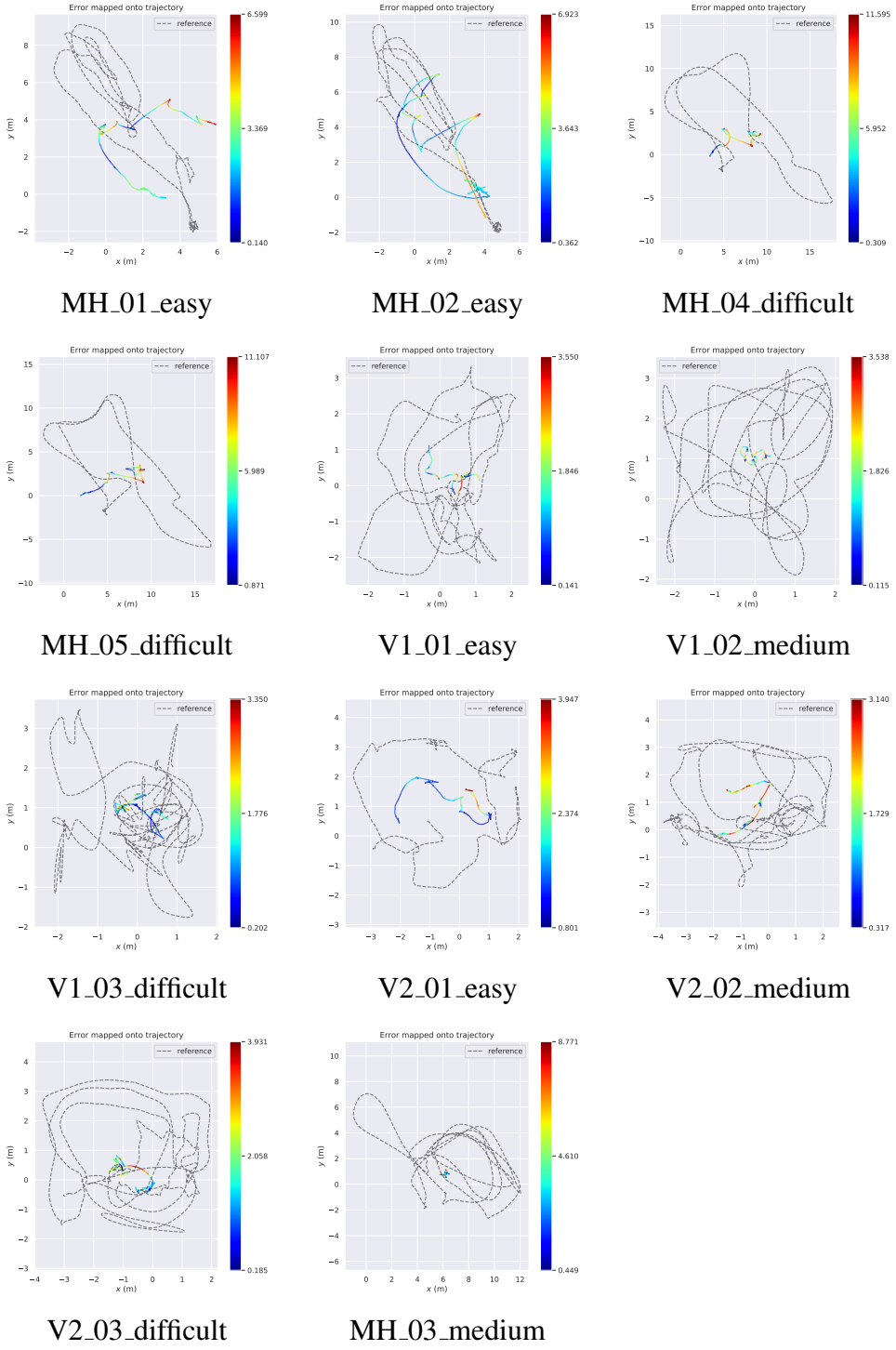


Figure 5.5. Qualitative VO results for KITTI model on EuRoC dataset for SC-SfMLearner. The ground truth trajectories are plotted in dotted lines, whereas the estimated ones are mapped in colors with error. The trajectories are aligned in 7 DoF.

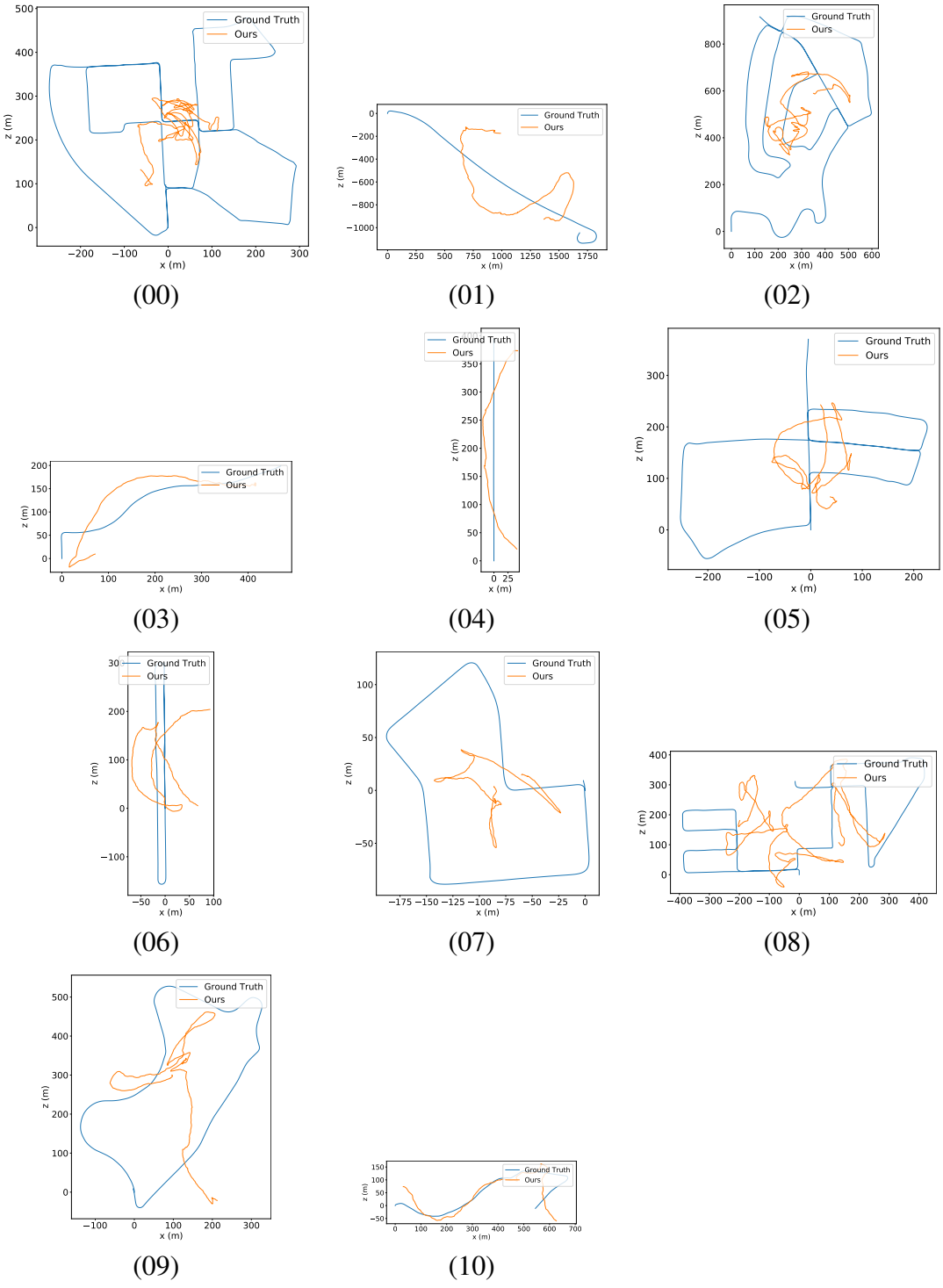
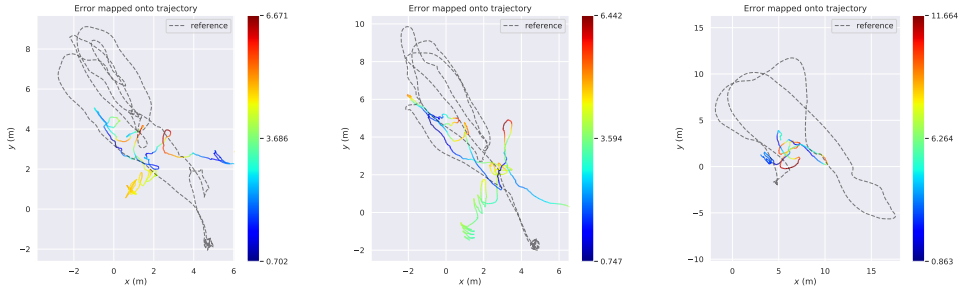


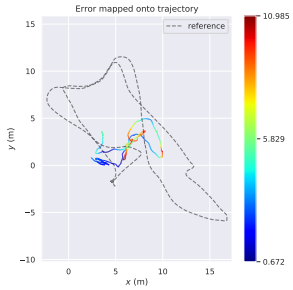
Figure 5.6. Qualitative VO results for EuRoC model on KITTI dataset for SC-SfM Learner.



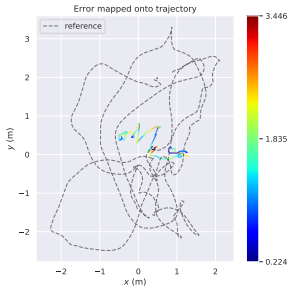
MH_01_easy

MH_02_easy

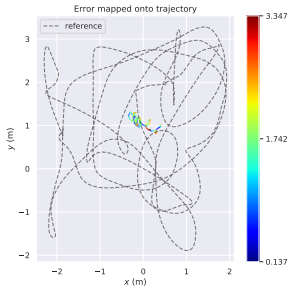
MH_04_difficult



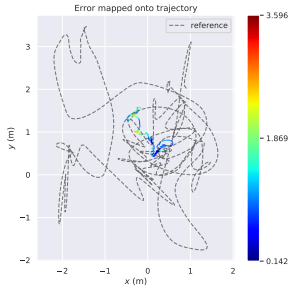
MH_05_difficult



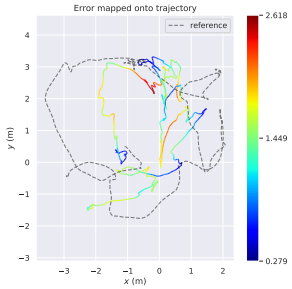
V1_01_easy



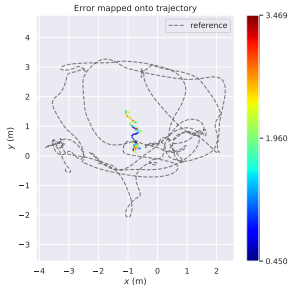
V1_02_medium



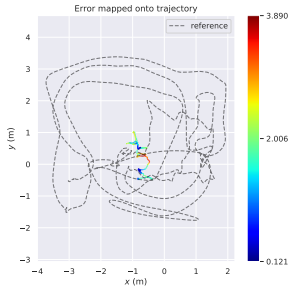
V1_03_difficult



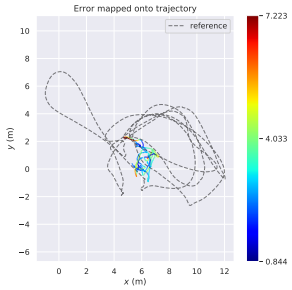
V2_01_easy



V2_02_medium



V2_03_difficult



MH_03_medium

Figure 5.7. Qualitative VO results for EuRoC model on EuRoC dataset for SC-SfM Learner.

Chapter 6

Conclusion

In this thesis, we look into camera pose estimation, visual odometry and SLAM, which are the key techniques in the applications of VR, AR, and autonomous driving. In Ch. 2, we state the problem formulation and have comprehensive introduction of visual odometry pipeline, as well as the recent development in the field. In Ch. 3, we look into a realistic SLAM system and identify the key designs. Inspired by the work, we propose a deep learning-based camera pose estimation in Ch. 4. To extend the work to deep learning-based VO or SLAM, we look into an existing work in Ch. 5 and identify the strengths and weaknesses. The main achievements are:

- An analysis for geometry-based system, ORB-SLAM, with thorough quantitative and qualitative results in indoor and outdoor datasets. The key factors are identified for the design of a robust SLAM system. (Ch. 3)
- A deep learning-based camera pose estimation incorporating geometric constraints. The pipeline is optimized for feature extraction, matching, outlier rejection and pose estimation in an end-to-end fashion. (Ch. 4)
- A deep learning-based pipeline adopting the advantages of correspondences and epipolar geometry. The pipeline has the benefits of keypoint-based method, and is optimized for the geometry-based objective. (Ch. 4)
- A thorough study of our deep learning-based method in a cross-dataset setting. Our

approach has better generalization ability than the learning-based baselines. (Ch. 4)

- An exploration of a learning-based system, SC-SfMLearner, with cross-dataset setting. The limitations are demonstrated, while future directions are pointed out. (Ch. 5)

6.1 Future Direction

In the field of visual odometry, deep learning-based methods show the potential, whereas the geometry-based methods still achieve better robustness. Recent work shows better performance by leveraging the advantages of both methods. However, the power of deep learning-based methods has yet to be discovered. We look into the geometry-based pipeline and identify the successful factors in Ch. 3.3. These factors bring robustness to the geometry-based methods. They also give ways for deep learning-based methods to improve for. In Ch. 4, we propose a deep learning-based pipeline for camera pose estimation. This pipeline can be further enhanced by replacing the modules with strong feature extraction or pose estimation modules. The pipeline can also be extended into visual odometry system. In Ch. 5, we point out the weaknesses of existing deep learning-based visual odometry. We refer to the key factors in geometry-based methods, and outline the future works in Ch. 5.3.

Standing on the wave of deep learning, I believe the benefits of geometry can not be forgotten. Instead, they point out the paths to a robust visual odometry system. By looking into the past and presence, we imagine and commit to the prospective future of visual odometry.

Bibliography

[1] GPS.gov: GPS Accuracy.

[2] Axis–angle representation, May 2020.

[3] Euler angles, May 2020.

[4] Quaternion, Apr 2020.

[5] Rodrigues' rotation formula, Apr 2020.

[6] S. Agarwal, N. Snavely, I. Simon, S. M. Seitz, and R. Szeliski. Building rome in a day. In *2009 IEEE 12th International Conference on Computer Vision*, pages 72–79, 2009.

[7] Vassileios Balntas, Karel Lenc, Andrea Vedaldi, and Krystian Mikolajczyk. Hpatches: A benchmark and evaluation of handcrafted and learned local descriptors. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[8] Vassileios Balntas, Shuda Li, and Victor Prisacariu. RelocNet: Continuous Metric Learning Relocalisation Using Neural Nets. In Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss, editors, *Computer Vision – ECCV 2018*, volume 11218, pages 782–799. Springer International Publishing, Cham, 2018.

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

[10] Jiawang Bian, Zhichao Li, Naiyan Wang, Huangying Zhan, Chunhua Shen, Ming-Ming Cheng, and Ian Reid. Unsupervised scale-consistent depth and ego-motion learning from monocular video. In H. Wallach, H. Larochelle, A. Beygelzimer, F. dAlché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 35–45. Curran Associates, Inc., 2019.

[11] Eric Brachmann, Alexander Krull, Sebastian Nowozin, Jamie Shotton, Frank Michel, Stefan Gumhold, and Carsten Rother. Dsac-differentiable ransac for camera localization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6684–6692, 2017.

[12] Eric Brachmann and Carsten Rother. Learning less is more-6d camera localization via 3d surface regression. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4654–4662, 2018.

[13] Michael Burri, Janosch Nikolic, Pascal Gohl, Thomas Schneider, Joern Rehder, Sammy Omari, Markus W Achtelik, and Roland Siegwart. The EuRoC micro aerial vehicle datasets. *The International Journal of Robotics Research*, 35(10):1157–1163, September 2016.

[14] Olivier Chapelle and Mingrui Wu. Gradient descent optimization of smoothed information retrieval metrics. *Information Retrieval*, 13(3):216–235, June 2010.

[15] Peter Hviid Christiansen, Mikkel Fly Kragh, Yury Brodskiy, and Henrik Karstoft. UnsuperPoint: End-to-end Unsupervised Interest Point Detector and Descriptor. *arXiv:1907.04011 [cs]*, July 2019. arXiv: 1907.04011.

[16] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Self-Improving Visual Odometry. *arXiv:1812.03245 [cs]*, December 2018. arXiv: 1812.03245.

[17] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superpoint: Self-supervised interest point detection and description. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2018.

[18] Ethan Eade. Lie groups for 2d and 3d transformations. *URL http://ethaneade. com/lie. pdf, revised Dec*, 2013.

[19] J. Engel, V. Koltun, and D. Cremers. Direct sparse odometry. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(3):611–625, March 2018.

[20] Jakob Engel, Thomas Schöps, and Daniel Cremers. LSD-SLAM: Large-Scale Direct Monocular SLAM. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Computer Vision – ECCV 2014*, Lecture Notes in Computer Science, pages 834–849. Springer International Publishing, 2014.

[21] Jakob Engel, Jurgen Sturm, and Daniel Cremers. Semi-dense Visual Odometry for a Monocular Camera. In *2013 IEEE International Conference on Computer Vision*, pages 1449–1456, Sydney, Australia, December 2013. IEEE.

[22] Martin A. Fischler and Robert C. Bolles. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. *Commun. ACM*, 24(6):381–395, June 1981.

[23] Christian Forster, Matia Pizzoli, and Davide Scaramuzza. SVO: Fast semi-direct monocular visual odometry. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 15–22, Hong Kong, China, May 2014. IEEE.

[24] F. Fraundorfer and D. Scaramuzza. Visual odometry : Part ii: Matching, robustness, optimization, and applications. *IEEE Robotics Automation Magazine*, 19(2):78–90, 2012.

[25] A Geiger, P Lenz, C Stiller, and R Urtasun. Vision meets robotics: The KITTI dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, September 2013.

[26] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013.

[27] Andreas Geiger, Julius Ziegler, and Christoph Stiller. Stereoscan: Dense 3d reconstruction in real-time. In *Intelligent Vehicles Symposium (IV)*, 2011.

[28] Clément Godard, Oisin Mac Aodha, Michael Firman, and Gabriel J Brostow. Digging into self-supervised monocular depth estimation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3828–3838, 2019.

[29] Michael Grupp. evo: Python package for the evaluation of odometry and slam. https://github.com/MichaelGrupp/evo, 2017.

[30] Bert M. Haralick, Chung-Nan Lee, Karsten Ottenberg, and Michael Nölle. Review and analysis of solutions of the three point perspective pose estimation problem. *International Journal of Computer Vision*, 13(3):331–356, December 1994.

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

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

[33] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. 2004. OCLC: 171123855.

[34] Xinyu Huang, Peng Wang, Xinjing Cheng, Dingfu Zhou, Qichuan Geng, and Ruigang Yang. The ApolloScape Open Dataset for Autonomous Driving and its Application. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–1, 2019. arXiv: 1803.06184.

[35] Rong Kang, Jieqi Shi, Xueming Li, Yang Liu, and Xiao Liu. DF-SLAM: A Deep-Learning Enhanced Visual SLAM System based on Deep Local Features. *arXiv:1901.07223 [cs]*, January 2019. arXiv: 1901.07223.

[36] Alex Kendall, Matthew Grimes, and Roberto Cipolla. Posenet: A convolutional network for real-time 6-dof camera relocalization. In *The IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[37] Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. Siamese neural networks for one-shot image recognition. In *ICML deep learning workshop*, volume 2, 2015.

[38] Vincent Lepetit, Francesc Moreno-Noguer, and Pascal Fua. EPnP: An Accurate O(n) Solution to the PnP Problem. *International Journal of Computer Vision*, 81(2):155–166, February 2009.

[39] Ruihao Li, Sen Wang, Zhiqiang Long, and Dongbing Gu. UnDeepVO: Monocular Visual Odometry Through Unsupervised Deep Learning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7286–7291, May 2018. ISSN: 2577-087X.

[40] Y. Li, Y. Ushiku, and T. Harada. Pose graph optimization for unsupervised monocular visual odometry. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 5439–5445, May 2019.

[41] S. Y. Loo, A. J. Amiri, S. Mashohor, S. H. Tang, and H. Zhang. Cnn-svo: Improving the mapping in semi-direct visual odometry using single-image depth prediction. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 5218–5223, 2019.

[42] David G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, 60(2):91–110, November 2004.

[43] Quang-Tuan Luong, Rachid Deriche, Olivier Faugeras, and Théodore Papadopoulo. On determining the fundamental matrix : analysis of different methods and experimental results. Research Report RR-1894, INRIA, 1993.

[44] Reza Mahjourian, Martin Wicke, and Anelia Angelova. Unsupervised Learning of Depth and Ego-Motion From Monocular Video Using 3d Geometric Constraints. pages 5667–5675, 2018.

[45] Kwang Moo Yi, Eduard Trulls, Yuki Ono, Vincent Lepetit, Mathieu Salzmann, and Pascal Fua. Learning to find good correspondences. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[46] Hans Peter Moravec. *Obstacle Avoidance and Navigation in the Real World by a Seeing Robot Rover*. PhD thesis, Stanford, CA, USA, 1980. AAI8024717.

[47] Raul Mur-Artal, J. M. M. Montiel, and Juan D. Tardos. ORB-SLAM: A Versatile and Accurate Monocular SLAM System. *IEEE Transactions on Robotics*, 31(5):1147–1163, October 2015.

[48] Raul Mur-Artal and Juan D. Tardos. ORB-SLAM2: an Open-Source SLAM System for Monocular, Stereo and RGB-D Cameras. *IEEE Transactions on Robotics*, 33(5):1255–1262, October 2017. arXiv: 1610.06475.

[49] Richard A. Newcombe, Steven J. Lovegrove, and Andrew J. Davison. DTAM: Dense tracking and mapping in real-time. In *2011 International Conference on Computer Vision*, pages 2320–2327, November 2011. ISSN: 2380-7504, 1550-5499, 1550-5499.

[50] D. Nister. An efficient solution to the five-point relative pose problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(6):756–770, 2004.

[51] D. Nister, O. Naroditsky, and J. Bergen. Visual odometry. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, volume 1, pages I–I, 2004.

[52] Jorge Nocedal and Stephen Wright. *Numerical optimization*. Springer Science & Business Media, 2006.

[53] Yuki Ono, Eduard Trulls, Pascal Fua, and Kwang Moo Yi. LF-Net: Learning Local Features from Images. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 6237–6247. Curran Associates, Inc., 2018.

[54] Omid Poursaeed, Guandao Yang, Aditya Prakash, Qiuren Fang, Hanqing Jiang, Bharath Hariharan, and Serge Belongie. Deep fundamental matrix estimation without correspondences. In *The European Conference on Computer Vision (ECCV) Workshops*, September 2018.

[55] Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[56] René Ranftl and Vladlen Koltun. Deep Fundamental Matrix Estimation. In Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss, editors, *Computer Vision – ECCV 2018*, volume 11205, pages 292–309. Springer International Publishing, Cham, 2018.

[57] Anurag Ranjan, Varun Jampani, Lukas Balles, Kihwan Kim, Deqing Sun, Jonas Wulff, and Michael J Black. Competitive collaboration: Joint unsupervised learning of depth, camera motion, optical flow and motion segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 12240–12249, 2019.

[58] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. ORB: An efficient alternative to SIFT or SURF. In *2011 International Conference on Computer Vision*, pages 2564–2571, November 2011. ISSN: 2380-7504, 1550-5499, 1550-5499.

[59] D. Scaramuzza and F. Fraundorfer. Visual odometry [tutorial]. *IEEE Robotics Automation Magazine*, 18(4):80–92, 2011.

[60] Hauke Strasdat, J Montiel, and Andrew J Davison. Scale drift-aware large scale monocular slam. *Robotics: Science and Systems VI*, 2(3):7, 2010.

[61] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. A benchmark for the evaluation of RGB-D SLAM systems. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 573–580, October 2012.

[62] Chengzhou Tang and Ping Tan. BA-Net: Dense Bundle Adjustment Network. *arXiv:1806.04807 [cs]*, June 2018. arXiv: 1806.04807.

[63] Jiexiong Tang, Rares Ambrus, Vitor Guizilini, Sudeep Pillai, Hanme Kim, and Adrien Gaidon. Self-Supervised 3D Keypoint Learning for Ego-motion Estimation. *arXiv:1912.03426 [cs]*, December 2019. arXiv: 1912.03426.

[64] Keisuke Tateno, Federico Tombari, Iro Laina, and Nassir Navab. CNN-SLAM: Real-Time Dense Monocular SLAM with Learned Depth Prediction. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6565–6574, Honolulu, HI, July 2017. IEEE.

[65] Philip HS Torr and Andrew Zisserman. Mlesac: A new robust estimator with application to estimating image geometry. *Computer vision and image understanding*, 78(1):138–156, 2000.

[66] Miroslav Trajković and Mark Hedley. Fast corner detection. *Image and Vision Computing*, 16(2):75–87, February 1998.

[67] Bill Triggs, Philip F. McLauchlan, Richard I. Hartley, and Andrew W. Fitzgibbon. Bundle Adjustment — A Modern Synthesis. In Bill Triggs, Andrew Zisserman, and Richard Szeliski, editors, *Vision Algorithms: Theory and Practice*, Lecture Notes in Computer Science, pages 298–372. Springer Berlin Heidelberg, 2000.

[68] S. Umeyama. Least-squares estimation of transformation parameters between two point patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(4):376–380, 1991.

[69] Chaoyang Wang, José Miguel Buenaposada, Rui Zhu, and Simon Lucey. Learning depth from monocular videos using direct methods. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2022–2030, 2018.

[70] Sen Wang, Ronald Clark, Hongkai Wen, and Niki Trigoni. DeepVO: Towards End-to-End Visual Odometry with Deep Recurrent Convolutional Neural Networks. *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2043–2050, May 2017. arXiv: 1709.08429.

[71] Sen Wang, Ronald Clark, Hongkai Wen, and Niki Trigoni. End-to-end, sequence-to-sequence probabilistic visual odometry through deep neural networks. *The International Journal of Robotics Research*, 37(4-5):513–542, 2018.

[72] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.

[73] Changchang Wu. Towards Linear-Time Incremental Structure from Motion. In *2013 International Conference on 3D Vision - 3DV 2013*, pages 127–134, June 2013. ISSN: 1550-6185.

[74] Po-Chen Wu. *Accurate 6 DoF Object Pose Estimation and Tracking*. PhD thesis, Graduate Institute of Electronics Engineering, National Taiwan University, 2018.

[75] Fei Xue, Xin Wang, Shunkai Li, Qiuyuan Wang, Junqiu Wang, and Hongbin Zha. Beyond tracking: Selecting memory and refining poses for deep visual odometry. In *Proceedings*

of the IEEE Conference on Computer Vision and Pattern Recognition, pages 8575–8583, 2019.

[76] Nan Yang, Lukas von Stumberg, Rui Wang, and Daniel Cremers. D3VO: Deep Depth, Deep Pose and Deep Uncertainty for Monocular Visual Odometry. *arXiv:2003.01060 [cs]*, March 2020. arXiv: 2003.01060.

[77] Kwang Moo Yi, Eduard Trulls, Vincent Lepetit, and Pascal Fua. LIFT: Learned Invariant Feature Transform. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *Computer Vision – ECCV 2016*, Lecture Notes in Computer Science, pages 467–483. Springer International Publishing, 2016.

[78] Zhichao Yin and Jianping Shi. GeoNet: Unsupervised Learning of Dense Depth, Optical Flow and Camera Pose. *arXiv:1803.02276 [cs]*, March 2018. arXiv: 1803.02276.

[79] Zhichao Yin and Jianping Shi. Geonet: Unsupervised learning of dense depth, optical flow and camera pose. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[80] Georges Younes, Daniel Asmar, Elie Shammas, and John Zelek. Keyframe-based monocular slam: Design, survey, and future directions. *Robotics and Autonomous Systems*, 98, 09 2017.

[81] Huangying Zhan, Ravi Garg, Chamara Saroj Weerasekera, Kejie Li, Harsh Agarwal, and Ian Reid. Unsupervised Learning of Monocular Depth Estimation and Visual Odometry With Deep Feature Reconstruction. pages 340–349, 2018.

[82] Huangying Zhan, Chamara Saroj Weerasekera, Jiawang Bian, and Ian D. Reid. Visual odometry revisited: What should be learnt? *CoRR*, abs/1909.09803, 2019.

[83] Fuzhen Zhang. Quaternions and matrices of quaternions. *Linear Algebra and its Applications*, 251:21 – 57, 1997.

[84] Huizhong Zhou, Benjamin Ummenhofer, and Thomas Brox. Deeptam: Deep tracking and mapping. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 822–838, 2018.

[85] Tinghui Zhou, Matthew Brown, Noah Snavely, and David G. Lowe. Unsupervised Learning of Depth and Ego-Motion from Video. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6612–6619, Honolulu, HI, July 2017. IEEE.