UNIVERSITY OF CALIFORNIA, SAN DIEGO

Real-Time Monocular Large-scale Multicore Visual Odometry

A dissertation submitted in partial satisfaction of the
requirements for the degree
Doctor of Philosophy

in

Electrical Engineering (Computer Engineering)

by

Shiyu Song

2014

DEDICATION

To my parents, my wife and my daughter.

EPIGRAPH

If you cannot explain it simply,
you don't know it well enough.
—Albert Einstein

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

This dissertation is the product of the constant and kind help of my supervisor, mentors, colleagues, friends and family.

First, I thank my supervisor Prof. Clark Guest for his unwavering encouragement, guidance and support. He led me into this exciting field of computer vision. His open research environment has always been great opportunities for me. My first vision project under his advising during the first year of my PhD was a valuable experience. I was impressed by the fantastic performance of the vision systems. It was a start and an eye-opener.

Much of the work of this PhD was done during my internship with my mentor, Dr. Manmohan Chandraker, at NEC Laboratories America. The breadth of his knowledge, meticulous work attitude and stringent requirements for my intern work are the most important elements making my work on monocular SFM successful. From start to finish, he never fails to give me valuable guidance, support and help.

My time at San Diego Supercomputer Center with Dr. Yifeng and Dr. Jun Zhou was a wonderful experience. It was a great opportunity for me get familiar with the state-of-the-art distributed computing technologies, which will be certainly extremely helpful in my future career.

I would like to thank my wife and my 1 year old daughter for their patience and continual support during my time working on this dissertation. I owe them a particularly large debt of the family time. It would be all too easy to take it for granted.

Finally, I would like to thank my parents. Their support for me to study abroad is tremendous and wholehearted. Their pain at being oceans apart from me is only surpassed by every little success I achieve now and in the future.

Parts of this dissertation are based on papers co-authored with my collaborators:

- Chapter 4 is based on “Parallel, real-time monocular visual odometry”, by Shiyu Song, Manmohan Chandraker, Clark C. Guest as it appears in proceedings of Robotics and Automation (ICRA), 2013 IEEE International Conference on, May 6-10 2013, Karlsruhe.

- Chapter 5 is based on “Robust Scale Estimation in Real-Time Monocular SFM

for Autonomous Driving”, by Shiyu Song, Manmohan Chandraker, as it appears in proceedings of Computer Vision and Pattern Recognition (CVPR), 2014 IEEE International Conference on, June 24-27 2014, Columbus, Ohio.

- Chapter 4, 5 and 6 in part, have been submitted for publication, as it may appear in “High Accuracy Monocular SFM and Scale Correction for Autonomous Driving”, by Shiyu Song, Manmohan Chandraker, Clark C. Guest, in IEEE Transactions on Pattern Analysis and Machine Intelligence.
- Chapter 7 has been submitted for publication, as it may appear in “High Accuracy Monocular 3D Object Localization for Autonomous Driving”, by Shiyu Song, Manmohan Chandraker, in proceedings of European Conference on Computer Vision, Zurich, September 6-12, 2014.

VITA

2004-2008	B. S. in Electrical Engineering, Tsinghua University, China
2008-2010	M. S. in Electrical and Computer Engineering, University of California, San Diego
2010-2014	Ph. D. in Electrical and Computer Engineering, University of California, San Diego

PUBLICATIONS

Shiyu Song, Mohammed Billoo, Kamran Mahbobi and Clark Guest, "Multi-Sensor Comparison and Data Fusion for Mapping Enclosed Spaces", *Journal of Electronic Imaging*, 21(2), 021104, May 10, 2012.

Shiyu Song, Manmohan Chandraker and Clark Guest, "Parallel, real-time monocular visual odometry", *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pp 4698 - 4705, May 6-10 2013, Karlsruhe

Shiyu Song, Manmohan Chandraker, "Robust Scale Estimation in Real-Time Monocular SFM for Autonomous Driving", *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE International Conference on*, June 24-27 2014, Columbus, Ohio

ABSTRACT OF THE DISSERTATION

Real-Time Monocular Large-scale Multicore Visual Odometry

by

Shiyu Song

Doctor of Philosophy in Electrical Engineering (Computer Engineering)

University of California, San Diego, 2014

Professor Clark C. Guest, Chair

We present a real-time, accurate, large-scale monocular visual odometry system for real-world autonomous outdoor driving applications. This dissertation makes four important contributions: First, we demonstrate robust monocular Structure from Motion (SFM) with a series of architectural innovations including a novel epipolar searching module in a parallel thread to replenish 3D points through a series of validation mechanism, a novel combination of local and global bundle adjustment that ensures accuracy, robustness and efficiency and so on (Chapter 4). These architectural innovations address the challenge of robust multithreading even for scenes with large motions and rapidly changing imagery. Our design is extensible for three or more parallel CPU threads. The novel epipolar searching module, operating in parallel with other threads to generate new 3D points at every frame, together with the local bundle adjustment in the primary thread, significantly boost robustness and accuracy. Secondly, we demonstrate robust monocular SFM with accuracy unmatched by prior state-of-the-art; over several kilometers, we achieve performance similar to stereo that far exceeds other monocular architectures. The key to this performance is scale drift correction using ground plane estimation that combines cues from sparse features and dense stereo (Chapter 5). Our third contribution is a data-driven mechanism for cue combination that learns models from training data to relate observation covariances for each cue to error behavior of underlying variables. During testing, this allows per-frame adaptation of observation covariances based on relative confidences inferred from visual data (Chapter 5). Finally, we present a framework for highly accurate 3D localization of objects like cars (Chapter 7) and a lane detection system (Chapter 8) based on our SFM poses and ground planes. Our accurate ground plane and SFM poses easily benefit 3D localization frameworks and vision-based lane detection for those applications, as demonstrated by our experiments. Our baseline SFM system is optimized to output pose within 50 ms in the worst case, while average case operation is over 30 fps. Evaluations are presented on the challenging KITTI dataset for autonomous driving, where we achieve better rotation and translation accuracy than other state-of-the-art systems.

Chapter 1

Introduction

The objective of this dissertation is to show how a robot can use images, which convey only 2D information, in a robust manner to locate itself and surroundings in 3D space. Effective use of video sensors for obstacle detection and navigation has been a goal in ground vehicle robotics for many years. In this work, we present a real-time, monocular vision-based system that relies on several innovations in multithreaded visual odometry for autonomous driving. It achieves outstanding accuracy in sequences spanning several kilometers of real-world environments.

Visual odometry (VO) is the process of determining the position and orientation of a robot or vehicle by analyzing images from associated video cameras. Application domains include robotics, wearable computing, augmented reality, and automotive navigation. The term VO was first used in 2004 by Nister in his ground-breaking paper [NNB04]. The problem of recovering camera poses and 3D scene structure from a set of ordered or unordered images is also known as structure from motion (SFM) in the computer vision community. Visual odometry is a special case of SFM, and SFM is a more general term than visual odometry. Besides estimating the camera poses, VO focuses on estimating camera poses sequentially and in real time. The three-dimensional structure of the scene is typically generated by visual odometry, too, as by-product. The structure of the scene is useful for applications such as obstacle detection, object localization and 3D reconstruction. The visual odometry system can be a base for these higher-level applications. Figure 1.1 demonstrates the concept of the visual odometry and its applications.

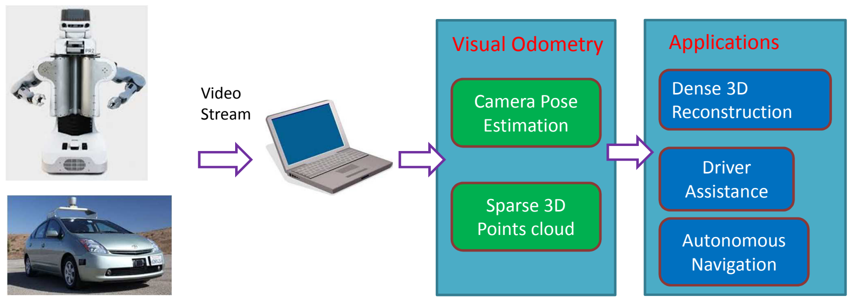


Figure 1.1: The concept of the visual odometry and its applications. Visual odometry is the process of determining the position and orientation of a robot or vehicle by analyzing images from associated video cameras.

Image streams are acquired from video cameras. A visual odometry system processes input image streams and produces a vehicle or robot localization map as shown in Figure 1.2. How to extract necessary information from the 2D images, how to compare and match them among neighboring frames and how to state-of-the-art perspective geometry technologies to compute the 3D motion of the camera are the fundamental problems that have been long addressed by the robotics and computer vision communities.

1.1 Motivation and Objectives

Visual odometry is a problem that has gained immense traction in recent years. It is a key component in robot navigation and real-world autonomous outdoor driving. While a system based on a laser scanner has been able to generate a detailed 3D map of its environment and drive itself by combining it with high-resolution maps of the world [TMD$^+$06], vision based systems are attractive for the automobile industry. The reasons behind this are economical and technical. Google's robotic cars have about $150,000 in equipment including a $70,000 LIDAR (laser radar) system, the Velodyne 64-beam laser. It has four radars to avoid obstacles. A position estimation sensor is mounted on the left rear wheel to determine the car's position on the map. Besides these, it also uses video cameras as the vision-based system does as shown in Fig 1.3. Clearly, such high equipment prices would limit purchases to only the most luxurious vehicles.

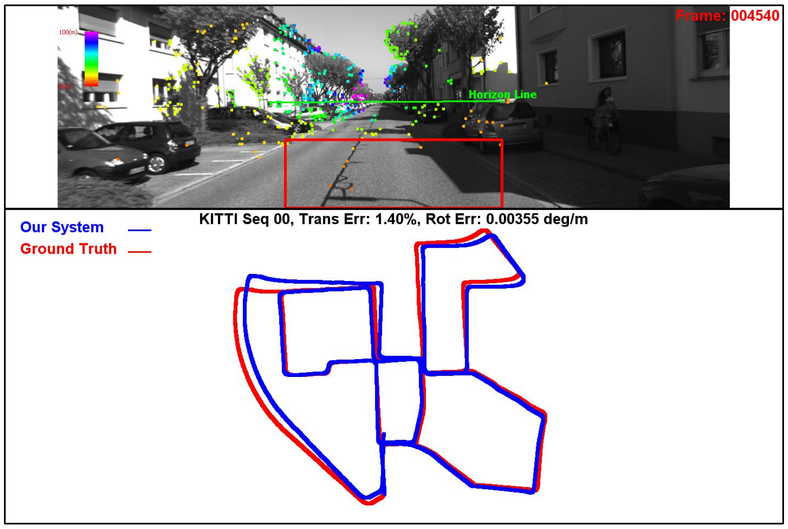


Figure 1.2: Input and output of our visual odometry system. (Top) Input images from video cameras. The colored points are estimated 3D points from our visual odometry system. (Bottom) Vehicle motion estimated from our visual odometry system comparing with ground truth.

On the other hand, costs of cameras have steadily declined in recent years, even for the industrial cameras that are produced in lesser volume, support high frame rates and are robust to extreme temperatures, weather and jitters. Technically, cameras are the most suitable sensors for essential functions such as pedestrian, vehicle and lane detection in an autonomous driving systems. The extremely rich information (shape, texture, color) provided by vision sensors is indispensable in an autonomous driving system. Actually, people also rely on vision sensors (human eyes) to drive a car. Thus, vision sensors such as cameras are the first choice. An autonomous driving system that only relies on cameras is the final goal of the automobile industry and visual odometry is an essential component in such a system.

Traditionally, odometry is used to estimate change in position over time. The traditional method of rotary encoders is to estimate the motion of the vehicle or robot by measuring wheel rotations. While it is useful for many wheeled or tracked vehicles, it cannot be applied to mobile robots with non-standard locomotion methods, such as legged robots. Additionally, traditional odometry suffers from precision problems, since wheels tend to slip and slide creating a non-uniform distance traveled as compared to

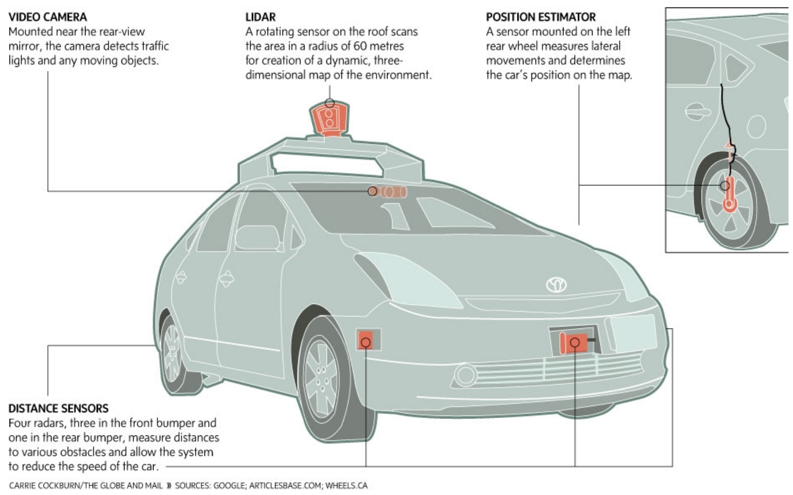

VIDEO CAMERA
Mounted near the rear-view mirror, the camera detects traffic lights and any moving objects.
LIDAR
A rotating sensor on the roof scans the area in a radius of 60 metres for creation of a dynamic, three-dimensional map of the environment.
POSITION ESTIMATOR
A sensor mounted on the left rear wheel measures lateral movements and determines the car's position on the map.
DISTANCE SENSORS
Four radars, three in the front bumper and one in the rear bumper, measure distances to various obstacles and allow the system to reduce the speed of the car.
CARRIE COCKBURN/THE GLOBE AND MAIL » SOURCES: GOOGLE; ARTICLESBASE.COM; WHEELS.CA

Figure 1.3: Google's robotic cars have high-cost equipment with it. Google's robotic cars have about $150,000 in equipment including a $70,000 LIDAR (laser radar) system, a Velodyne 64-beam laser. It has four radars to avoid obstacles. A position estimation sensor is mounted on the left rear wheel to determine the car's position on the map. Besides these, it also equips video cameras. By contrast, our monocular visual odometry system only requires video images from a single camera as input.

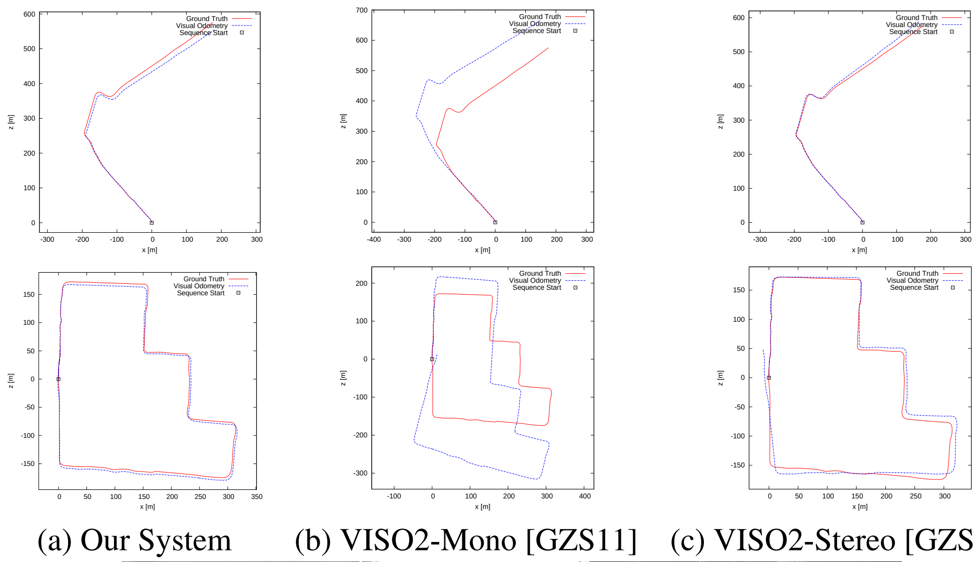
the wheel rotations. The error is compounded when the vehicle operates on non-smooth surfaces. Odometry readings become increasingly unreliable over time as these errors accumulate. Visual odometry allows for enhanced navigational accuracy in robots or vehicles using any type of locomotion on any surface. In ground vehicle applications, visual odometry can provide an alternative or compliment with respect to wheel odometry since it is not affected by wheel slip in uneven terrain or other adverse conditions. It has been demonstrated that visual odometry provides more accurate estimates than wheel odometry [SF11]. This capability promotes visual odometry as a good supplement to systems such as the global positioning system (GPS), inertial measurement units (IMU), and laser odometry. In indoor environment where GPS is not available, visual odometry has even greater application and advantage.

Traditional visual odometry adopts stereo camera solutions. While stereo SFM systems routinely achieve high accuracy and real-time performance, the challenge remains daunting for monocular ones. Yet, monocular systems are attractive for the automobile industry since they are cheaper and the calibration effort is lower. The accuracy of the

Figure 1.4: Stereo vision and monocular vision from nature. (Top row) Lions' and cats' eyes face forward and create binocular vision (stereo vision) with a large overlap area in the field of vision for both eyes. (Bottom row) Rabbits and pigeons' eyes face sideways and create monocular vision with a small overlap between the field of vision for both eyes.

inter-camera stereo calibration has direct impact on the accuracy of the visual odometry output. Accuracy is dependent on stereo calibration which can be hard to ensure if the cameras are separated significantly. Much more effort is needed to maintain a calibrated constant baseline between the pair of the cameras than maintaining a single calibrated camera. Stereo calibration is not needed for monocular visual odometry systems. Besides the application for autonomous driving, visual odometry is also an important technology for robot navigation. Monocular visual odometry systems are inherently good for small object robotics by saving the space of the baseline between the pair of cameras in stereo vision systems. Finally, interfacing and synchronization are more difficult for stereo cameras compared to a monocular camera. Nature also adopts stereo or monocular vision system in different situations as shown in Figure 1.4.

In this work, we present a real-time, monocular vision-based system that relies on several innovations in multithreaded visual odometry for autonomous driving. It achieves outstanding accuracy in sequences spanning several kilometers of real-world environments. We present results on the public challenging KITTI dataset, which is


(a) Our System (b) VISO2-Mono [GZS11] (c) VISO2-Stereo [GZS11]

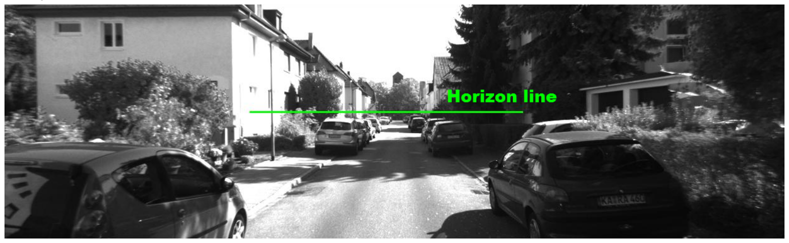
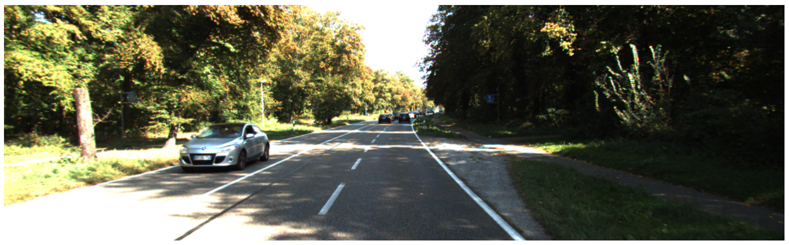


Figure 1.5: Performance of our monocular visual odometry system. (Top row) (a) Our monocular visual odometry yields camera trajectories close to the ground truth over several kilometers of real-world driving. (b) Our monocular system significantly outperforms prior works that also use the ground plane for scale correction. (c) Our performance is comparable to stereo-based visual SFM. (Bottom row) Scale drift correction using a novel, adaptive ground plane estimation allows such accuracy and robustness (Chapter 5). The green line is the horizon from the estimated ground plane.

collected specifically for autonomous driving applications [GLU12], and other datasets. In KITTI dataset, imagery is collected at only $10Hz$ and vehicle speeds can be high, which results in large inter-frame motion of corresponding features. In extensive experiments on the challenging KITTI dataset, we achieve a rotation accuracy of 0.0057 degrees per frame, outperforming several state-of-the-art stereo systems. Our translation error is 2.53%, which is also comparable to stereo and unmatched by other state-of-the-art monocular systems. Sample results are shown in Fig 1.5.


(a) Frame 000037 in Sequence 0018

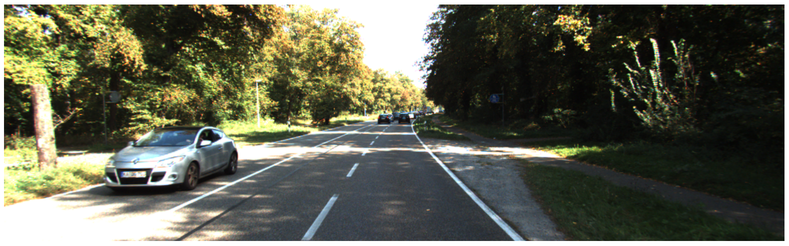

(b) Frame 000038 in Sequence 0018

Figure 1.6: Sample frames in KITTI odometry dataset. Two neighboring frames Frame 000037 and 000038 in Sequence 0018 of KITTI Tracking Dataset. The vehicle speed is high and large motions may occur between consecutive frames. This places severe demands on an autonomous driving visual odometry system, necessitating extensive validation and refinement mechanisms that conventional systems do not require.

1.2 Challenges

The challenges of monocular visual odometry for autonomous driving are both fundamental and practical. For instance, it has been observed empirically and theoretically that forward motion is a "high error" situation for visual odometry [Oli05]. By contrast, stereo visual odometry systems typically have a wide baseline to overcome this issue.

Vehicle speeds in outdoor environments can be high, so even with high frame rate cameras, large motions may occur between consecutive frames, as shown in Figure 1.6. This places severe demands on an autonomous driving visual odometry system, necessitating extensive validation and refinement mechanisms that conventional systems do not require. Our system makes judicious use of a novel multithreaded design to ensure that location estimates (and the underlying map variables) become available only after extensive validation with long-range constraints and thorough optimization, such as bundle adjustments [TMHF00], but without undue delay.

The timing requirements for visual odometry in autonomous driving are equally stringent – a pose must be output at every frame in a fixed amount of time. Thus, our system is optimized for worst-case timing scenarios, rather than the average-case optimization for most traditional systems. Traditional systems may produce a spike in delays when special frames are processed, or some operations are performed. For instance, [CLFP10] has a delay when system detects a loop in the scene when the loop closure is performed. In particular, our multithreaded system produces pose outputs in at most 50 ms, regardless of the type of the frame added or the operation performed. The average frame rate of our system is much higher, at above 30 fps (33 ms delay only).

An important aspect of our system design is its judicious use of multithreading. Visual odometry is an inherently sequential operation. This is especially true for outdoor autonomous driving, as opposed to indoor or desktop applications where the possibility of repeatedly viewing the same scene structures is high. For our application, with rapidly changing features in the visible field of view, optimization such as bundle adjustment must be per-frame, while highly accurate new 3D points must be added to the system with no luxury of off-cycles or revisited regions of the map to perform delayed refinements.

Thus, designing a multithreaded system requires achieving a delicate balance between accuracy and latency. A key emphasis of this work is to illustrate that besides the obvious speed advantages, well-designed multithreading can also greatly contribute to the accuracy and robustness of the system. The high redundancy epipolar search mechanism (Sec. 4.1 in Chapter 4), novel keyframe architectures that allow updating trackable 3D points with reliable long tracks (Sec. 4.2 in Chapter 4) and thread safe modules that allow online bundle adjustment are some of the innovations that allow our system to meet the competing demands of speed, accuracy and robustness. As an example, consider our epipolar constrained search. A single-thread version of a system that relies on 2D-3D correspondences might update its stable point set by performing an epipolar search in the frame preceding a keyframe. However, the support for the 3D points introduced by this mechanism is limited to just the triplet used for circular matching and triangulation. By moving the epipolar search to a separate thread and performing circular matching at every frame, we may supply 3D points with tracks of length up to the distance from the preceding keyframe. The set of long tracks provided by the epipolar thread in our multithread system is far more likely to be free of outliers. Our multithread architecture allows elegant extension to as many threads as desired. The strength of our results demonstrates the necessity of such a design to robustly and efficiently meet the challenge of monocular autonomous driving.

Monocular vision-based frameworks are attractive due to lower cost and calibration requirements. However, the lack of a fixed stereo baseline leads to inevitable scale drift, which is a primary bottleneck that has prevented monocular visual SFM from attaining accuracy comparable to stereo. To counter scale drift, prior knowledge is used, which is commonly the height of the camera above the ground plane. Thus, a robust and accurate estimation of the ground plane is crucial to achieve good performance in monocular scene understanding. However, in real-world autonomous driving, the ground corresponds to a rapidly moving, low-textured road surface, which makes its estimation from image data challenging.

We overcome this challenge with two innovations in Chapter 5: first, we incorporate cues from multiple methods of ground plane estimation and second, we combine them in a principled framework that accounts for their per-frame relative confidences, using models learned from extensive training data. While prior works have used sparse feature matching for ground plane estimation [GZS11, SS08, SCG13], it is demonstrably inadequate in practice and must be augmented by other cues such as the plane-guided dense stereo of Sec. 5.2 in Chapter 5. To combine cues, Sec. 5.3 in Chapter 5 proposes a Kalman filter framework that adapts the fusion observation covariances at every frame to reflect the relative uncertainty of each cue. This is achieved by a training procedure on over 20000 frames from the KITTI dataset, whereby models are learned that relate the observation covariance for each cue to the error behaviors of its underlying variables. To the best of our knowledge, such adaptive estimation of observation covariances for cue combination is novel.

1.3 VO vs V-SLAM

Visual odometry (VO) and Visual Simultaneous localization and mapping (V-SLAM) are two systems that have many similar aspects but focus on different applications. We discuss the relationship of VO and V-SLAM in this section.

The goal of a VO system is to recover the camera trajectory incrementally. Conversely, a SLAM system seeks to obtain a global, consistent map of the environment. A trajectory of the camera is inevitably estimated during this procedure. A common technology used in SLAM systems is loop closure. When a loop closure is detected, this information is used to refine both the map and the camera trajectory. A V-SLAM system is usually more complex and computationally expensive, because loop closure has to be detected and a global optimization (called global bundle adjustment) has to be performed to obtain a consistent map. On the other hand, VO systems care about local consistency of the trajectory, and a local optimization (called windowed bundle adjustment or local bundle adjustment) is used to obtain a more accurate local trajectory.

Most V-SLAM systems are designed to work in small, indoor workspaces. However, VO systems can usually work in large-scale environments, but scale drift is the most significant obstacle for monocular visual odometry systems to achieve good accuracy as discussed in Section 1.2. Our system is a VO system. We focus on local trajectory accuracy more than global. We don't detect things like loop closure and don't perform a huge global bundle adjustment, either. However, this doesn't imply that our system's accuracy is worse than state-of-the-art V-SLAM systems in our application, since loop closure rarely happens in practical autonomous driving. For an image sequence without loop, loop closure and global bundle adjustment lose their foundation to improve the accuracy globally. In Chapter 6, we show that our system performs better than one of the state-of-the-art V-SLAM systems, EKFMonoSLAM [CGDM09].

1.4 Contributions of the Dissertation

Let us summarize our discussions so far. The main contributions of this dissertation are:

1. **Real-time Parallel Monocular Visual Odometry**: Our first set of contributions is a real-time parallel monocular visual odometry system that consists of several technical and methodological innovations to address the problem of monocular visual odometry:

   (a) Highly accurate, robust, scale-corrected and real-time monocular SFM with performance comparable to stereo, including several architectural innovations to address the challenge of robust multithreading even for scenes with large motions and rapidly changing imagery.

   (b) A novel data-driven framework that combines multiple cues for ground plane estimation using learned models to adaptively weight per-frame observation covariances.

   (c) Scale drift correction by adaptively combining multiple cues for ground plane estimation using learned models to correctly weight per-frame observation covariances.

2. **A monocular 3D object localization framework**: Our next set of contributions is a 3D object localization framework based on the monocular visual odometry system. We demonstrate that a significant improvement in ground plane estimation leads to excellent performance in applications such as monocular visual odometry and 3D object localization.

   (a) We introduce novel cue from object detection to improve ground plane estimation for 3D object localization, including novel use of detection cues for ground estimation, which boosts 3D object localization accuracy.

   (b) A joint optimization framework for 3D object localization that combines SFM cues such as 3D points and ground plane, with object cues like bounding boxes and detection scores, to achieve accuracy in both near and far fields.

   (c) Incorporation of raw detection scores to allow 3D bounding boxes to “undo" tracking errors, that is, achieve consistency with both 3D geometry as well as detection scores.

1.5 Organization of the Dissertation

The structure of the rest of this thesis is as follows. In Chapter 2 we introduce the state-of-the-art prior works, including brief comparison with them. In Chapter 3, we introduce the notation, geometrical models and other preliminaries that we will build the theory of the rest of the thesis on. In Chapters 4, we concentrate on the system architecture of our real-time monocular SFM system, and describe each component of the system in details. In Chapter 5, we present our novel ground plane estimation. The ground plane estimation is an important contribution of this thesis. Not only it helps the SFM system to correct the scale drift, but also it benefits the monocular object localization and the lane detection. In Chapter 6, we present the performance of our SFM system, and demonstrate that it is comparable to stereo systems on real-world driving sequences. In Chapter 7, we present a 3D localization framework based on our ground plane estimation and other cues. The benefits of having these cues are also demonstrated by experiments. In Chapter 8, we present a lane detection system that incorporates both the SFM poses and the ground plane estimation results, in contrast to traditional pure image-based works. Finally, we conclude in Chapter 9 with summaries and some thoughts on potential future work.

Chapter 2

State of the Art

The original contribution addressing the problem of the structure from motion (SFM) can be dated to 1988 by Harris [HP88] and 1987 by Longuet-Higgins [LH87]. The early work of estimating a vehicle's ego-motion with a vision based system starts in 1980s. [Mor80] [MS87] [OMSM00] were done for the NASA Mars exploration program to provide all-terrain rovers that can measure their motion in uneven and rough terrains. In these works, they use stereo vision or a single sliding camera (It can be considered as stereo because the robot moved in a stop-and-go fashion and the baseline length is known.). The alternative to stereo vision is to use a single camera in a monocular vision system. The challenge of the monocular visual odometry system is that motion can only be recovered up to a scale factor. The absolute scale then has to be determined from other sources.

Over the years, monocular and stereo visual odometry systems have progressed as two almost independent lines of research, so we separately introduce their development in the following sections. Some sections in this chapter are based on the survey works [SF11, FS12, DWB06, BDW06].

2.1 Monocular Visual Odometry

The challenge of the monocular visual odometry is that one can not extract depth information from a single frame. Since the absolute scale is unknown, more complicated mechanisms are required to maintain the computed 3D structure. Therefore when a new image arrives, the relative scale and the camera motion between the previous image and the new image has to be accurately estimated. Unfortunately, during this procedure, scale drift happens inevitably.

In this section, we give a brief review of the existing monocular visual odometry systems. The limitation and the drawbacks of these prior systems are discussed. In constrast to prior real-time SLAM systems, our system architecture is intricately designed to meet the challenge of accurate and efficient monocular autonomous driving. In Chapter 4, we will discuss how our design is fundamentally different, better suited to the application, easily extensible, and more accurate.

2.1.1 Monocular Visual Odometry by Nister et al.

The first real-time, large-scale visual odometry system with a single camera was developed by Nister et al in his breakthrough work [NNB04]. This work is known not only for the achievement of the first real-time visual odometry system, but also for a series of mechanism innovations.

First, it's a feature-based method, but instead of tracking features among frames such as Kanade–Lucas–Tomasi (KLT) feature tracker [LK81], it detects features (Harris corners [HS88] [ST94a]) independently in frames and only allows matches between them. This improves tracking quality by avoiding tracking drift, which is a common issue in the traditional approach.

Second, during initialization, a five-point algorithm [Nis04] together with RANdom SAmple Consensus (RANSAC) [FB81a] is used to compute the initial relative motion between two frames. In contrast to the five-point algorithm, traditionally an eight-point algorithm [LH87] [Har97] has been used to solve the same problem. The five-point algorithm has significant advantages over the traditional eight-point algorithm, for example in sideways motions. After this work, the five-point algorithm became a standard approach for the initialization stage in a monocular VO system [NNB06] [TPD08] [MLD$^+$06] [SFS09] [PMP11]. After initialization, the system triangulates the observed feature tracks into 3D points.

Third, they used a pose estimation algorithm based on 3D-to-2D matches. This problem is known as a Perspective-n-Point (PnP) problem in the field. In particular, they used a 3-point algorithm in [HLON91] together with RANSAC to solve it.

Finally, as mentioned before, they incorporated RANSAC [FB81a] outlier rejection scheme into the framework, so that the quality of the 3D points and 2D matches can be well maintained. It provided an opportunity for future work, such as PTAM [DRMS07, KM07, KM08] and also our approach to incorporate the important key-frame mechanism, which enables use of local and global bundle adjustment, since bundle adjustment is known to be sensitive to outliers [TMHF00].

Our visual odometry system is developed based on all these novel designs from Nister's work. Standing on the shoulders of giants, we have introduced many mechanism innovations, such as using a more powerful feature extraction (ORB [RRKB11a]) during initialization, replacing the 3-point algorithm with the more efficient and robust EPnP algorithm [LMNF09] to solve the PnP problem, a complete new multithreading framework including a novel epipolar searching thread to replenish high quality 3D points and 2D matches, and other innovations. We will discuss these in Chapter 4. Additionally, neither the scale drift nor the scale initialization issue is discussed in Nister's work.

2.1.2 Libviso Mono by Geiger et al.

Libviso Mono is a monocular visual odometry system proposed by Geiger et al [GZS11] which has a completely different framework than Nister's.

It relies on matching and computing relative pose between every consecutive pair of frames through a fundamental matrix estimation using an eight-point algorithm [LH87] [Har97]. Fundamental matrix estimation algorithms, such as eight-point or five-point algorithms, have the limitation that the absolute scale can't be recovered, as we discussed in the section 1.2 of Chapter 1. Similar to our approach, Libviso Mono estimates a locally planar ground to solve the absolute scale issue.

This simple approach has the advantage of robustness. Because it is based on two-view estimation, ideally the system never breaks down as long as there are enough good features that can be extracted in the scene. But there are also several drawbacks of such an approach. First, it is known that two-view estimation leads to high translational errors in the case of narrow baseline forward motion [Oli05]. This leads to high error when the vehicle's speed is low, as shown in Figure 2.1. Second, there is higher drift

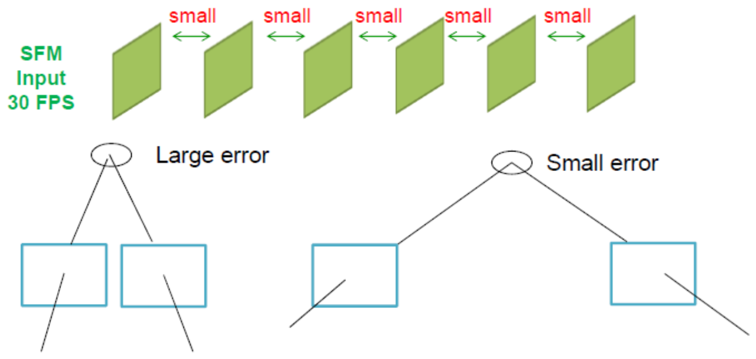


Figure 2.1: The drawback of a small baseline in fundamental matrix estimation. A small baseline between two frames in fundamental matrix estimation leads to a large error. In contrast, a large baseline can improve the accuracy. It's the key difference between our approach (also Nister's) and Libviso Mono. Our approach is carefully designed to introduce long-range constraints while maintaining efficiency and relies on 3D-2D pose estimation rather than narrow baseline fundamental matrix estimation.

since distant constraints from long tracks are not used. Using local tracks can improve the local consistency of the computed trajectory. In conclusion, Libviso Mono is a baseline monocular visual odometry method and its accuracy is low due to its architecture defects.

In contrast, our approach is carefully designed to introduce long-range constraints while maintaining efficiency and relies on 3D-2D pose estimation rather than narrow baseline fundamental matrix estimation. We also introduce a novel cue combination framework for ground plane estimation that significantly boosts the accuracy of our system.

2.1.3 EKF MonoSLAM by Davison et al.

As we introduced in Section 1.3 of Chapter 1, in contrast to a VO system, another category is the visual simultaneous localization and mapping (V-SLAM) system, which focuses more on map building. The camera trajectory is estimated during this procedure. The first real-time monocular V-SLAM system is MonoSLAM developed by Andrew Davison in his breakthrough work [Dav03]. This work built on earlier but more limited filtering-based approaches such as the work of Chiuso et al. [CFJS00]. It successfully adapted the extended Kalman filter (EKF) approach with full-covariance to the monocular

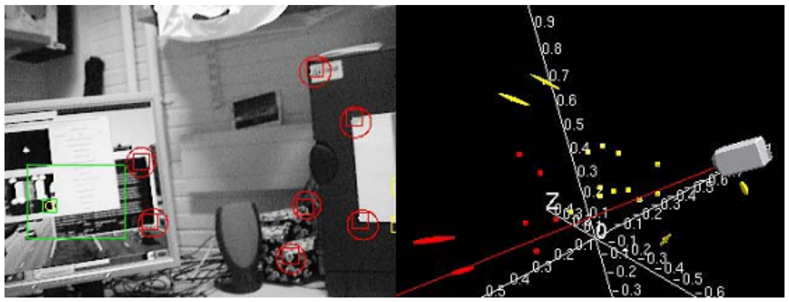


Figure 2.2: Typical results output by EKF MonoSLAM. (Left) Detected feature patches are in red rectangles. Feature searching regions are in red circles. (Right) The map built by the system. Feature location uncertainties are represented with various sizes of circles. Red features are currently being tracked. Yellow features are currently not selected for measurement.

SLAM field. The advantage of Davison's work was to observe repeatable localizations in a fixed amount of time. The camera pose and localizations of a sparse set of feature points are estimated by maintaining full covariance over the complete state vector. The covariance matrix is updated during this procedure so that the map is refined. Typical results output by EKF MonoSLAM are shown in Figure 2.2.

Later, Handa et al. [HCSD10] improved Davison's by an active matching technique based on a probabilistic framework. Civera et al. [CGDM09, CGDM10] proposed to integrate a 1-point RANSAC within the Kalman filter that uses the available prior probabilistic information from the EKF in the RANSAC model hypothesize stage. This allows the minimal sample size to be reduced to one, resulting in large computational savings without the loss of discriminative power.

The limitation of the EKF MonoSLAM is discussed in the next section together with the approach of PTAM. In Chapter 6, we will show some comparison results between our system and Civera's EKFMonoSLAM.

2.1.4 PTAM by Klein et al.

Recently, a few purely vision-based monocular systems have achieved good localization accuracy, but mainly for smaller indoor environments [DRMS07, KM07, KM08]. PTAM is an elegant two-thread architecture separating the tracking and mapping aspects of monocular visual SLAM that has been proposed by Klein and Murray [KM07]. It proposes to include feature measurements only from wide-baseline key-frames that are selected heuristically based on the number of frames observed and the change that has happened in the scene. The key-frames provide the 2D correspondences of the 3D points to reside on. It enables the use of global bundle adjustment to refine the map and also the poses. PTAM also introduced a multi-threading architecture including two threads. One thread runs local and global bundle adjustment and refines the map continuously, as shown in Figure 2.3. Another thread computes the camera motion accurately by projecting hundred of features in the map to the current image and matching them, similar to Nister's work. A PnP solver is also used in this step. Finally, PTAM spends more time on robust tracking, locating to sub-pixel accuracy hundreds of features in different image pyramids. The sheer number of features significantly improves the robustness and mapping accuracy.

Lately, Strasdat et al. [SMD10a] presented a new framework for large-scale V-SLAM. It takes advantage of the key-frame architecture from PTAM and also introduces loop closure to address the scale drift issue. We'll discuss loop closure in Section 2.2.1. Weiss et al. [AAWS11, WSS11] ported the original PTAM to Robot Operating System (ROS) platform [QCG+09], and introduced some mechanism innovations to reduce the VSLAM framework to a visual odometry framework. For example, the extension of PTAM by Weiss has a maximum number of key-frames retained in the map to make it more applicable as a large-scale visual odometry system. Weiss also improves the feature handling in PTAM to robustly handle self-similarity in the environment, e.g. the asphalt in urban areas or the grass in rural areas.

We note that the intuitions behind Klein's (or Nister's) and Davison's approaches are completely different. Both of them have proven successful, but they solve the problem in different ways. Filtering methods, such as Davison's, marginalize out past poses and summarize the information gained over time with a probability distribution. Key-frame methods retain the optimization approach of global bundle adjustment, but computationally must select only a small number of past frames to process. A intensive comparison and analysis between these two different approaches are conducted by Strasdat et al. [SMD10b, SMD12]. In terms of map building accuracy, they concluded

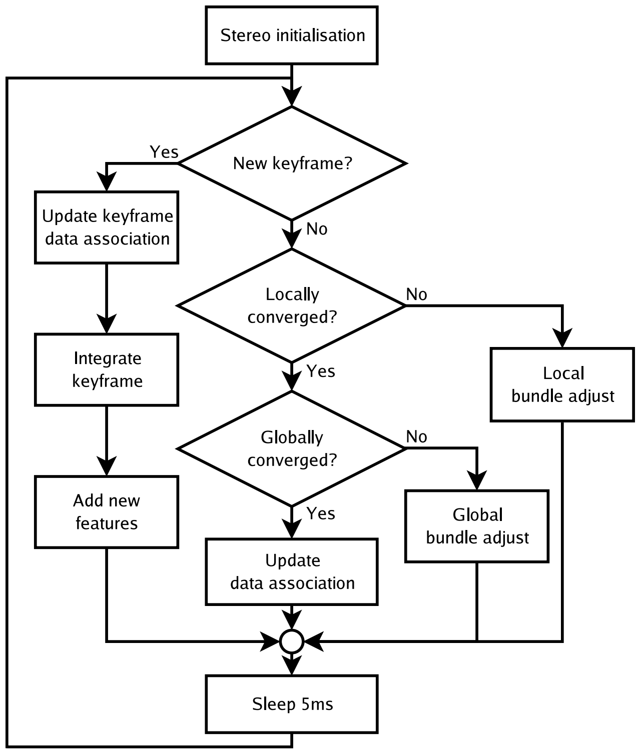


Figure 2.3: The workflow of the mapping thread of PTAM. After initialization, the thread runs local and global bundle adjustment. When a key-frame is detected, the thread adds new features into the map. In contrast, our system proposes a more elegant epipolar searching thread to replace the new feature adding step in the map thread. Our system uses this novel multi-threaded design to ensure that feature matches (and the underlying map measurements) become available only after extensive validation with long-range constraints, but without delay.

that it is computationally more efficient to increase the number of tracked features than the number of frames. This conclusion confirms that the bundle adjustment approach (Klein's) is more efficient than the filtering one, therefore bundle adjustment is adopted in our design.

However, PTAM is a SLAM system, so it focuses on building an accurate map for small workspace environments and relies extensively on repeatedly observing a small set of 3D points. Even for the extension of the PTAM from Weiss that targets the application of an autonomous helicopter in the outdoor environment, the workspace is also a small environment essentially, because the helicopter can repeatedly observe 3D points in the world in a top-down view. These systems do not scale well to large outdoor environments, such as driving situations where scene points rapidly disappear from the field of view. The latter is an important restriction that motivates our improved architecture for the application of large-scale visual odometry.

2.2 Scale Drift Correction

Scale drift is a crucial challenge that prevents monocular autonomous driving from emulating the performance of stereo. Unlike stereo, the lack of a fixed baseline leads to scale drift, which is the main bottleneck that has prevented monocular SFM from attaining accuracy comparable to stereo. Robust and accurate monocular SFM that effectively counters scale drift in real-world road environments has significant potential benefits for mass-produced autonomous driving systems. In this section, we discuss the state-of-the-art scale drift correction technology in monocular visual odometry systems.

2.2.1 Loop Closure

The most common approach to refine the map and counter scale drift is loop closure. Events such as observing a landmark again after not seeing it for a long time or coming back to a previously mapped area are called loop closure [BDW06]. Strasdat et al. [SMD10a] propose a large-scale monocular system that handles scale drift with loop closure. Loop closure is not only valuable for monocular VO or SLAM systems, but also important for stereo systems, since the incremental error during pose estimation in

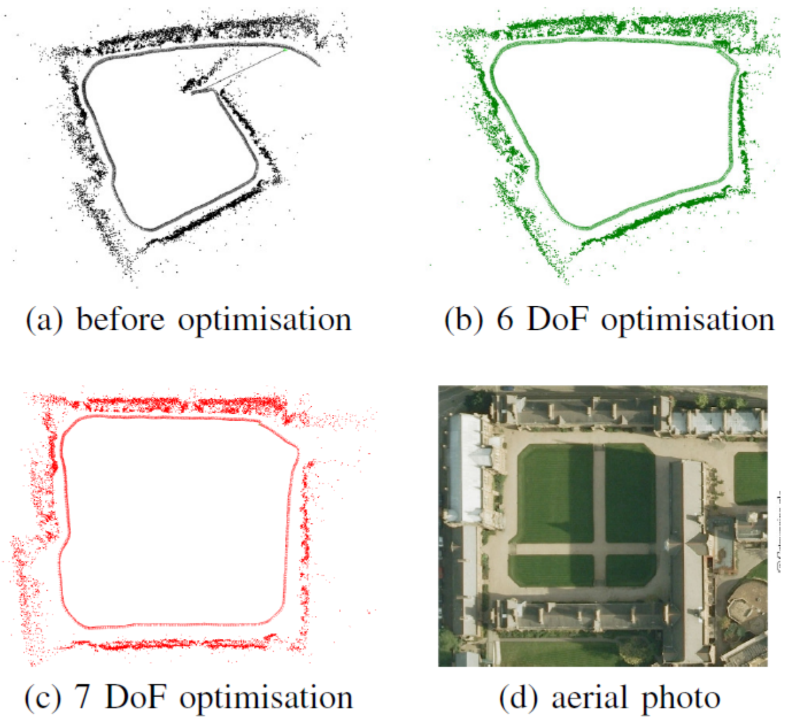
SLAM systems can't be neglected.

Usually, loop closure can be detected by evaluating visual similarity between current and past camera images. Ulrich [UN00] and Jogan [JL00] use global image descriptors to evaluate visual similarity. However, it's more popular to detect loops using local image descriptors. One of the most successful methods is based on visual words [NCH06] [CN08] [FEN07] [FWmFP08] [AFDM08]. The advantage of the visual word-based approach is its efficiency. It is extremely fast to evaluate visual similarities between large sets of images using a inverted index [AFDM08] or a visual vocabulary tree [NS06]. After loop detection, a pose graph can be constructed. In the pose graph, the camera poses are the nodes and the rigid-body transformations between camera poses are the edges between nodes. The rigid-body transformation estimated from the detected loop can be added to the pose graph as an additional loop constraint, so that the map can be updated [OLT06] [GKS+10]. This can efficiently correct translational and rotational drift, but it can not deal with scale drift for the monocular system. Strasdat et al. [SMD10a] propose a optimization based on seven degrees of freedom (DoF) constraints instead of the traditional six DoF to solve this issue. A result is shown in Figure 2.4.

Although loop closure is common in existing large-scale SLAM systems, it is not applicable in autonomous driving. First, autonomous driving requires real-time scale correction on a per-frame basis. A delayed scale correction is not acceptable. Second, the purpose of autonomous driving is to transport passengers from the starting point to the destination. Loop closure rarely occurs in practice. Given the above, we adopt another approach, ground plane estimation, to allow our monocular system to correct the scale.

2.2.2 Ground Plane Estimation

As with ours, other systems handle scale drift by estimating camera height above the ground plane [SS08, GZS11, KRC+11]. However, they usually rely on triangulation or homography decomposition from feature matches that are noisy for low-textured road surfaces, or do not provide unified frameworks for including multiple cues. An example image is shown in Figure 2.5. As we can see, the ground plane often refers to a low-textured road surface, which renders sole reliance on such feature matches impractical.



(a) before optimisation (b) 6 DoF optimisation

(c) 7 DoF optimisation (d) aerial photo

Figure 2.4: Results before and after loop closure detection. (a) The trajectory and the map computed without loop closure. (b) A traditional graph optimization closes the loop but leaves the scale drift unchanged. (c) The result generated by performing seven DoF graph optimization proposed by Strasdat et al. [SMD10a]. (d) Aerial photo of the trajectory.

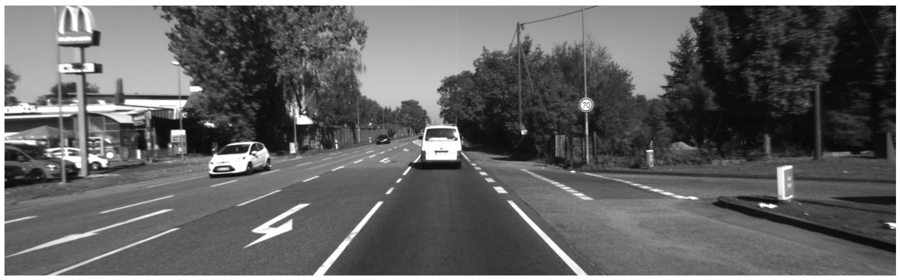


Figure 2.5: A frame from the KITTI odometry benchmark dataset. As we can see, the ground plane often refers to a low-textured road surface, which renders sole reliance on such feature matches impractical.

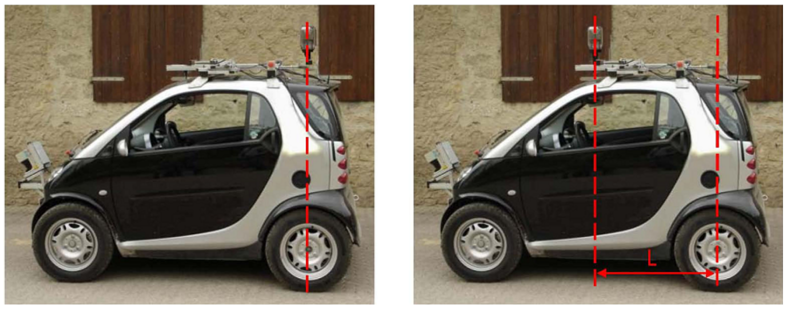


Figure 2.6: Compute the absolute scale by special mounting camera. (Left) If the camera is located on the vehicle's non-steering axle, the rotation and translation of both the camera and the car are exactly the same. The motion estimation of the vehicle is simplified in this case. (Right) A camera is mounted with an offset to the axle, so rotation and translation of camera and car are different. This special case can be used to compute the absolute scale from a single camera.

In contrast, we achieve far superior results by combining cues from both sparse features and plane-guided dense stereo, in a data-driven framework whose observation covariances are weighted by instantaneous visual data. In contrast to most of the above systems, we present strong monocular SLAM results on publicly available real-world driving benchmarks over several kilometers [GLU12, DG09] and report accurate localization performance relative to ground truth in Chapter 6.

2.2.3 Others

There are also methods used to counter scale drift that have special designed hardware or prior knowledge of the environment, such as nonholonomic constraints for wheeled robots [SFPS09] or the geometry of circular pipes [HARB11].

In Scaramuzza's work [SFPS09], a camera is mounted with an offset to the axle, so rotation and translation of camera and car are different. This special case can be used to compute the absolute scale from a single camera, as shown in Figure 2.6.

Hansen's work [HARB11] enforces all world points observed in the images to lie on the interior surface of a straight cylindrical pipe with constant radius. The radius of the pipe provides the absolute scale for the visual odometry system, as shown in Figure 2.7.

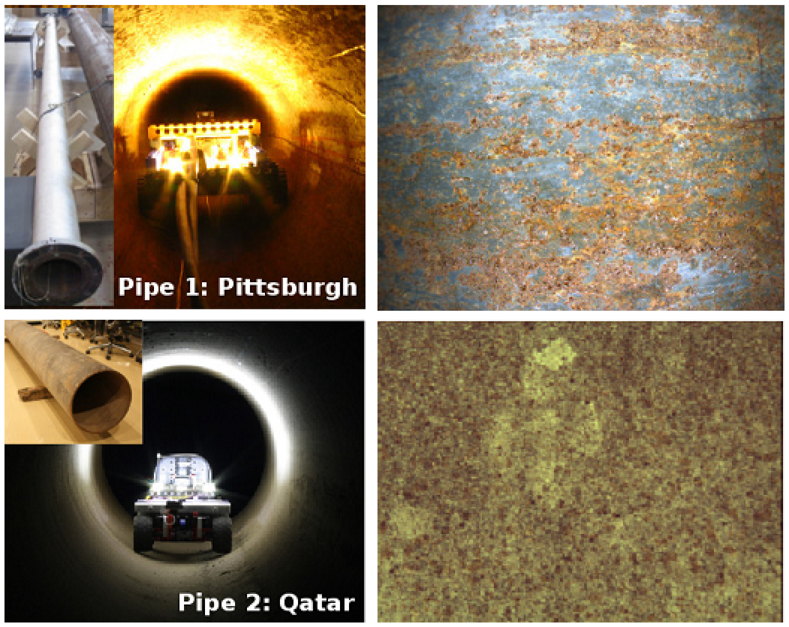


Figure 2.7: Robotics that use the pipe radius to estimate the absolute scale. (Left) The two pipes and robotic platforms. (Right) Example images from camera. The radius of the pipe provides the absolute scale of the visual odometry system.

2.3 Stereo Visual Odometry

Stereo-based SLAM systems now routinely achieve real-time performance in both indoor [CLFP10] and outdoor environments [NNB04], or even extraterrestrial terrains [OMSM01]. Parallel implementations for visual stereo SLAM that harness the power of GPUs have been demonstrated to achieve frame rates exceeding 30 fps in indoor environments [CLFP10].

A more complete presentation of stereo visual odometry can be found in Scaramuzza's visual odometry tutorial [SF11]. Our work focuses on the monocular visual odometry. We summarize the core part here for completion.

Building upon Moravec's [Mor80] work, Matthies [MS87] replaced the sliding camera with stereo cameras and used Moravec's detection and tracking of corners. Matthies took advantage of the error covariance matrix of the triangulated features and incorporated it into the motion estimation step. They reached superior results for a planetary rover with 2% relative error on a $5.5m$ path. Olson et al [OMSM00] [OMSM03] later extended the work by integrating an absolute orientation sensor (e.g., compass or omnidirectional camera) and and using the Forstner corner detector, which is significantly faster to compute than Moravec's operator. When the absolute orientation sensor is incorporated, the error growth can be reduced to a linear function of the distance traveled. This led them to a relative error of 1.2% on a $20m$ path.

Lacroix et al. [LMCG99] used dense stereo to replace the Forstner detector and then selected the candidate key points by analyzing the correlation function around its peaks – an approach that was later exploited in [MS06, How08]. Later, Cheng et al. used this approach in their final VO implementation for the Mars rovers [CMM05, MCM07]. They also improved earlier implementations by using RANSAC in the least-squares motion estimation step for outlier rejection. A different approach of outlier removal was proposed by Milella and Siegward [MS06]. In their implementation, an outlier removal stage was integrated into the iterative closest point (ICP) algorithm [BM92].

All the works cited above have in common that they triangulated 3D points from every stereo pair, and the relative motion is solved as a 3D-to-3D registration problem. Nister [NNB04] first proposed to use Perspective-n-Point (PnP) algorithms [HLON91] to solve the camera pose estimation problem using 3D-to-2D matches.

Chapter 3

Preliminaries: Background

Broadly, a visual system is a collection of devices that transform measurements of light into a representation of the surrounding environment with spatial or material properties. Images in a camera depend on the geoemtry of the scene. Inferring that information from a set of images is no easy task. In this chapter, we review some of the important concepts and theories: projective geometry, epipolar geometry, homography geometry, and so on. These theories are the cornerstones of visual odometry systems. They equip vision technology to positively impact our quality of life. Large segments borrow liberally from Yi Ma et al. [MSKS03], where more detailed related material can be found.

3.1 Rigid-body Motion

Three dimensional Euclidean space $\mathbb{E}^3$ can be represented by a Cartesian coordinate system:

$$\mathbf{X} = [X, Y, Z]^T \in \mathbb{R}^3. \tag{3.1}$$

In principle, the motion of object can be specified by the trajectory of every single point on the object. However, for rigid objects, it is sufficient to specify motion with a translation and a rotation.

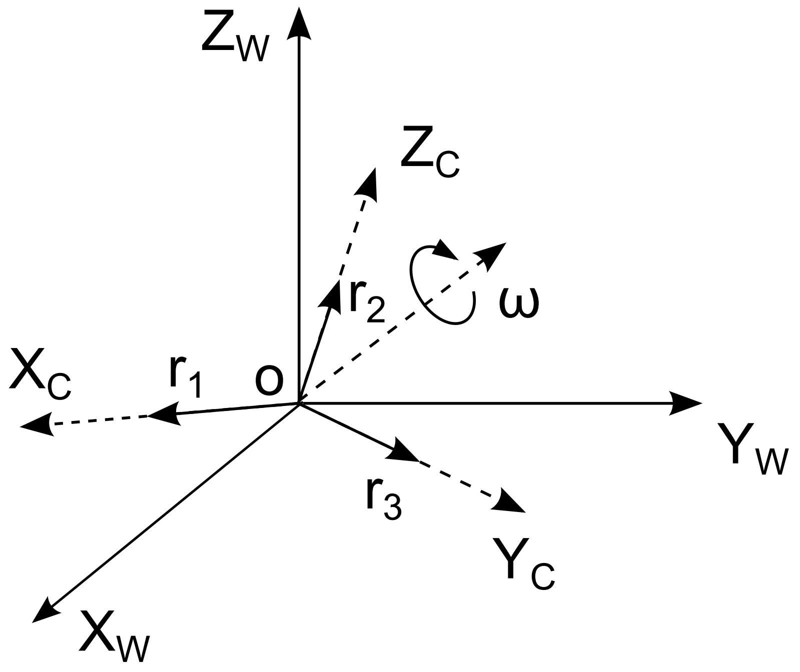


Figure 3.1: Rotation of a rigid body about the point o (the original point of the coorindate frame W and C) and along the axis ω. The coordinate frame W is before the rotation, and the coordinate frame C is after. The rotation is specified by two orthonormal vectors r_1, r_2, r_3.

3.1.1 Representations of Rotation

A 3D translation may be specified simply as a vector $\mathbf{T} = [T_x, T_y, T_z]^T$ giving the offset in each of the 3 coordinates. However, for a rotation, it's a little more complicated.

Rotation Matrix

Rotation in three dimensional Euclidean space can be represented by a rotation matrix:

$$R = [r_1, r_2, r_3] \in \mathbb{R}^{3\times 3}, \tag{3.2}$$

where r_1, r_2, r_3 are three orthonormal vectors relative to the frame W, that determine the configuration of the new coordinate frame C, as shown in Figure 3.1.

A rotation matrix has properties such as: $R^T R = I$, so it's an *orthogonal matrix*. Also the determinant of R must be 1. Hence R is a *special orthogonal matrix*. The space of all such special orthogonal matrices in $\mathbb{R}^{3\times3}$ is denoted by:

$$SO(3) = \{R \in \mathbb{R}^{3\times3} | R^T R = I, det(R) = 1\}. \tag{3.3}$$

To rotate a 3D point with the rotation matrix R, suppose a given 3D point $\mathbf{X}_W = [X_W, Y_W, Z_W]^T \in \mathbb{R}^3$ is in the coordinate frame W, we have:

$$\mathbf{X}_W = R_{WC}\mathbf{X}_C, \tag{3.4}$$

where $\mathbf{X}_C$ is the coordinate of the same point with respect to the frame C.

Euler Angles

The final rotation representation is Euler Angles. According to Euler's rotation theorem, Euler angles specify a rotation with a sequence of three elemental rotations. The three Euler angles are defined as follows, shown in Figure 3.2:

1. α is the angle between the x axis and the N axis.
2. β is the angle between the z axis and the Z axis.
3. γ is the angle between the N axis and the X axis.

To convert Euler angles to a rotation matrix, the rotation must be represented with three basic rotation matrices. The following three basic rotation matrices rotate vectors by an angle ω about the x, y, or z axis:

$$R_x(\omega) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\omega & \sin\omega \\ 0 & -\sin\omega & \cos\omega \end{bmatrix} \tag{3.5}$$

$$R_y(\omega) = \begin{bmatrix} \cos\omega & 0 & -\sin\omega \\ 0 & 1 & 0 \\ \sin\omega & 0 & \cos\omega \end{bmatrix} \tag{3.6}$$

$$R_z(\omega) = \begin{bmatrix} \cos\omega & \sin\omega & 0 \\ -\sin\omega & \cos\omega & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{3.7}$$

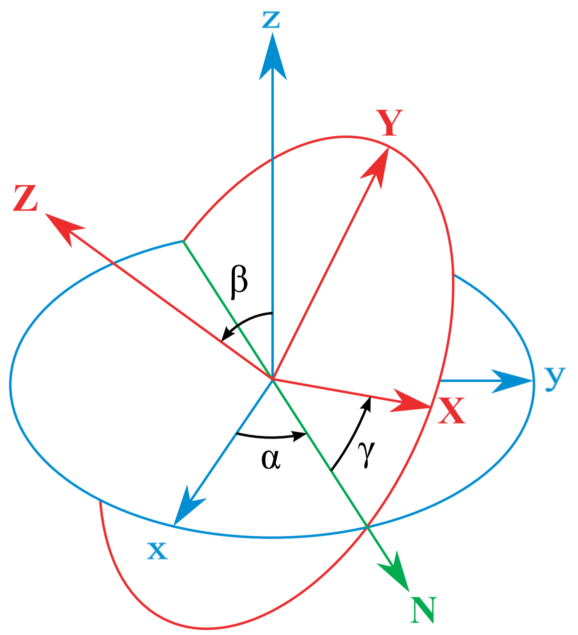


Figure 3.2: Proper Euler angles representing rotations about z, N, and Z axes. The xyz (original) system is shown in blue, the XYZ (rotated) system is shown in red. The line of nodes (N) is shown in green.

Therefore, the rotation specified with Euler angles α, β and γ can be representated with a rotation matrix such as:

$$R = \begin{bmatrix} \cos\gamma & \sin\gamma & 0 \\ -\sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\beta & \sin\beta \\ 0 & -\sin\beta & \cos\beta \end{bmatrix} \begin{bmatrix} \cos\alpha & \sin\alpha & 0 \\ -\sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{3.8}$$

The definition given above sometimes refers to the *classic* Euler angles. Another popular set of Euler angles are called *Tait-Bryan* angles or *Cardan* angles. It uses well-known yaw, pitch and roll angles (ϕ, θ, ψ) to define a rotation. Tait-Bryan angles have several different possibilities for choosing the order of rotation axes. In this section, we use "$z-y-x$”, as shown in Figure 3.3. The conversion of this to rotation matrix is:

$$R = \begin{bmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & -\sin\phi & \cos\phi \end{bmatrix}. \tag{3.9}$$

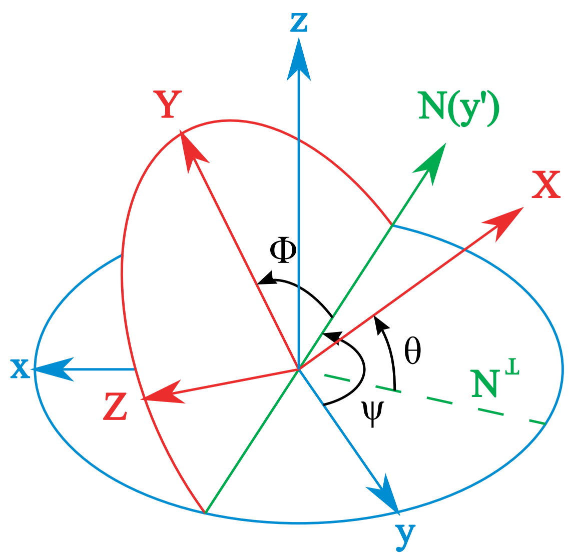


Figure 3.3: Tait-Bryan angles and rotations in a $z-y-x$ sequence.

Rotation Vector

Another method to represent a rotation in three dimensional Euclidean space is using a vector. A rotation by an angle $\|\omega\|$ around a fixed axis ω can be specified by the vector $\omega = [\omega_x, \omega_y, \omega_z]^T \in \mathbb{R}^3$, again as shown in Figure 3.1.

A rotation vector can be converted to a rotation matrix using Rodrigues' formula:

$$R = e^{\hat{\omega}} = I + \frac{\hat{\omega}}{\|\omega\|}\sin(\|\omega\|) + \frac{\omega\omega^T - I}{\|\omega\|^2}(1-\cos(\|\omega\|)), \tag{3.10}$$

where I is the 3×3 identity matrix, and $\hat{\omega}$ denotes the *cross-product matrix* of ω:

$$\hat{\omega} = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix}. \tag{3.11}$$

Rotation Quaternions

If we normalize the rotation vector ω to a unit vector θ, a rotation can be represented as the unit vector $[\theta_1, \theta_2, \theta_3]^T$ with the scale $\|\omega\|$.

Quaternions are defined by generalizing complex numbers in a fashion similar to how complex numbers are generalized from real numbers. We know that complex numbers are defined as $\mathbb{C} = \mathbb{R} + \mathbb{R}i$, with $i^2 = -1$. A set of quaternions, denoted by $\mathbb{H}$, is defined as:

$$\mathbb{H} = \mathbb{C} + \mathbb{C}j, \text{with } j^2 = -1 \text{ and } i \cdot j = -j \cdot i. \tag{3.12}$$

An element of $\mathbb{H}$ is then:

$$\mathbf{q} = q_0 + q_1 i + q_2 j + q_3 ij, \quad q_0, q_1, q_2, q_3 \in \mathbb{R}. \tag{3.13}$$

A special subgroup of $\mathbb{H}$ is called the *unit quaternions* defined as:

$$\|\mathbf{q}\|^2 = q_0^2 + q_1^2 + q_2^2 + q_3^2 = 1. \tag{3.14}$$

Given a rotation matrix $R = e^{\hat{\omega}}$, normalized unit vector θ and the scale $\|\omega\|$, we can associate a unit quaternion with it as:

$$q(R) = \cos(\frac{\|\omega\|}{2}) + \sin(\frac{\|\omega\|}{2})(\theta_1 i + \theta_2 j + \theta_3 ij) = q_0 + q_1 i + q_2 j + q_3 ij. \tag{3.15}$$

$\mathbf{q} = [q_0, q_1, q_2, q_3]^T$ is called *rotation quaternions*. We want to highlight the advantage of the rotation quaternions. Compared to Euler angles, they are simpler to compose and avoid the problem of *gimbal lock*.

Gimbal lock is the loss of one degree of freedom in a three-dimensional, three-gimbal mechanism that occurs when the axes of two of the three gimbals are driven into a parallel configuration, "locking" the system into rotation in a degenerate two-dimensional space. The problem of gimbal lock appears when one uses Euler angles in rotation representations. Gimbal lock occurs because the map from Euler angles to rotations is not a covering map – it is not a local homeomorphism at every point, and thus at some points the rank (degrees of freedom) must drop below 3, at which point gimbal lock occurs.

Compared to rotation matrices, they are more numerically stable and may be more efficient. They are the recommended representation of rotations in optimization problems, such as bundle adjustment (discussed in Section 3.5) [TMHF00]. Quaternions have found their way into applications in computer graphics, computer vision, robotics, navigation, molecular dynamics, flight dynamics, and orbital mechanics of satellites.

3.1.2 Rigid-body Motion

In sections above, we showed that a rigid body motion consists of a translation and a rotation. The rotation denotes a special orthogonal group $SO(3)$, and has various representations, such as rotation matrix, rotation quaternions and Euler Angles.

In this section, we introduce the rigid-body motion. Suppose a given 3D point $\mathbf{X}_W = [X_W, Y_W, Z_W]^T \in \mathbb{R}^3$ given with respect to the coordinate frame W, then we have:

$$\boxed{\mathbf{X}_W = R_{WC}\mathbf{X}_C + T_{WC}}, \tag{3.16}$$

where $\mathbf{X}_C$ is the coordinate of the same point with respect to the frame C.

The set of all possible configurations of a rigid body can be described by the space of rigid-body motions or special Euclidean transformations:

$$SE(3) \doteq \{g = (R,T) | R \in SO(3), T \in \mathbb{R}^3\}. \tag{3.17}$$

To introduce a matrix representation $g = (R,T)$ for $SE(3)$, we need to define homogeneous coordinates by appending a “1” to the coordinates $\mathbf{X} = [X,Y,Z]^T \in \mathbb{R}^3$ of a point $p \in \mathbb{E}^3$. That yields a vector in $\mathbb{R}^4$:

$$\tilde{\mathbf{X}} \doteq \begin{bmatrix} \mathbf{X} \\ 1 \end{bmatrix} = \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \in \mathbb{R}^4. \tag{3.18}$$

Using the new notation, the transformation of Equation 3.16 can be rewritten in a linear form as:

$$\boxed{\tilde{\mathbf{X}}_W = \begin{bmatrix} \mathbf{X}_W \\ 1 \end{bmatrix} = \begin{bmatrix} R_{WC} & T_{WC} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{X}_C \\ 1 \end{bmatrix} \doteq \tilde{g}\tilde{\mathbf{X}}_C.} \tag{3.19}$$

So finally, we have a natural matrix representation of the special Euclidean transformations:

$$SE(3) \doteq \{\tilde{g} = \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix} | R \in SO(3), T \in \mathbb{R}^3\} \subset \mathbb{R}^{4\times 4}. \tag{3.20}$$

3.2 Projective Camera

3.2.1 Perspective Camera without Distortion

A camera is an optical instrument that records images. Typically, a camera is composed of a set of lenses with optical distortions. In this section, we discuss the perspective camera model without taking non-linear distortions into consideration. This is defined by the *intrinsic matrix*:

$$K = \begin{bmatrix} fs_x & fs_\theta & o_x \\ 0 & fs_y & o_y \\ 0 & 0 & 1 \end{bmatrix}, \tag{3.21}$$

where f is the camera *focal length*. s_x and s_y are called the *scaling factors*. They define the size of the pixel (in metric units) along the x and y direction. When $s_x = s_y$, each pixel is square. s_θ is called a *skew factor* and it proportional to $\cot(\theta)$, where θ is the angle between the image axes x and y. (o_x, o_y) are the coordinates (in pixels) of the principal point relative to the image reference frame.

Thus, the model to transform homogeneous coordinates of a 3D point to homogeneous coordinates of its image is:

$$\lambda \tilde{\mathbf{x}}' = \lambda \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = K\Pi_0 \tilde{\mathbf{X}} = \begin{bmatrix} fs_x & fs_\theta & o_x \\ 0 & fs_y & o_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}, \tag{3.22}$$

where $\lambda = Z$ is written as an arbitrary positive scalar $\lambda \in \mathbb{R}_+$, since Z (the depth of the point) is usually unknown.

To take the motion of the camera into consideration, a given 3D point $\mathbf{X}_W = [X_W, Y_W, Z_W]^T \in \mathbb{R}^3$ is given with respect to the world coordinate frame W, and the camera introduces its coordinate frame C by moving $g = (R_{WC}, T_{WC})$ with respect to the world coordinate frame. Based on Equation 3.19, we have

$$\tilde{\mathbf{X}}_C = \begin{bmatrix} R_{CW} & T_{CW} \\ 0 & 1 \end{bmatrix} \tilde{\mathbf{X}}_W, \tag{3.23}$$

where

$$\begin{bmatrix} R_{CW} & T_{CW} \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} R_{WC} & T_{WC} \\ 0 & 1 \end{bmatrix}^{-1}, \tag{3.24}$$

so $R_{CW} = R_{WC}^T$ and $T_{CW} = -R_{WC}^T T_{WC}$.

Putting Equation 3.23 and Equation 3.22 together, we have

$$\lambda \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} fs_x & fs_\theta & o_x \\ 0 & fs_y & o_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} R_{CW} & T_{CW} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} X_W \\ Y_W \\ Z_W \\ 1 \end{bmatrix} \tag{3.25}$$

In matrix form, we have

$$\boxed{\lambda \tilde{\mathbf{x}}' = K \Pi_0 \tilde{g} \tilde{\mathbf{X}}_W}. \tag{3.26}$$

In summary, at this stage, we know how to project a 3D point in the world coordinate frame to an image with known camera rigid motion and camera calibrated intrinsic matrix. In the next section, we'll discuss the camera distortion issue.

3.2.2 Camera Distortion

In camera intrinsic matrix K, we can only handle linear distortions. In reality, if a camera with a wide field of view is used, significant non-linear distortion can often be observed. Non-linear distortion is handled by introducing a radial distortion model and a tangential distortion model. This model is used by *Camera Calibration Toolbox for Matlab*.

Radial Distortion

Radial distortion is defined by additional parameters (a_{r1}, a_{r2}, a_{r3}). Given undistorted coordinates (x, y), to incorporate the distortion, distorted coordinates (x_d, y_d) can be computed by:

$$\begin{bmatrix} x_d \\ y_d \end{bmatrix} = (1 + a_{r1} r^2 + a_{r2} r^4 + a_{r3} r^6) \begin{bmatrix} x \\ y \end{bmatrix}, \tag{3.27}$$

where $r^2 = x^2 + y^2$.

Tangential Distortion

Tangential distortion is defined by two additional parameters (a_{t1}, a_{t2}). Given undistorted coordinates (x, y), to incorporate the distortion, distorted coordinates (x_d, y_d) can be computed by:

$$\begin{bmatrix} x_d \\ y_d \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} 2a_{t1}xy + a_{t2}(r^2 + 2x^2) \\ a_{t1}(r^2 + 2y^2) + 2a_{t2}xy \end{bmatrix}, \tag{3.28}$$

where again $r^2 = x^2 + y^2$.

Putting Equation 3.27 and Equation 3.28 together, we have

$$\begin{bmatrix} x_d \\ y_d \end{bmatrix} = (1 + a_{r1}r^2 + a_{r2}r^4 + a_{r3}r^6) \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} 2a_{t1}xy + a_{t2}(r^2 + 2x^2) \\ a_{t1}(r^2 + 2y^2) + 2a_{t2}xy \end{bmatrix} \tag{3.29}$$

Summary

By now, we have built complete theories to project 3D points in the world to a camera image with distortion. Through these procedures, (x_d, y_d) are the observed coordinates in a distorted image in practice.

3.3 Epipolar Geometry

In the sections above, we discussed rigid motion and camera perspective. This solved the problem of how to project a 3D point in the world to an image in the camera. In the following two sections, we discuss the constraints of the projections of a 3D point in two views.

One important constraint on the pixels between two views is the *epipolar constraint*. If we call $\mathbf{x}_1, \mathbf{x}_2$ corresponding points in two views, the epipolar constraint says: Given $\mathbf{x}_1$ and the relative pose between two views (R, T), a *epipolar line* where $\mathbf{x}_2$ must reside on can be computed. In Section 4.1.2 of Chapter 4, the epipolar searching of the epipolar update module in our system is based on the epipolar constraint.

3.3.1 Calibrated Camera

In this section, we discuss the epipolar geometry of a calibrated case. For a calibrated camera, the intrinsic parameters of the camera are known. For simplicity, we assume the intrinsic matrix K is the identity matrix in the following discussions.

Assume the 3D coordinates of a 3D point relative to the two coordinate frames are $\mathbf{X}_1 \in \mathbb{R}^3$ and $\mathbf{X}_2 \in \mathbb{R}^3$. From Section 3.1, we know that they are related by a rigid-body transformation:

$$\mathbf{X}_2 = R\mathbf{X}_1 + T. \tag{3.30}$$

Assuming the camera intrinsic matrix K is the identity and λ_1, λ_2 are the unknown depths of the two views, we have:

$$\lambda_2 \tilde{\mathbf{x}}_2 = R\lambda_1 \tilde{\mathbf{x}}_1 + T. \tag{3.31}$$

Multiply both sides with $\tilde{\mathbf{x}}_2^T$ and the cross product $\hat{T}$ of T, then we have:

$$\tilde{\mathbf{x}}_2^T \lambda_2 \hat{T} \tilde{\mathbf{x}}_2 = \mathbf{x}_2^T \hat{T} R \tilde{\mathbf{x}}_1. \tag{3.32}$$

Since $\hat{T}\tilde{\mathbf{x}}_2 = T \times \tilde{\mathbf{x}}_2$ is perpendicular to $\tilde{\mathbf{x}}_2$, we have proven:

$$\boxed{\tilde{\mathbf{x}}_2^T \hat{T} R \tilde{\mathbf{x}}_1 = 0}\,. \tag{3.33}$$

The matrix $E \doteq \hat{T}R \in \mathbb{R}^{3\times 3}$ is called the *essential matrix*. The epipolar geometry is shown in Figure 3.4. Given an essential matrix $E = \hat{T}R$, it defines an epipolar relationship between two views. The intersections of the line (o_1, o_2) with each image plane are called *epipoles*, denoted by (e_1, e_2). The plane (o_1, o_2, p) is called an *epipolar plane*. The intersection between the epipolar plane and the two views are two lines $(\mathbf{l}_1, \mathbf{l}_2)$, called the *epipolar line*. As shown in the figure, given $\mathbf{x}_1$ with unknown depth, the coordinate of its corresponding projection ($\mathbf{x}_2$ or $\mathbf{x}_2'$) lies on the epipolar line $\mathbf{l}_2$.

There are some important properties of epipoles and epiolar lines:

1. The two epipoles $(e_1, e_2) \in \mathbb{R}^3$ are the left and right null spaces of E:

$$e_2^T E = 0, \;\; E e_1 = 0. \tag{3.34}$$

2. To compute (e_1, e_2), we have $e_2 \sim T$ and $e_1 \sim R^T T$, where "$\sim$" indicates equality up to scalar factor.

3. x_2 lying on $\mathbf{l}_2$ can be expressed as $\tilde{\mathbf{x}}_2^T \mathbf{l}_2 = 0$, so we can compute $\mathbf{l}_2$ by:

$$\mathbf{l}_2 \sim E\tilde{\mathbf{x}}_1 \in \mathbb{R}^3. \tag{3.35}$$

   Similarly, we have:

$$\mathbf{l}_1 \sim E^T \tilde{\mathbf{x}}_2 \in \mathbb{R}^3. \tag{3.36}$$

4. In each image, (e_1, e_2) lie on the epipolar line $(\mathbf{l}_1, \mathbf{l}_2)$, respectively, thus we have:

$$\mathbf{l}_1^T e_1 = 0, \;\; \mathbf{l}_2^T e_2 = 0. \tag{3.37}$$

Given $\mathbf{x}_1$ and (R, T), an epipolar line $\mathbf{l}_2$ that the corresponding $\mathbf{x}_2$ lies on can be computed. Sometimes, we also need to solve the problem such as: given a set of matches ($\mathbf{x}_1, \mathbf{x}_2$), how can we find E? E consists of R and T. R has three degrees of freedom, and T has two, thus E has 5 degrees of freedom. Therefore, we need at least five matches to compute E, which leads to Nister's famous 5-point algorithm. Please refer to [Nis04] for details. Another available algorithm to compute the essential matrix is the 8-point algorithm, which is more straightforward and less complicated than the 5-point algorithm for understanding. A brief description of the 8-point algorithm can be found in Appendix A.1.

3.3.2 Uncalibrated Camera

In the previous section, we assume the camera intrinsic matrix is known. When the camera intrinsic matrix is unknown, we face the uncalibrated case. Although, our visual odometry system requires known camera parameters to work, we also briefly introduce the epipolar constraint under the uncalibrated condition for completeness.

From Equation 3.22, we know $\lambda \tilde{\mathbf{x}} = K\mathbf{X}$, where $\tilde{\mathbf{x}}$ is the homogeneous representation of the 2D projection and $\mathbf{X}$ is the 3D point. Substitute this into Equation 3.30, it becomes:

$$\lambda_2 K_2^{-1} \tilde{\mathbf{x}}_2 = R\lambda_1 K_1^{-1} \tilde{\mathbf{x}}_1 + T. \tag{3.38}$$

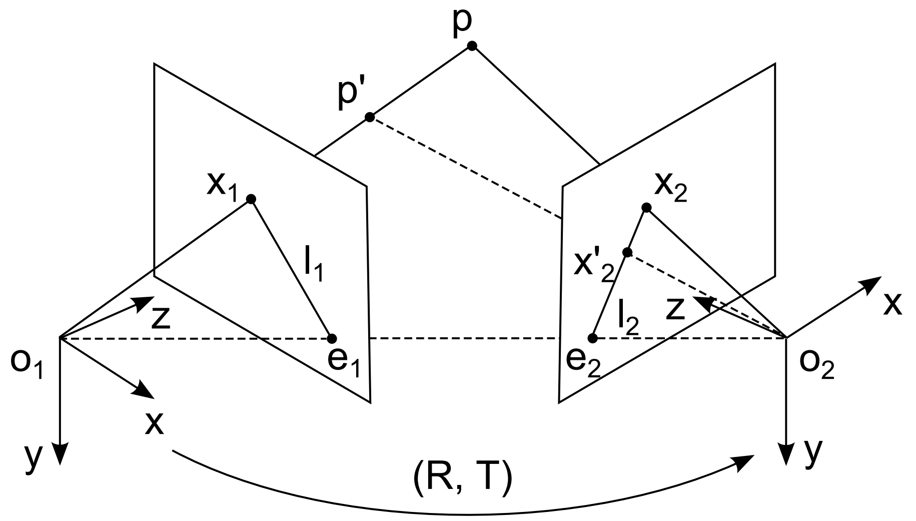


Figure 3.4: The epipolar geometry. Two projections $\mathbf{x}_1, \mathbf{x}_2 \in \mathbb{R}^3$ of a 3D point p. The rigid transformation between the two views is $(R, T) \in SE(3)$. The intersections (e_1, e_2) of the line (o_1, o_2) with each image plane are called *epipoles*. The lines $(\mathbf{l}_1, \mathbf{l}_2)$ are called *epipolar lines*, which are the intersection of the plane (o_1, o_2, p) with the two image planes. Given $\mathbf{x}_1$ with unknown depth, the coordinate of its corresponding projection ($\mathbf{x}_2$ or $\mathbf{x}_2'$) lies on the epipolar line $\mathbf{l}_2$.

Similarly, multiply both sides with $(K_2^{-1}\tilde{\mathbf{x}}_2)^T$ and the cross product $\hat{T}$ of T, and then because $\hat{T}K_2^{-1}\tilde{\mathbf{x}}_2 = T \times (K_2^{-1}\tilde{\mathbf{x}}_2)$ is perpendicular to $K_2^{-1}\tilde{\mathbf{x}}_2$, we have:

$$\boxed{\tilde{\mathbf{x}}_2 K_2^{-T} \hat{T} R K_1^{-1} \tilde{\mathbf{x}}_1 = 0}\ . \tag{3.39}$$

$F = K_2^{-T}\hat{T}RK_1^{-1}$ is called the *fundamental matrix*. Similar to the calibrated case, we may interpret the equation $\mathbf{l}_2 \sim F\tilde{\mathbf{x}}_1$ as F transferring a point in the first view to a line in the second view.

3.3.3 Summary

In summary, in this section, we discussed an important geometric constraint for two corresponding coordinates $\mathbf{x}_1, \mathbf{x}_2$ at two views. Regardless of the depth of the 3D point p, given $\mathbf{x}_1$ and (R, T), an epipolar line $\mathbf{l}_2$ where the corresponding $\mathbf{x}_2$ lies on can be computed with Equation 3.35. This can significantly boost the efficiency of the feature matching in Section 4.1.2 of Chapter 4.

3.4 Homography Geometry

The projections of the 3D point in general positions in two views comply with the epipolar constraints. If we have more knowledge (or assumptions) of the 3D points, what else can we do? In this section, we discuss the homography geometry. It can be applied when we believe a set of 3D points are in the same plane. In Chapter 5, we use this assumption to estimate the ground plane for the scale drift correction in our visual odometry system.

Starting from Equation 3.30 again, we have $\mathbf{X}_2 = R\mathbf{X}_1 + T$. Let $N = [n_1, n_2, n_3]^T$ be the unit normal vector of the plane P with respect to the first camera frame, and let d be the distance from the plane P to the first camera o_1, shown in Figure 3.5. Then we have:

$$\frac{1}{d} N^T \mathbf{X}_1 = 1. \tag{3.40}$$

Substituting Equation 3.40 into Equation 3.30 gives:

$$\mathbf{X}_2 = R\mathbf{X}_1 + T\frac{1}{d} N^T \mathbf{X}_1 = (R + \frac{1}{1} T N^T)\mathbf{X}_1. \tag{3.41}$$

We call the matrix $H \doteq = R + \frac{1}{1} T N^T$ the *homography matrix*. Let λ_1, λ_2 be the depth of the 3D point p with the respect to two views, respectively. Assuming the intrinsic matrix K is known and the identity gives:

$$\lambda_2 \tilde{\mathbf{x}}_2 = H \lambda_1 \tilde{\mathbf{x}}_1 \Rightarrow \tilde{\mathbf{x}}_2 \sim H \tilde{\mathbf{x}}_1. \tag{3.42}$$

Equation 3.42 can be interpreted as a pixel-to-pixel homography mapping induced by a plane, instead of a pixel-to-line mapping by the epipolar constraint. In Chapter 5, we build a similar homography mapping to find the optimal ground plane by varying the plane parameters.

The homography matrix H has 8 degrees of freedom. To compute the homography matrix, we need at least 4 pairs of matches in two views. Please refer to Appendix A.2 for details.

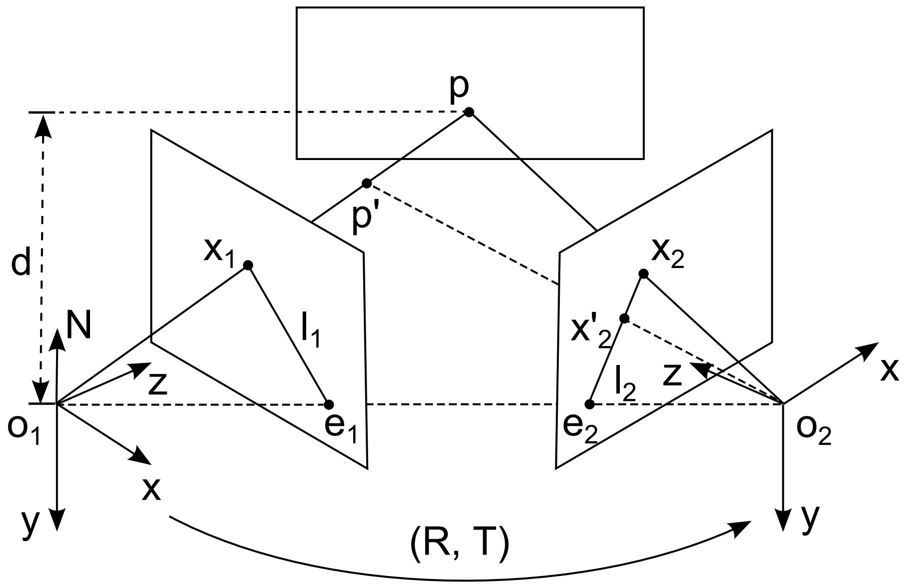


Figure 3.5: The homography geometry. Two projections $\mathbf{x}_1, \mathbf{x}_2 \in \mathbb{R}^3$ of a 3D point p on a plane P. They are related by a homography H that is introduced by the plane. $N = [n_1, n_2, n_3]^T$ is the unit normal vector of the plane P with respect to the first camera frame. d is the distance from the plane P to the first camera o_1.

3.5 Bundle Adjustment

Given a set of 3D points with their corresponding 2D feature matches in a set of views and the relative poses between these views, bundle adjustment is a set of optimization methods used in computer vision to refine both the 3D points and the view poses. Bundle adjustment is almost always used as the last step of every feature-based 3D reconstruction algorithm. V-SLAM systems typically use it to refine the 3D scene structure, and some visual odometry systems use it to refine the trajectory as well.

Let $P^i = [RT], i = 1 \cdots M$ are a set of view poses. Let $\mathbf{X_j}, j = 1 \cdots N$ are a set of 3D points. (u^i_j, v^i_j) are the 2D feature matches of the 3D points $\mathbf{X_j}$ in i-th view. The bundle adjustment solves an optimization problem, such as:

$$\min_{P^i, \tilde{\mathbf{X}}_j} \sum_{i,j} \omega^i_j((u^i_j - \frac{P^i_1 \tilde{\mathbf{X}}_j}{P^i_3 \tilde{\mathbf{X}}_j})^2 + (v^i_j - \frac{P^i_2 \tilde{\mathbf{X}}_j}{P^i_3 \tilde{\mathbf{X}}_j})^2), \tag{3.43}$$

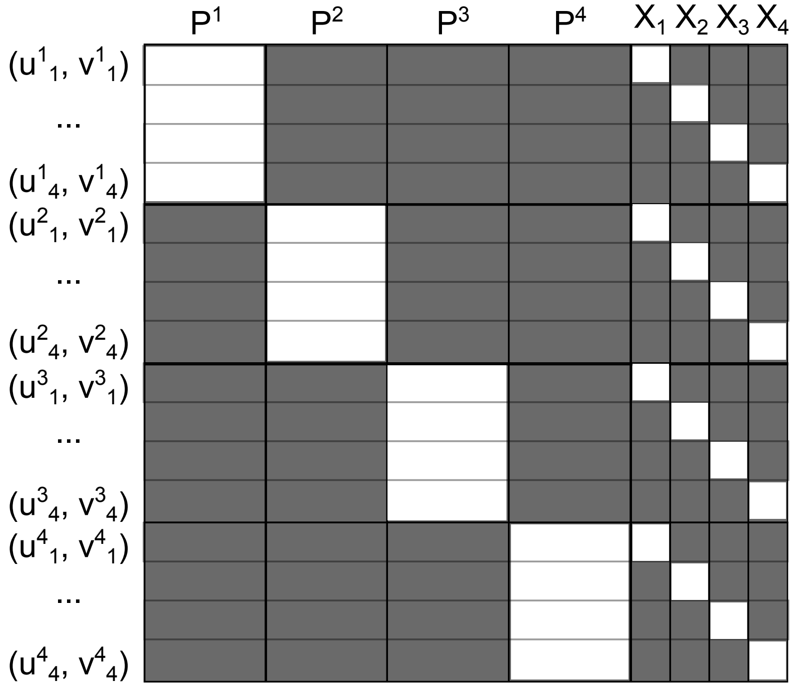


Figure 3.6: Sparse Jacobian matrix structure for a bundle adjustment. Structure of the a sparse Jacobian matrix for a bundle adjustment problem consisting of 4 cameras and 4 3D points. The gray entries are all zero.

where

$$P_{3\times 4} = \begin{bmatrix} R & T \end{bmatrix} = \begin{bmatrix} P_1 \\ P_2 \\ P_3 \end{bmatrix}. \tag{3.44}$$

As Triggs et al. pointed out in [TMHF00], there are some common misconceptions in the literature. The most important one may be **Bundle adjustment is slow**: These statements treat bundle adjustment as a general optimization problem using a general-purpose optimization routine that completely ignores the problem structure and sparseness, as shown in Figure 3.6. In practice, bundle adjustment is much more efficient than this. Therefore, unlike other visual odometry systems, effective use of bundle adjustment technology to refine both 3D structure and the trajectory through a novel multi-threaded design in our system yields good accuracy and efficiency. We'll discuss our bundle adjustment module design in Chapter 4.

3.6 Kalman Filter

The Kalman Filter is an algorithm that uses a series of measurements observed over time, containing noise and other uncertainties, and produces estimates of the unknown variables and their estimation uncertainties. An important advantage of the Kalman Filter is that it is a recursive algorithm, which means that it only requires the previous state and the current measurement to estimate the current new state. Compared to alternative approaches, such as least-square optimization, this advantage is obvious, and it has been proven that the estimation of the Kalman filter is optimal if all the assumptions hold (the Gaussian noise model, the covariance of both process and observation noise, and so on). It should be noted that it is possible to derive the same results using the least square argument [Jaz70]. Rudolf Kalman first introduced its theory in [KRE60].

Let $\mathbf{x}$ be the underlying variables to estimate. The Kalman Filter requires a state-transition model F to transit the underlying variables $\mathbf{x}_{k-1}$ to $\mathbf{x}_k$ by:

$$\mathbf{x}_k = F_k \mathbf{x}_{k-1} + B_k \mathbf{u}_k + \mathbf{w}_k, \tag{3.45}$$

where

1. $\mathbf{x}_k$ is the underlying variables to estimate.
2. F_k is the state transition model that transits $\mathbf{x}_{k-1}$ to $\mathbf{x}_k$.
3. $\mathbf{u}_k$ is the input control variables, which is optional (can always be 0).
4. B_k is the control-input model.
5. $\mathbf{w}_k$ is the process noise, which is assumed to be a zero mean multivariate Gaussian noise with covariance Q_k, such as $\mathbf{w}_k \sim N(0, Q_k)$.

The relationship between the underlying variables $\mathbf{x}$ and the observation (or measurement) $\mathbf{z}$ is given by:

$$\mathbf{z}_k = H_k \mathbf{x}_k + \mathbf{v}_k, \tag{3.46}$$

where

1. H_k is the observation model that transforms the state variables to the observed variables. When we can directly observe the state variables, H can be identity.

2. $\mathbf{v}_k$ is the observation noise, which is assumed to be zero mean multivariate Gaussian noise with covariance $\mathbf{U}_k$, such as $\mathbf{v}_k \sim N(0, U_k)$.

The state of the Kalman filter is represented by two variables:

1. $\mathbf{x}$ is the state variable.

2. P is the error covariance of the variable, which reflects its certainty.

The Kalman filter operates recursively on a series of noisy input data, and produces statistically optimal estimates of the underlying variables. The algorithm works in a two-step process. Before the measurement $\mathbf{z}_k$ is observed, the Kalman filter predicts the estimates of the current state variables $\mathbf{x}_{k|k-1}$ along with their uncertainties $P_{k|k-1}$ using the transition model F, input model B, input variables $\mathbf{u}$ and the process noise covariance Q. Here $\mathbf{x}_{k|k-1}$ is called the *priori* state estimate. They are the predicted estimates at time k given observations up to and including at time $k-1$. Similarly, we have a priori error covariance $P_{k|k-1}$.

Once the measurements $\mathbf{z}$ with their measurement uncertainties U are obtained, these a priori estimates $\mathbf{x}_{k|k-1}$ are updated to the *posteriori* estimates $\mathbf{x}_{k|k}$ in a way that more weight is given to the estimates with higher certainty, using the observation model H and the observation noise covariance U. $P_{k|k-1}$ is updated to $P_{k|k}$ as well. Therefore, the estimates are expected to be optimal.

The algorithm details are shown below:

1. **Predict**

   (a) Predicted priori state estimate

   $$\mathbf{x}_{k|k-1} = F_k \mathbf{x}_{k-1|k-1} + B_k \mathbf{u}_k. \tag{3.47}$$

   (b) Predicted priori estimate covariance

   $$P_{k|k-1} = F_k P_{k-1|k-1} F_k^T + Q_k. \tag{3.48}$$

2. **Update**

   (a) Measurement residual
$$\mathbf{y}_k = \mathbf{z}_k - H_k \mathbf{x}_{k|k-1}. \tag{3.49}$$

   (b) Residual covariance
$$S_k = H_k P_{k|k-1} H_k^T + U_k. \tag{3.50}$$

   (c) Kalman gain
$$K_k = P_{k|k-1} H_k^T S_k^{-1}. \tag{3.51}$$

   (d) Updated posteriori state estimate
$$\mathbf{x}_{k|k} = \mathbf{x}_{k|k-1} + K_k \mathbf{y}_k. \tag{3.52}$$

   (e) Updated posteriori estimate covariance
$$P_{k|k} = (I - K_k H_k) P_{k|k-1}. \tag{3.53}$$

The Kalman Filter is used in our system in ground estimation. Ground plane variables, such as pitch angle, ground height and so on, are the state variables and also the observed variables by each individual cue, such as 3D points triangulation, dense inter-frame stereo, and object detection. A novel data-driven mechanism is proposed to learn models from training data that relate observation covariances for each cue to error behavior of its underlying variables. During the system run, this allows per-frame adaptation of observation covariances based on relative confidences inferred from visual data. Our framework significantly boosts not only the accuracy of scale drift correction of the monocular visual odometry, but also that of applications of monocular object localization that rely on the ground plane. We'll discuss these in Chapter 5 and 7.

Chapter 4

Monocular Architectures

In this chapter, we describe the system architecture of our monocular structure from motion (SFM) system. The system has three different phases, steady state, keyframe and keyframe + 1. The architecture of our system in different phases is carefully designed to meet an unusual real-time requirement for a SFM system, that is our system is optimized to output pose within 50 ms in the worst case on a laptop with Intel Core i7 2.40 GHz processor with 8GB DDR3 RAM and 6M cache, while average case operation is over 30 fps (33ms). In contrast to this, typically, a SFM system may produce a spike in delays when keyframes are added, or loop closure is performed [CLFP10]. We also addressed the challenge of robust multithreading, even for scenes with large motions and rapidly changing imagery. Section 4.1 covers the system architecture of the steady state, and Section 4.2 covers the keyframe and keyframe + 1 phases.

4.1 Steady State Architecture

To initialize, the system extracts FAST corners [RPD10] with ORB descriptors [RRKB11b] and matches between consecutive frames using Fast Library for Approximate Nearest Neighbors (FLANN) [ML09] [ML12]. With sufficient baseline (around 5 frames), a set of 3D points is initialized by relative pose estimation [Nis04], triangulation and bundle adjustment. Each frame during initialization is processed within 10 ms.

At steady state, the system has access to a stable set of at least N_s 3D points that have undergone extensive bundle adjustment in prior frames (we choose $N_s = 100$).

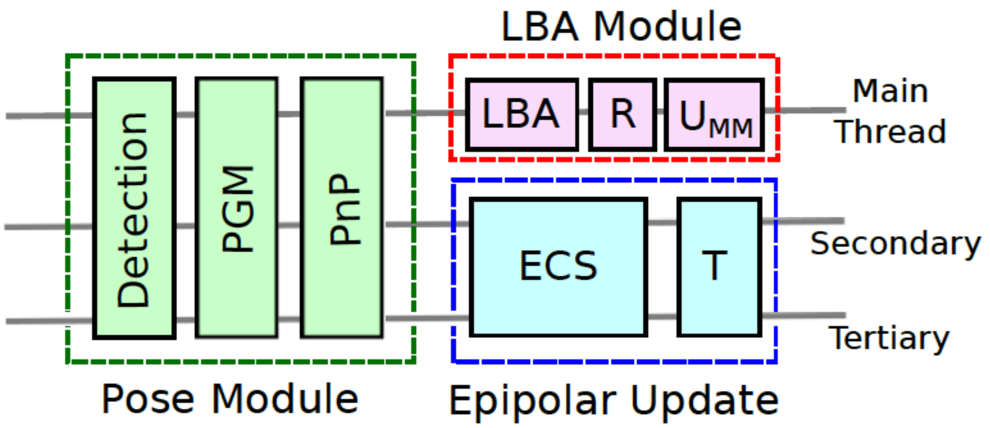


Figure 4.1: System architecture for every steady state frame. The acronyms above represent PGM: Pose-guided matching, LBA: local bundle adjustment, R: re-finding, U: Update motion model, ECS: Epipolar constrained search, T: triangulation. The modules are depicted in their multithreading arrangement, in correct synchronization order but not to scale.

The preceding poses have also undergone multiple non-linear refinements, and so can be considered highly accurate. The system architecture at every frame in steady state operation is illustrated in Figure 4.1.

4.1.1 Pose Module

Around 2000 FAST corners with Shi-Tomasi filtering [ST94b] are extracted from a typical outdoor image and ORB descriptors [RRKB11b] are computed. Using the pose of the previous frame, the pose of the current frame is predicted, assuming constant velocity. Note that this simple motion model is not an integral part of our estimation. It is only used to expedite matching by using previously estimated pose as guidance – we explicitly compute the camera pose at each frame using point correspondences. The existing set of stable 3D points are projected into the image using the predicted pose and the ORB descriptor for each is compared to those within a window of side $2r_s$ pixels (we choose $r_s = 15$). Given these 2D-3D correspondences, we compute the actual camera pose using perspective n-point (PnP) pose estimation in a robust RANSAC framework [FB81b]. We use the EPnP method of [LMNF09] with a model size of four points. The RANSAC pose with the largest consensus set is refined using a Levenberg-

Table 4.1: Timings for various stages of the pose module.

FAST corner detection + Shi-Tomasi	1 ms
ORB descriptor extraction	5 ms
Pose-guided matching	1 ms
PnP (RANSAC, 500 iterations)	15 ms
Nonlinear pose refinement	1 ms

Marquardt nonlinear optimization [LA09].

Our system can easily handle other choices for matching, in particular, we have achieved similar results using normalized cross-correlation (NCC) instead of ORB. But associating a descriptor like ORB with a 3D point can have ancillary benefits, as we will observe in the following sections.

Feature and descriptor extraction, pose-guided matching, and pose estimation are all easily computed in parallel across multiple threads, using a shared memory multiprocessing platform such as OpenMP [DM98]. Across three threads, the timings for various components of the pose module are summarized in Table 4.1.

4.1.2 Epipolar Update Module

If the application scenario involves scenes where the set of 3D points being viewed remains unchanged, then the pose module by itself would be sufficient to maintain the camera pose over extended periods. However, in outdoor applications such as autonomous navigation, 3D scene points rapidly move out of view within a few frames. Thus, the stable set of points used for pose computation must be continually updated, which is the task entrusted to our epipolar search module.

As depicted in Figure 4.1, the epipolar search module is computed in parallel across two threads and follows pose estimation at each frame. The mechanism for epipolar search is illustrated in Figure 4.2. Let the most recent prior keyframe be frame 0. After pose computation at frame n, for every feature f_0 in the keyframe at location (x_0, y_0), we consider a window of side $2r_e$ centered at $(x_0 + \Delta x, y_0 + \Delta y)$ in frame n, with r_e proportional to camera velocity. The displacement $(\Delta x, \Delta y)$ is computed based on the distance of (x_0, y_0) from the horizon (computed using the ground plane in Sec. 5). Adapting r_e and $(\Delta x, \Delta y)$ to the velocity helps in fast highway sequences, where disparity ranges can vary significantly between the far and near fields. Thus, an accurate ground plane estimation also has benefits for epipolar search.

We consider the intersection region of this square with a rectilinear band p pixels wide, centered around the epipolar line corresponding to f_0 in frame n. The ORB descriptors or NCC scores for all FAST corners that lie within this intersection region are compared to the descriptor for f_0. The closest match, f_n, is found in terms of Hamming distance. This epipolar matching procedure is also repeated by computing the closest match to f_n in frame $n-1$, call it f_{n-1}. A match is accepted only if f_{n-1} also matches f_0. Note that only two sets of matches with respect to frames $(0,n)$ and $(n-1,n)$ must be computed at the frame n, since the matches between $(0,n-1)$ have already been computed at frame $n-1$.

The parameter r_e is automatically determined by the size of the motion. We use $r_e = \min\{1200\|\boldsymbol{\omega}\|^2, 10\}$, where $\boldsymbol{\omega}$ is the differential rotation between frames $n-1$ and n. Since pose estimates are highly accurate due to continuous refinement by bundle adjustment (Sec. 4.1.3), epipolar lines are deemed accurate and we choose a stringent value of $p = 3$ to impose the epipolar constraint. The computation of the Hamming distance for 256-bit ORB descriptors or the NCC score in a region of interest is performed as a block, with a fast SSE implementation. To rapidly search for features that lie within the above region of interest, the detected features in an image are stored in a lookup table data structure. The key into the table is the column index of the feature and within each bucket, features are stored in sorted row order. Across two threads, this allows circular matching for a triplet of images, with up to 500 features in each, in $10-15$ ms. As opposed to a brute-force search, the lookup table results in speedups by up to a factor of 10, especially in our outdoor driving application where images traditionally have wide aspect ratios (to cover greater field of view while limiting uninformative regions such as the sky).

The features that are circularly matched in frame n are triangulated with respect to the most recent keyframe (frame 0). This two-view triangulation requires approximately 2 ms per frame. The reconstructed 3D point is back-projected in all the frames $1,\cdots,n-1$ and is retained only if a match is found within a very tight window of side $2r_b$ pixels (we choose $r_b = 3$). Working together with the local bundle adjustment in Section 4.1.3, this

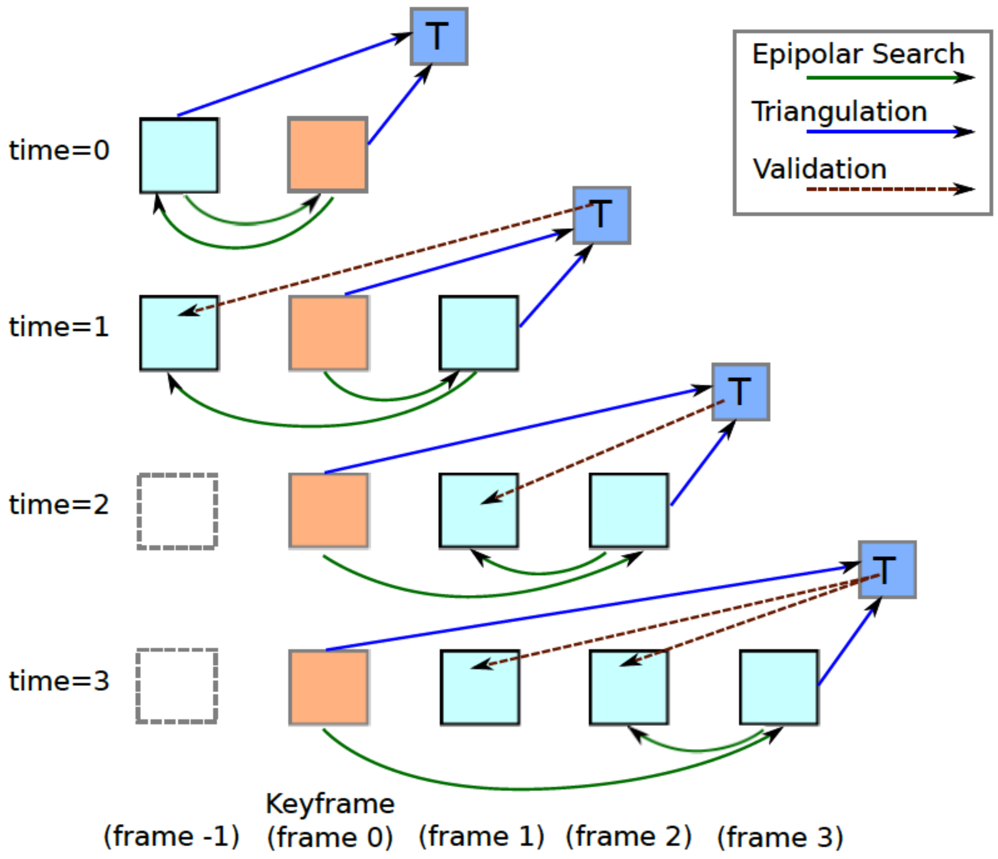


Figure 4.2: Mechanism of epipolar constrained search, triangulation and validation by reprojection to existing poses. For current frame n, only 3D points that are validated against all frames 1 to $n-1$ are retained. Only persistent 3D points that survive for greater than L frames may be collected by the next keyframe.

acts as a replacement for a more accurate, but expensive, multiview triangulation and is satisfactory since epipolar search produces a large number of 3D points, but only the most reliable ones may be used for pose estimation. However, these 3D points are not added to the stable point cloud yet. For that they must first undergo a local bundle adjustment and be collected by the main thread at a keyframe, which are aspects explained in the following sections. Also, note in Figure 4.2, the exception for the keyframe and frame 1, where validation is not possible. However, this is not an issue since only long tracks are eventually collected by the next keyframe, as explained in Section 4.2.

As an example of the architectural difference, PTAM uses the existing distribution of points to restrict epipolar search range, which is not desirable for fast-moving vehicles. It also performs epipolar search on demand and data association refinement is used to validate the point in other frames during free time on the mapping thread when exploring already seen regions. Our system does not have the leeway to revisit regions of the map, so all our refinement must be online. These considerations led us to move epipolar search to a separate thread of its own, but it presents a unique opportunity for enhancing robustness, as described in Section 4.1.2. Another distinction from PTAM necessary for autonomous driving is a bundle adjustment at every keyframe regardless of computational load of other tracking or mapping tasks. So we introduce novel keyframe architectures to accommodate timing constraints and maintain thread safety, as discussed in Section 4.2.

4.1.3 Local Bundle Adjustment Module

To refine camera poses and 3D points incorporating information from multiple frames, we implement a sliding window local bundle adjustment. The key data structure is the local bundle cache, which is composed of a frame cache and a match cache. The frame cache stores feature locations, descriptors and camera poses from the most recent N frames. It also stores images for those N frames, for display and debugging purposes. In our system, $N = 10$. The match cache is a list of tables, one element corresponding to each frame. The key into the table is the identity of a 3D point visible in the frame and the stored entries are the identities of the corresponding 2D features in various frames.

During small motions, the system prevents addition of new keyframes and ensures that the previous keyframe is included in the bundle cache. This guarantees that the

Table 4.2: Epipolar update and local bundle timings in steady state (parallel modules).

Module	Operation	Timing
Epipolar Update	Constrained search	$10-15$ ms
	Triangulation	$1-3$ ms
Local Bundle	Windowed bundle adjustment	$10-20$ ms
	Re-find 3D points	1 ms
	Update motion model	0 ms

baseline between the previous keyframe and the current frame does not become too small, which improves the stability of bundle adjustment and yields accurate pose estimates even in near-stationary situations.

After bundle adjustment, we give the system a chance to re-find lost 3D points using the optimized pose. Since the system spends considerable effort in maintaining a high-quality set of 3D points for pose compuation, it is worthwhile to incur a small overhead to recover any temporarily lost ones (due to image artifacts such as blur, specular reflections or shadows). In fact, a stable 3D point is permanently discarded only when its projection using the current pose falls outside the image boundaries. Since the bundle adjusted pose is highly accurate, we can perform re-finding by matching ORB descriptors on FAST corners within a very tight window of side $2r_f$ pixels (we choose $r_f = 10$). This ensures re-finding is rapidly achieved within 1 ms.

We use the publicly available SBA package [LA09] for bundle adjustment. In parallel, the motion model for predicting the pose of the next frame is also updated in this module. The timings for the parallel epipolar update and local bundle adjustment modules are summarized in Table 4.2.

4.1.4 Discussion

We highlight that besides the obvious speed advantages, moving epipolar search to a new thread also greatly contributes to the accuracy and robustness of the system.A system that relies on 2D-3D correspondences might update its stable point set by performing an epipolar search in the frame preceding a keyframe. However, the support for the 3D points introduced by this mechanism is limited to just the triplet used for the circular matching and triangulation, and the quality of these 3D points are not guaranteed. By performing the circular matching at every frame, we may supply 3D points with tracks of length up to the distance from the preceding keyframe. Additionally, further validation and outlier rejection mechanisms happen in this step. Clearly, the extensively validated set of long tracks provided by the epipolar thread in our multithread system is far more likely to be free of outliers, while contributing longer-range constraints for a more stable pose estimation.

Our novel multithreaded architecture also has efficiency advantages. Bundle adjustment has become a standard component in SFM or SLAM system [MLD$^+$06,KM07, WSS11]. Our system performs a local bundle and a keyframe bundle including the 10 and 5 most recent frames and keyframes, respectively. These two bundle adjustments are for better local accuracy. In our design, the epipolar module operates in parallel with the local bundle module. In contrast to large scale multithreaded bundle adjustment [WACS11], small scale bundle (for example, a local bundle having 10 views and hundreds of points) is not significantly faster with multithreading. The epipolar update module, thus, allows better 3D points while occupying the idle secondary and tertiary threads.

4.2 Keyframe and Recovery Architectures

4.2.1 Keyframe

The system cannot maintain steady state indefinitely, since 3D points are gradually lost due to tracking failures or when they move out of the field of view. The latter is an important consideration in "forward moving" systems for autonomous driving (as opposed to "browsing" systems such as PTAM), so the role of keyframes is very important in keeping the system alive. The purpose of a keyframe is threefold:

- Collect 3D points with long tracks from the epipolar thread, refine them with local bundle adjustment and add to the set of stable points in the main thread.
- Trigger global bundle adjustment based on the previous few keyframes that refines 3D points and keyframe poses.
- Provide the frame where newly added 3D points "reside".

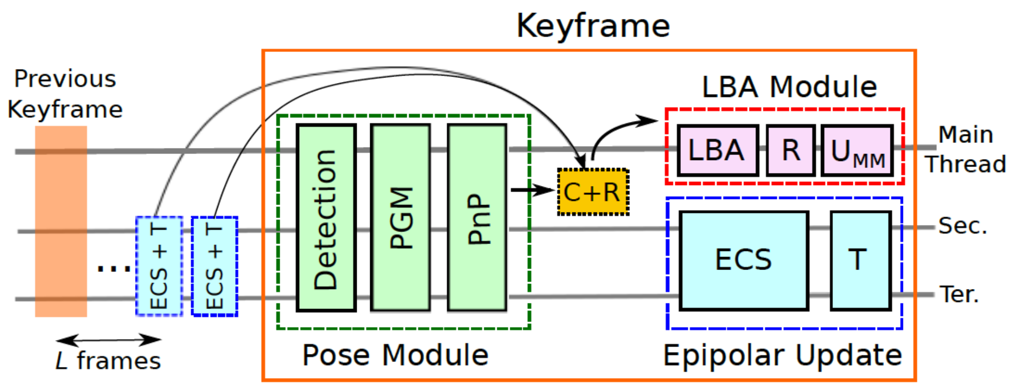


Figure 4.3: System architecture for keyframes. C+R stands for a collection and refinding module. It collates persistent 3D points tracked over at least L frames in the epipolar thread and re-finds them in the current frame using the output of the pose module. The LBA is now different from that for steady state, since its cache has been updated with 3D points and their corresponding 2D locations in all the relevant frames on the epipolar thread.

The modules that define operations at a keyframe are illustrated in Figure 4.3. The pose module remains unchanged from the steady state. It is followed by a collection stage, where 3D points triangulated at each frame in the epipolar thread are gathered by the main thread. Only persistent 3D points that stem from features matched over at least L frames are collected (our circular matching for epipolar search ensures this is easily achieved by seeking 3D points only from at least L frames after the previous keyframe). Note that this mechanism imposes two necessary conditions for a point to be considered for inclusion into the stable set – it must be visible in at least two keyframes and must be tracked over at least L frames. While stringent, these conditions inherently enhance the chances that only reliable 3D points are added into the main thread. In our system, $L = 3$ regardless of environment, although similar results are also obtained for L up to 5 in outdoor environments.

The collected 3D points must reside on a keyframe for all subsequent operations, so a re-finding operation is performed by projecting them using the estimated pose for the frame and finding the best ORB match in a circular region of radius 10 pixels. Now the existing stable 3D points, the collected 3D points from the epipolar thread, their projections in all the frames within the local bundle cache and the corresponding cameras undergo local bundle adjustment. Note that the bundle adjustment at keyframes differs from steady state operation, but adding long tracks into the bundle adjustment at keyframes is a reason we can avoid more expensive multiview triangulation at each frame in the epipolar thread. These refined 3D points are now added to the stable point set and can be tracked for pose computations in subsequent frames. The residence of these 3D points is the current keyframe and their 2D feature locations

The modules that define operations at the frame immediately after a keyframe are illustrated in Figure 4.4. The pose module re-finds the (new) set of stable 3D points. The RANSAC-based PnP will also discard outliers among the newly added 3D points. The pose module is now split across only two threads, in order to accommodate a global bundle adjustment in the main thread. This bundle adjustment involves the previous K keyframes and their associated 3D points, in order to introduce long-range constraints to better optimize the newly added set of 3D points. For our system, choosing $K = 5$ allows the global bundle adjustment to finish within 15 ms. There are two reasons a more expensive bundle adjustment involving a much larger set of previous keyframes (or even the whole map) is not necessary to refine 3D points with long-range constraints. First, the imagery in autonomous driving applications is fast moving and does not involve repetitions, so introducing more keyframes into the global bundle adjustment yields at best marginal benefits. Second, our goal is instantaneous pose output rather than map-building, so even keyframes are not afforded the luxury of delayed output. This is in contrast to parallel systems such as [CLFP10] where keyframes may produce a noticeable spike in the per-frame delays.

Following global bundle adjustment, the 3D coordinates of all the points are updated. Note that overlapping sets of 3D points are used by both global bundle adjustment and pose modules in parallel, however, both may also cause this set to change (PnP may reject 3D points that are outliers, while bundle adjustment may move the position of 3D points). To ensure thread safety, an update module is included that reconciles changes in the 3D point cloud from both the prior parallel modules. The local bundle adjustment module, which simply reads in 3D point identities, receives this updated set for optimization based on the N frames in the local bundle cache. In parallel with local bundle adjustment, the epipolar search also makes use of the updated keyframe pose.

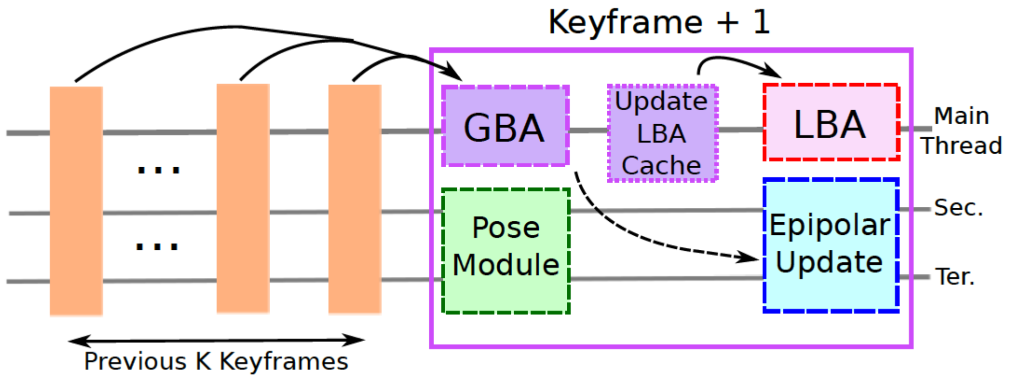


Figure 4.4: System architecture for frame following a keyframe. GBA stands for global bundle adjustment. Note that GBA usually finishes within the time consumed by the pose module. The cache update module reconciles the 3D points modified by both PnP and GBA, before it is used by LBA.

Note that while the keyframe pose has seen a global bundle adjustment, the pose of the subsequent frame has not. This does not cause any inconsistency in practice since poses tend to be much more stable than points – a camera is constrained by hundreds of points, but a point is visible only in a few cameras. Thereafter, the system resumes steady-state operation until the next keyframe, unless a recovery or firewall condition is triggered. The following sections explain those concepts in detail.

4.2.2 Error-Correcting Mechanisms

On rare occasions, the system might encounter a frame where pose-guided matching fails to find any features (due to imaging artifacts or a sudden large motion). In such a situation, a recovery mode is triggered, as illustrated in Figure 4.5. Let the frame where system recovery initiates be n and let k be the immediately preceding keyframe. During recovery, the frames $(n, n-1)$ are matched by comparing ORB descriptors over the entire image using fast FLANN and accepting only bidirectional matches. Relative pose is computed using the 5-point algorithm [Nis04] in a robust RANSAC framework and inlier matches are triangulated.

However, scale information is lost in the process. So, we also consider 3D points observed between frames $(n-1, k)$. Both the sets of 3D points are moved to the

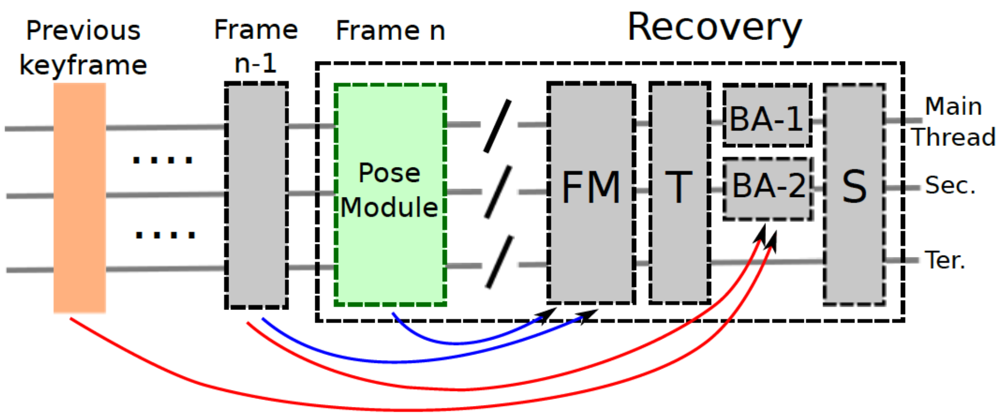


Figure 4.5: System architecture for a recovery frame. FM stands for Feature matching, BA-1 and BA-2 are bundle adjustments and S denotes scale recovery.

coordinate system of frame $n-1$ and a 1-point RANSAC is performed. The hypothesis for the RANSAC is the ratio of the norms of the sampled 3D point in the two sets. The corrected scale factor between frames $(n, n-1)$ is assigned as the average ratio in the largest consensus set. To ensure that 3D points used for scale recovery are as accurate as possible, two instances of bundle adjustments are run in parallel – one between frames $(n, n-1)$ and another between frames $(n-1, k)$. The system also has an alternative mechanism to recover the scale, which is to simply use the scale between frames $(n-1, k)$ for frames $(n, n-1)$. When the system doesn't keep repeating the recovery, this mechanism is used to recover the scale. For sequences in the KITTI dataset, recovery is required on an average once in 1500 frames.

4.3 Summary

We have presented a novel multithreaded system for largescale, real-time, monocular visual odometry, targeted towards autonomous driving applications with fast-changing imagery. Our key contribution is a demonstration that judicious multithreaded design can boost both the speed and accuracy for handling challenging road conditions. Our system is optimized to provide pose output in real-time at every frame, without delays for keyframe insertion or global bundle adjustment. This is achieved through a novel per-frame epipolar search mechanism that generates redundantly validated 3D points persistent across long tracks and an efficient keyframe architecture to perform online thread-safe global bundle adjustment in parallel with pose computation.

This chapter is based on “Parallel, real-time monocular visual odometry”, by Shiyu Song, Manmohan Chandraker, Clark Guest as it appears in proceedings of Robotics and Automation (ICRA), 2013 IEEE International Conference on, May 6-10 2013, Karlsruhe.

This chapter in part, has been submitted for publication, as it may appear in “High Accuracy Monocular SFM and Scale Correction for Autonomous Driving”, by Shiyu Song, Manmohan Chandraker, Clark C. Guest, in IEEE Transactions on Pattern Analysis and Machine Intelligence.

Chapter 5

Ground Plane Estimation

Scale drift correction is an integral component of monocular SLAM. In practice, it is the single most important aspect that ensures accuracy. We estimate the depth and orientation of the ground plane relative to the camera for scale correction.

Multiple methods like triangulation of sparse feature matches and dense stereo between successive frames can be used to estimate the ground plane. We propose a principled approach to combine these cues to reflect our belief in the relative accuracy of each cue. Naturally, this belief should be influenced by both the input at a particular frame and observations from training data. We achieve this by learning models from extensive training data to relate the observation covariance for each cue to the error behavior of its underlying variables. During testing, the error distributions at every frame adapt the data fusion observation covariances using those learned models. The framework is illustrated in Figure 5.1 and the following subsections discuss the technical details. In Section 5.1, we introduce some background for scale drift of a monocular SLAM system, cue fusion based on the Kalman filter and some notations we use. In Section 5.2, we describe how we collect observations from each method. In Section 5.3, we talk about how we trained the learning models and how we used them in testing.

5.1 Background

A vector in $\mathbb{R}^n$ is denoted in bold fonts, as $\mathbf{x} = (x_1, \cdots, x_n)^\top$. A matrix is denoted in bold capitals, as $\mathbf{X}$. The homogeneous representation of vector $\mathbf{x}$ is denoted as

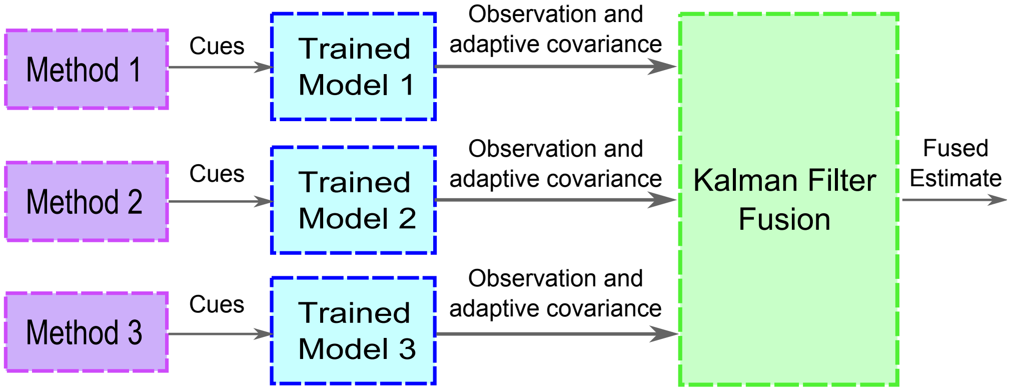


Figure 5.1: Adaptive cue combination framework. Trained models for each method relate their observation covariances to the error behaviors of underlying variables. These models allow determination of the relative confidence in each method during testing, based on per-frame visual data. A Kalman filter is used to fuse them and produce the optimal estimate.

$\widetilde{\mathbf{x}} = (\mathbf{x}^\top, 1)^\top$. A variable x in frame k of a sequence is denoted with superscripts, as x^k.

5.1.1 Ground Plane Estimation

As shown in Figure 5.2, the camera height (sometimes also called the ground height) h is defined as the distance from the camera principal center to the ground plane. Usually, the camera is not perfectly parallel to the ground plane and there exists a non-zero pitch angle θ. The unit normal vector $\mathbf{n} = (n_1, n_2, n_3)^T$ and the ground height h define the ground plane. For a 3D point $(X, Y, Z)^T$ on the ground plane, we have:

$$h = Y cos(\theta) - Z sin(\theta) \tag{5.1}$$

Under scale drift, any estimated length l is ambiguous up to a scale factor $s = \text{ł}/l^*$, where l^* is the ground truth length. The objective of scale correction is to compute s. Given the calibrated height of camera from ground h^*, computing the apparent height h yields the scale factor $s = h/h^*$. Then the camera translation $\mathbf{t}$ can be adjusted as $\mathbf{t}_{\text{new}} = \mathbf{t}/s$, thereby correcting the scale drift. In Section 5.2, we describe a novel, highly accurate method for estimating the ground height h using an adaptive cue combination mechanism.

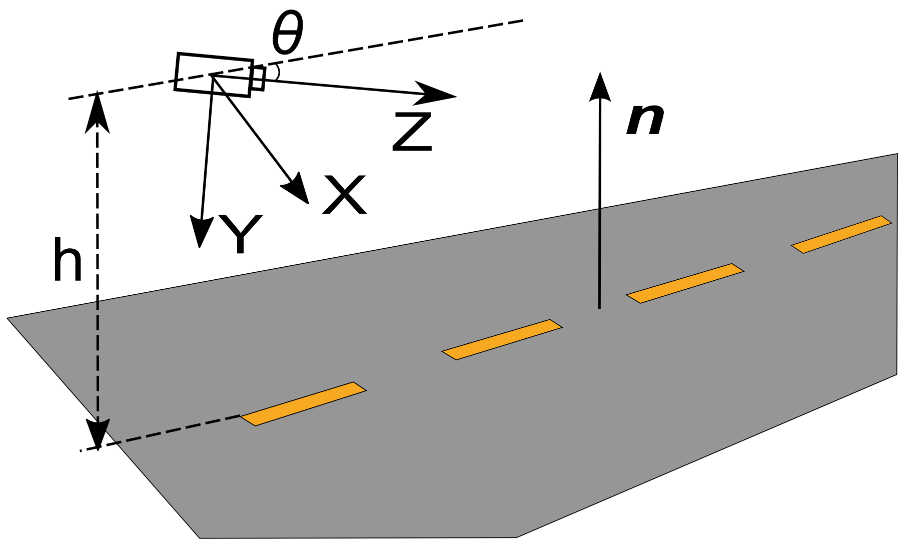


Figure 5.2: The geometry of ground plane estimation. The camera height h is defined as the distance from the camera principal center to the ground plane. The camera pitch angle is θ and $\mathbf{n}$ denotes the ground plane normal vector. Thus, the ground plane is defined by $\{\mathbf{n}, h\}$

.

5.1.2 Data Fusion with Kalman Filter

To combine estimates from various methods, a natural framework is a Kalman filter. Its model of state evolution is

$$\begin{aligned} \mathbf{x}^k &= \mathbf{A}\mathbf{x}^{k-1} + \mathbf{w}^{k-1}, \quad p(\mathbf{w}) \sim N(0, \mathbf{Q}), \\ \mathbf{z}^k &= \mathbf{H}\mathbf{x}^k + \mathbf{v}^{k-1}, \quad p(\mathbf{v}) \sim N(0, \mathbf{U}), \end{aligned} \tag{5.2}$$

where $\mathbf{x}$ is the state variable, $\mathbf{z}$ the observation, while $\mathbf{Q}$ and $\mathbf{U}$ are the covariances of the process and observation noise, respectively, that are assumed to be zero mean multivariate normal distributions. Our state variable in (5.2) is simply the equation of the ground plane, thus, $\mathbf{x} = (\mathbf{n}^\top, h)^\top$. Since $\|\mathbf{n}\| = 1$, n_2 is determined by n_1 and n_3 and our observation is $\mathbf{z} = (n_1, n_3, h)^\top$. Thus, our state transition matrix and the observation model are given by

$$\mathbf{A} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}^\top & 1 \end{bmatrix}^\top, \quad \mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{5.3}$$

Suppose methods $j = 1, \cdots, m$ are used to estimate the ground plane, each with its observation covariance $\mathbf{U}_j$. Then, with

$$\mathbf{U}^k = (\sum_{i=1}^{m} (\mathbf{U}_i^k)^{-1})^{-1}, \tag{5.4}$$

the fusion equations at time instant k are

$$\mathbf{z}^k = \mathbf{U}^k \sum_{j=1}^{m} (\mathbf{U}_j^k)^{-1} \mathbf{z}_j^k, \qquad \mathbf{H}^k = \mathbf{U}^k \sum_{j=1}^{m} (\mathbf{U}_j^k)^{-1} \mathbf{H}_j^k. \tag{5.5}$$

Meaningful estimation of $\mathbf{U}^k$ at every frame, with the correctly proportional $\mathbf{U}_j^k$ for each cue, is essential for principled cue combination. Traditionally, fixed covariances are used to combine cues, which does not account for the per-frame variation in the effectiveness of each cue across a video sequence. In contrast, in the following sections, we propose a rigorous data-driven mechanism to learn models to adapt per-frame covariances for each cue, based on error distributions of the underlying variables.

5.2 Cues for Ground Plane Estimation

We propose using multiple methods such as triangulation of sparse feature matches, dense stereo between successive frames and object detection bounding boxes to estimate the ground plane. The cues provided by these methods are combined in a principled framework that accounts for their per-frame relative effectiveness. In this section, we describe the cues and the next section describes their combination.

5.2.1 Plane-Guided Dense Stereo

We assume that a region of interest (ROI) in the foreground (middle fifth of the lower third of the image) corresponds to a planar ground. For a hypothesized value of

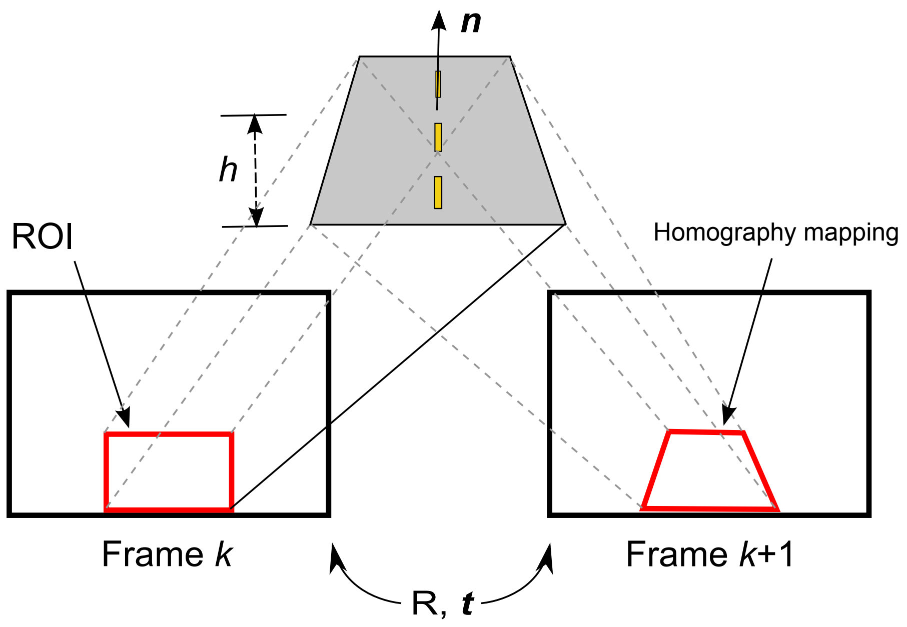


Figure 5.3: Homography mapping for plane-guided dense stereo. For a hypothesized ground plane $\{\mathbf{n}, h\}$ and relative camera pose $(\mathbf{R}, \mathbf{t})$ between frames k and $k+1$, a per-pixel mapping can be computed within a region of interest (ROI) by using the homography matrix $\mathbf{G} = \mathbf{R} + h^{-1}\mathbf{t}\mathbf{n}^\top$.

$\{h, \mathbf{n}\}$ and relative camera pose $\{\mathbf{R}, \mathbf{t}\}$ between frames k and $k+1$ (For KITTI's 10Hz input rate, there is often little overlap of ROI between frames k and k+2. Conversely, a baseline between k and k+1 is sufficient. For other data with 30Hz imagery, we adapt the baseline accordingly.), a per-pixel mapping can be computed using the homography matrix

$$\mathbf{G} = \mathbf{R} + \frac{1}{h}\mathbf{t}\mathbf{n}^\top. \tag{5.6}$$

The homography mapping is illustrated in Figure 5.3. Note that $\mathbf{t}$ differs from the true translation $\mathbf{t}^*$ by an unknown scale drift factor, encoded in the h we wish to estimate. Pixels in frame $k+1$ are mapped to frame k (subpixel accuracy is important for good performance) and the sum of absolute differences (SAD) is computed over bi-linearly interpolated image intensities. With $\rho = 1.5$, a Nelder-Mead simplex routine is used to estimate the $\{h, \mathbf{n}\}$ that minimize:

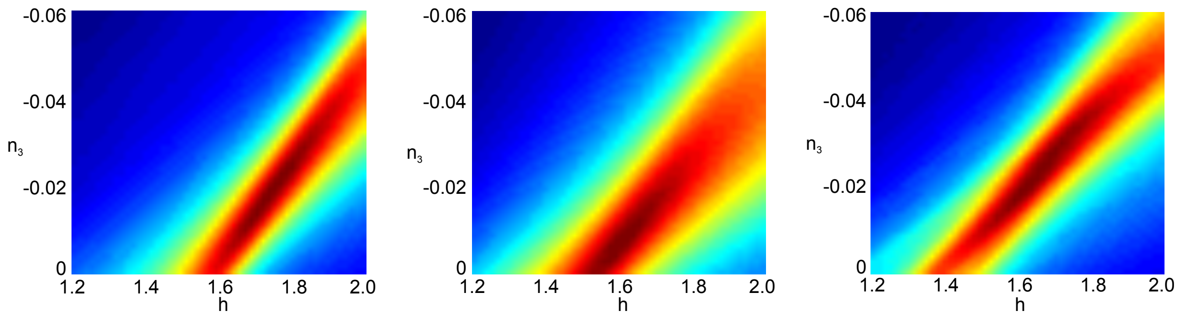


Figure 5.4: The cost volumes for the dense stereo cue. 2D slices of the $\rho^{-\mathrm{SAD}}$ cost volumes along h and n_3 at frames 490, 810 and 940 of Sequence 08 in the KITTI dataset. In most cases, clear local extrema are obtained.

$$\min_{h,\mathbf{n}} \left(1-\rho^{-\mathrm{SAD}}\right). \tag{5.7}$$

Note that the optimization only involves h, n_1 and n_3, since $\|\mathbf{n}\| = 1$. Enforcing the norm constraint has marginal effect, since the calibration pitch is a good initialization and the cost function usually has a clear local minimum in that vicinity, as shown for a few examples in Fig. 5.4. The optimization requires about 10 ms per frame. The $\{h, \mathbf{n}\}$ that minimizes (5.7) is the estimated ground plane from stereo cues.

5.2.2 Triangulated 3D Points

Next, we consider matched sparse SIFT [Low04] descriptors between frames k and $k+1$, computed within the above region of interest (we find SIFT a better choice than ORB for the low-textured road, and real-time performance is attainable for SIFT in the small ROI). To fit a plane through the triangulated 3D points, one option is to estimate $\{h, \mathbf{n}\}$ using a 3-point RANSAC for plane-fitting. However, in our experiments, better results are obtained using the method of [GZS11], by assuming the camera pitch to be fixed from calibration. For every triangulated 3D point, the height h is computed using (5.1). The height difference Δh_{ij} is computed for every 3D point i with respect to every other point j. The estimated ground plane height is the height of the point i corresponding to the maximal score q, where

$$q = \max_i \left\{ \sum_{j \neq i} \exp\left(-\mu \Delta h_{ij}^2\right) \right\}, \text{ with } \mu = 50. \tag{5.8}$$

Note: Prior works such as [SS08,SCG13] decompose the homography $\mathbf{G}$ between frames to yield the camera height [FL88]. However, in practice, the decomposition is very sensitive to noise, which is a severe problem since the homography is computed using noisy feature matches from the low-textured road. We also note that the homography decomposition cannot be expected to perform better than the 3D points cue, since they both rely on the same set of feature matches. Further, the fact that road regions may be mapped by a homography is already exploited by our plane-guided dense stereo.

5.2.3 Object Detection Cues

We can also use object detection bounding boxes as cues when they are available, for instance, within the object localization application. The ground plane pitch angle θ can be estimated from this cue. Recall that $n_3 = \sin\theta$, for the ground normal $\mathbf{n} = (n_1, n_2, n_3)^\top$.

From (7.2), given the 2D bounding box, we can compute the 3D height h_b of an object through the ground plane. Given a prior height $\bar{h}_b$ of the object, n_3 is obtained by solving:

$$\min_{n_3} (h_b - \bar{h}_b)^2. \tag{5.9}$$

The ground height h used in (7.2) is set to the calibration value to avoid incorporating SFM scale drift and n_1 is set to 0 since it has negligible effect on object height.

Note: Object bounding box cues provide us unique long distance information, unlike dense stereo and 3D points cues that only focus on a ROI close to our vehicle. An inaccurate pitch angle can lead to large vertical errors for distant objects. Thus, the 3D localization accuracy of far objects is significantly improved by incorporating this cue, as shown in Sec. 7.6.1.

5.3 Data-Driven Cue Combination

We now propose a principled approach to combine the above cues while reflecting the per-frame relative accuracy of each. The cues provided by the above methods are combined in a Kalman filter framework significantly different from prior works. Naturally, the combination should be influenced by both the visual input at a particular frame and prior knowledge. We achieve this by learning models from training data to relate the observation covariance for each cue to error behaviors of its underlying variables. During testing, our learned models adapt each cue's observation covariance on a per-frame basis. The error distributions at every frame adapt the observation covariances using those learned models, on a per-frame basis.

5.3.1 Training

For the dense stereo and 3D points cues, we use the KITTI visual odometry dataset for training, consisting of $F = 23201$ frames. Sequences 0 to 8 of the KITTI tracking dataset are used to train the object detection cue. To determine the ground truth h and $\mathbf{n}$, we label regions of the image close to the camera that are road and fit a plane to the associated 3D points from the provided Velodyne data. No labelled road regions are available or used during testing.

Each method i described in Sec. 5.2 has a scoring function f_i that can be evaluated for various positions of the ground plane variables $\boldsymbol{\pi} = \{h, \mathbf{n}\}$. The functions f_i for stereo, 3D points and object cues are given by (5.7), (5.8) and (5.9), respectively.

Then, Algorithm 1 is a general description of the training.

Intuitively, the parameters $\mathbf{a}_i^k$ of model $\mathscr{A}_i^k$ reflect belief in the effectiveness of cue i at frame k. Quantizing the parameters $\mathbf{a}_i^k$ from F training frames into L bins allows estimating the variance of observation error at bin centers $\mathbf{c}_i^l$. The model $\mathscr{C}_i$ then relates these variances, v_i^l, to the cue's accuracy (represented by quantized parameters $\mathbf{c}_i^l$). Thus, at test time, for every frame, we can estimate the accuracy of each cue i based purely on visual data (that is, by computing $\mathbf{a}_i$) and use the model $\mathscr{C}_i$ to determine its observation variance.

Now we describe the specifics for training the models $\mathscr{A}$ and $\mathscr{C}$ for each of the

Algorithm 1 Data-Driven Training for Cue Combination

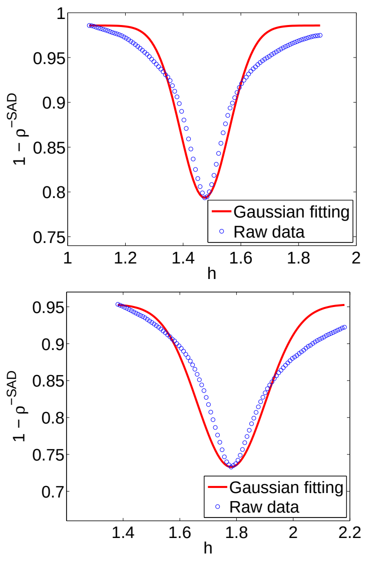
```
for Training frames $k = 1 : F$ do
    • For various values of $\boldsymbol{\pi} = \{h, \mathbf{n}\}$, fit a model $\mathscr{A}_i^k$ to observations $(\boldsymbol{\pi}, f_i(\boldsymbol{\pi}))$.
    Parameters $\mathbf{a}_i^k$ of model $\mathscr{A}_i^k$ reflect belief in accuracy of cue $i$ at frame $k$. (For
    instance, when $\mathscr{A}$ is a Gaussian, $\mathbf{a}$ can be its variance.)
    • Compute error $e_i^k = |\arg\min_{\boldsymbol{\pi}} f_i(\boldsymbol{\pi}) - \boldsymbol{\pi}^{*k}|$, where the ground truth ground plane
    in frame $k$ is $\boldsymbol{\pi}^{*k}$.
end for
• Quantize model parameters $\mathbf{a}_i^k$, for $k = 1, \cdots, F$, into $L$ bins centered at $\mathbf{c}_i^1, \cdots, \mathbf{c}_i^L$.
• Histogram the errors $e_i^k$ according to quantized $\mathbf{c}_i^l$. Let $v_i^l$ be the bin variances of $e_i^k$,
for $l = 1, \cdots, L$.
• Fit a model $\mathscr{C}_i$ to observations $(\mathbf{c}_i^l, v_i^l)$.
```

dense stereo, 3D points and object cues. We will use the notation that $i \in \{s, p, d\}$, denote the dense stereo, 3D points and object detection methods, respectively.

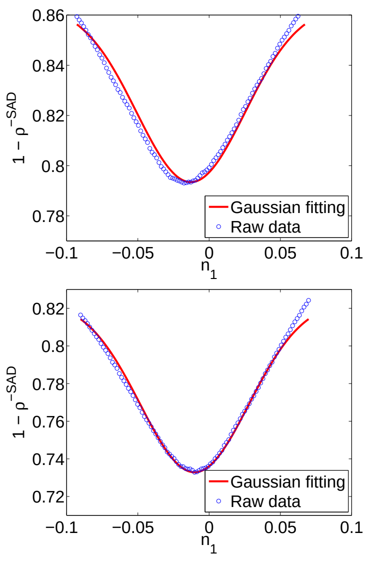
Dense Stereo

The objective of training is to find a model from extensive training data to relate the observation covariance for each cue to error behavior of its underlying variables. The error behavior of dense stereo between two consecutive frames is characterized by variation in SAD scores between road regions related by the homography (5.6), as we independently vary each variable h, n_1 and n_3. The variance of this distribution of SAD scores represents the error behavior of the stereo cue with respect to its variables. Recall that the scoring function for stereo, f_s, is given by (5.7). Then a model is trained to relate the observation covariance of the stereo method, σ_s, to σ_s'. In the following paragraph, we explain these steps in detail. We assume that state variables are uncorrelated. Thus, we will learn three independent models corresponding to h, n_1, and n_3.

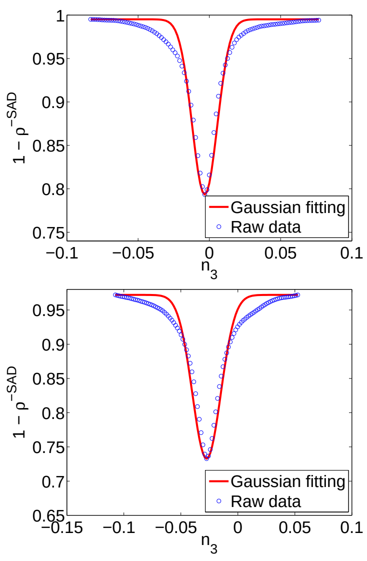
Learning the model $\mathscr{A}_s$: For a training image k, let $\{\widehat{h}^k, \widehat{\mathbf{n}}^k\}$ be the ground plane estimated by the dense stereo method, by optimizing f_s in (5.7). We first fix $n_1 = \widehat{n}_1^k$ and $n_3 = \widehat{n}_3^k$ and for 50 uniform samples of h in the range $[0.5\widehat{h}^k, 1.5\widehat{h}^k]$, construct homography mappings from frame k to $k+1$, according to (5.6) (note that $\mathbf{R}$ and $\mathbf{t}$ are



(a) h



(b) n_1



(c) n_3

Figure 5.5: Examples of 1D Gaussian fits to estimate parameters $\mathbf{a}_s^k$ for h, n_1 and n_3 of the dense stereo method respectively.

already estimated by monocular SFM, up to scale). For each homography mapping, we compute the SAD score $f_s(h)$ using (5.7). A univariate Gaussian is now fit to the distribution of $f_s(h)$. Its variance, $a_{s,h}^k$, captures the sharpness of the SAD distribution, which reflects belief in accuracy of height h estimated from the dense stereo method at frame k. The k in $\sigma_{s,h}^{\prime k}$ stands for frame k, the s indicates stereo (between frame k and frame $k+1$) and h means the curve is fitted between the value $1-\rho^{-SAD}$ and h. A similar procedure yields variances a_{s,n_1}^k and a_{s,n_3}^k corresponding to orientation variables. Example fits are shown in Fig. 5.5. Referring to Algorithm 1 above, $a_{s,h}^k$, a_{s,n_1}^k, a_{s,n_3}^k are precisely the parameters $\mathbf{a}_s^k$ that indicate accuracy of the stereo cue at frame k.

Learning the model $\mathscr{C}_s$: For frame k, let $e_{s,h}^k = |\widehat{h}^k - h_k^*|$ be the error in ground height, relative to ground truth. We also have a parameter $a_{s,h}^k$ at frame k. We quantize the parameters $a_{s,h}^k$ into $L = 100$ bins and consider the resulting histogram of $e_{s,h}^k$. The bin centers $c_{s,h}^l$ are positioned to match the density of $a_{s,h}^k$ (that is, we distribute F/L errors $e_{s,h}^k$ within each bin). A similar process is repeated for n_1 and n_3. The histograms for the KITTI dataset are shown in Fig. 5.6. We have now obtained the $\mathbf{c}_s^l$ of Algorithm 1.

Next, we compute the variance $v_{s,h}^l$ of the errors within each bin l, for $l = 1, \cdots, L$.

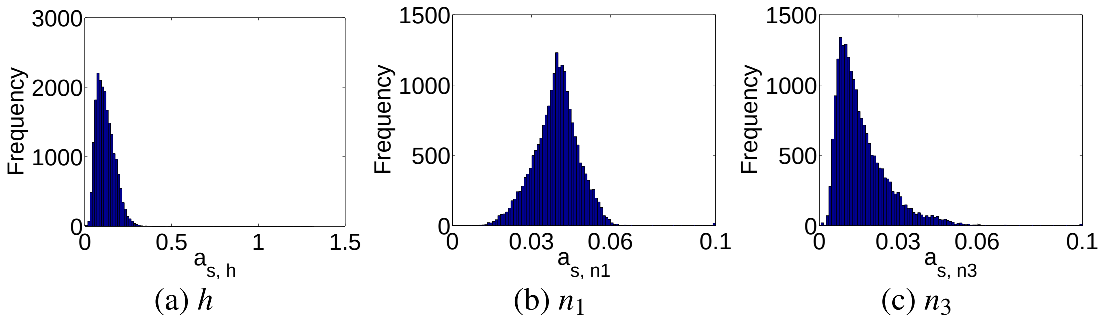


Figure 5.6: Histograms of errors e_s^k from dense stereo cue against the quantized accuracy parameters $\mathbf{a}_s$ of model $\mathscr{A}_s$, for h, n_1 and n_3.

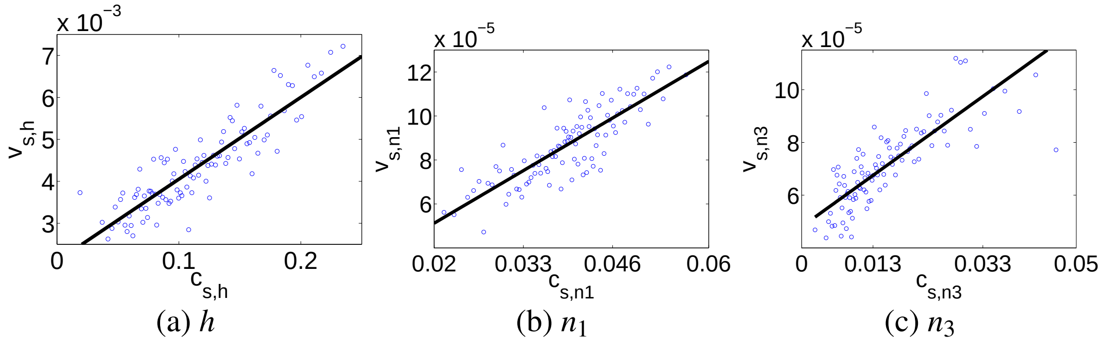


Figure 5.7: Fitting a model $\mathscr{C}_s$ to relate observation variance v_s to the belief in accuracy $\mathbf{c}_s$ of dense stereo, for h, n_1 and n_3.

This indicates the observation error variance. We now fit a curve to the distribution of $v_{s,h}$ versus $c_{s,h}$, which provides a model to relate observation variance in h to the effectiveness of dense stereo. The result for the KITTI dataset is shown in Fig. 5.7, where each data point represents a pair of observation error covariance $v_{s,h}^l$ and parameter $c_{s,h}^l$. Empirically, we observe that a straight line suffices to produce a good fit. A similar process is repeated for n_1 and n_3. Thus, we have obtained models $\mathscr{C}_s$ (one each for h, n_1 and n_3) for the stereo method.

3D Points

Similar to dense stereo, the objective of training is again to find a model that relates the observation covariance of the 3D points method to the error behavior of its

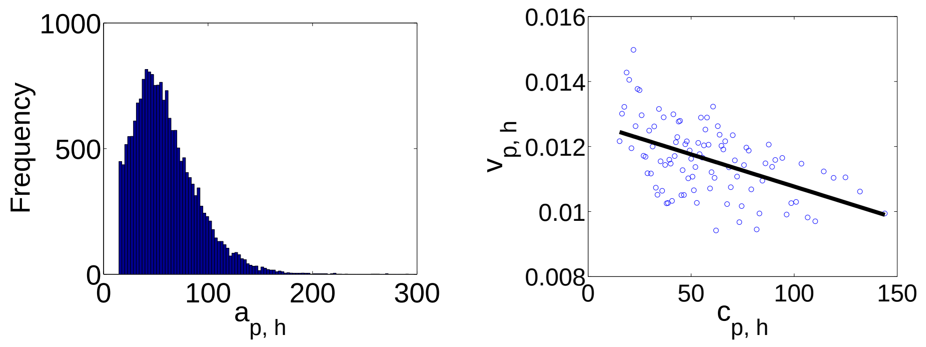


Figure 5.8: Histogram of height error of the 3D points cue. (Left) Histogram of height error versus plane fitting cost $q_{p,h}$ in (5.8). (Right) Line fit to the distribution of error variance $\sigma_{p,h}$ against cost $q_{p,h}$, for the entire KITTI training set.

underlying variables. Recall that the scoring function f_p is given by (5.8).

Learning the model $\mathscr{A}_p$: We observe that the score q returned by f_p is directly an indicator of belief in accuracy of the ground plane estimated using the 3D points cue. Thus, for Algorithm 1, we may directly obtain the parameters $a_p^k = q^k$, where q^k is the optimal value of f_p at frame k, without explicitly learning a model $\mathscr{A}_p$.

Learning the model $\mathscr{C}_p$: The remaining procedure mirrors that for the stereo cue. Let $\widehat{h}_p^k$ be ground height estimated at frame k using 3D points, that is, the optimum for (5.8). The error $e_{p,h}^k$ is computed with respect to ground truth. The above $a_{p,h}^k$ are quantized into $L = 100$ bins centered at $c_{p,h}^l$ and a histogram of observation errors $e_{p,h}^k$ is constructed. A model $\mathscr{C}_p$ may now be fit to relate the observation variances $v_{p,h}^l$ at each bin to the corresponding accuracy parameter $c_{p,h}^l$. As shown in Fig. 5.8, a straight line fit is again reasonable.

Object Detection

We assume that the detector provides several candidate bounding boxes and their respective scores (that is, bounding boxes before the nonmaximal suppression step of traditional detectors). A bounding box is represented by $\mathbf{b} = (x, y, w, h_b)^\top$, where x, y is its 2D position and w, h_b are its width and height. The error behavior of detection is quantified by the variation of detection scores α with respect to bounding box $\mathbf{b}$.

Learning the model $\mathscr{A}_d$: Our model $\mathscr{A}_d^k$ is a mixture of Gaussians. At each

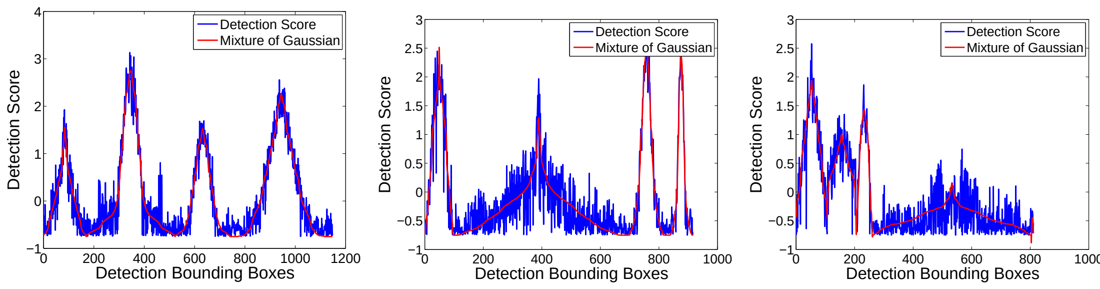


Figure 5.9: Examples of mixture of Gaussians fits to detection scores. Note that our fitting (red) closely reflects the variation in noisy detection scores (blue). Each peak corresponds to an object.

frame, we estimate 4×4 full rank covariance matrices Σ_m centered at $\boldsymbol{\mu}_m$, as:

$$\min_{A_m,\mu_m,\Sigma_m} \sum_{n=1}^{N} \left(\sum_{m=1}^{M} A_m e^{-\frac{1}{2}\epsilon_{mn}\Sigma_m^{-1}\epsilon_{mn}} - \alpha_n \right)^2, \tag{5.10}$$

where $\epsilon_{mn} = \mathbf{b}_n - \boldsymbol{\mu}_m$, M is number of objects and N is the number of candidate bounding boxes (the dependence on k has been suppressed for convenience). Due to highly irregular data, the traditional EM method doesn't work. Instead, we seek to solve this problem as a non-linear optimization problem. Example fitting results are shown Fig. 5.9. It is evident that the variation of noisy detector scores is well-captured by the model $\mathscr{A}_d^k$.

Recall that the scoring function f_d of (5.9) estimates n_3. Thus, only the entries of Σ_m corresponding to y and h_b are significant for our application. Let σ_y and σ_{h_b} be the corresponding diagonal entries of the Σ_m closest to the tracking 2D box. We combine them into a single parameter, $a_d^k = \frac{\sigma_y \sigma_{h_b}}{\sigma_y + \sigma_{h_b}}$, which reflects our belief in the accuracy of this cue.

Learning the model $\mathscr{C}_d$: The remaining procedure is similar to that for the stereo and 3D points cues. The accuracy parameters a_d^k are quantized and related to the corresponding variances of observation errors, given by the f_d of (5.9). The fitted linear model $\mathscr{C}_d$ that relates observation variance of the detection cue to its expected accuracy is shown in Fig. 5.10.

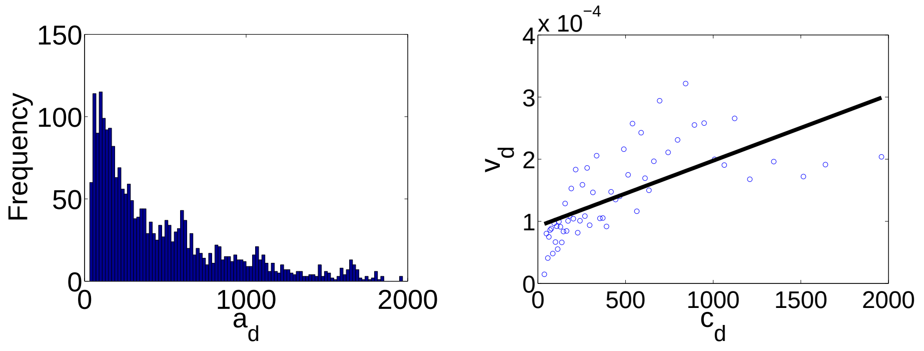


Figure 5.10: Histogram of height error of the 3D points cue. (Left) Histogram of height error versus plane fitting cost $q_{p,h}$ in (5.8). (Right) Line fit to the distribution of error variance $\sigma_{p,h}$ against cost $q_{p,h}$, for the entire KITTI training set.

5.3.2 Testing

During testing, at every frame k, we fit a model A_i^k corresponding to each cue $i \in \{s, p, d\}$ and determine its parameters $\mathbf{a}_i^k$ that convey expected accuracy. Next, we use the models $\mathscr{C}_i$ to determine the observation variances corresponding to the expected accuracy.

Dense Stereo The observation $\mathbf{z}_s^k = (n_1^k, n_3^k, h^k)^\top$ at frame k is obtained by minimizing f_s, given by (5.7). We fit 1D Gaussians to the homography-mapped SAD scores to get the values of $a_{s,h}^k$, a_{s,n_1}^k and a_{s,n_3}^k. Using the models $\mathscr{C}_s$ estimated in Fig. 5.7, we predict the corresponding variances v_s^k. The observation covariance for the dense stereo method is now available as $\mathbf{U}_1^k = \mathrm{diag}(v_{s,n_1}^k, v_{s,n_3}^k, v_{s,h}^k)$.

3D Points At frame k, the observation $\mathbf{z}_p^k$ is the estimated ground height h obtained from f_p, given by (5.8). The value of q^k obtained from (5.8) directly gives us the expected accuracy parameter a_p^k. The corresponding variance $v_{p,h}^k$ is estimated from the model $\mathscr{C}_p$ of Fig. 5.8. The observation covariance for this cue is now available as $\mathbf{U}_p^k = v_{p,h}^k$.

Object Detection At frame k, the observation $\mathbf{z}_d^{k,m}$ is the ground pitch angle n_3 obtained by minimizing f_d, given by (5.9), for each object $m = 1, \cdots, M$. For each object m, we obtain the parameters $a_d^{k,m}$ after solving (5.10). Using the model $\mathscr{C}_d$ of Fig. 5.10, we predict the corresponding error variances $v_d^{k,m}$. The observation covariances for this method are now given by $\mathbf{U}_d^{k,m} = v_d^{k,m}$.

Fusion Finally, the adaptive covariance for frame k, $\mathbf{U}^k$, is computed by combining $\mathbf{U}_s^k$, $\mathbf{U}_p^k$ and the $\mathbf{U}_d^{k,m}$ from each object m. Then, our adaptive ground plane estimate $\mathbf{z}^k$ is computed by combining $\mathbf{z}_s^k$, $\mathbf{z}_p^k$ and $\mathbf{z}_d^{k,m}$, using (5.5).

In [SC14], we used a cross validation mechanism between dense stereo cue and 3D points cue to reject outliers. That is, we only used the fused height estimate to update the Kalman filter when both the cues are available and the estimates of them are close to each other. However we found that the 3D points cue is much worse than the dense stereo cue. Eliminating good estimates from dense cue just because the 3D points cue is bad seems a waste. In this work, we allow the system to use dense stereo cue solely when we believe it is well converged.

Thus, we have described a ground plane estimation method that uses models learned from training data to adapt the relative importance of each cue – stereo, 3D points and detection bounding boxes – on a per-frame basis.

The entire procedure is summarized in Figure 5.11, which specifically lists the cues, models and learned observation covariances illustrated in Figure 5.1.

This chapter is based on “Robust Scale Estimation in Real-Time Monocular SFM for Autonomous Driving”, by Shiyu Song, Manmohan Chandraker, as it appears in proceedings of Computer Vision and Pattern Recognition (CVPR), 2014 IEEE International Conference on, June 24-27 2014, Columbus, Ohio.

This chapter in part, has been submitted for publication, as it may appear in “High Accuracy Monocular SFM and Scale Correction for Autonomous Driving”, by Shiyu Song, Manmohan Chandraker, Clark C. Guest, in IEEE Transactions on Pattern Analysis and Machine Intelligence.

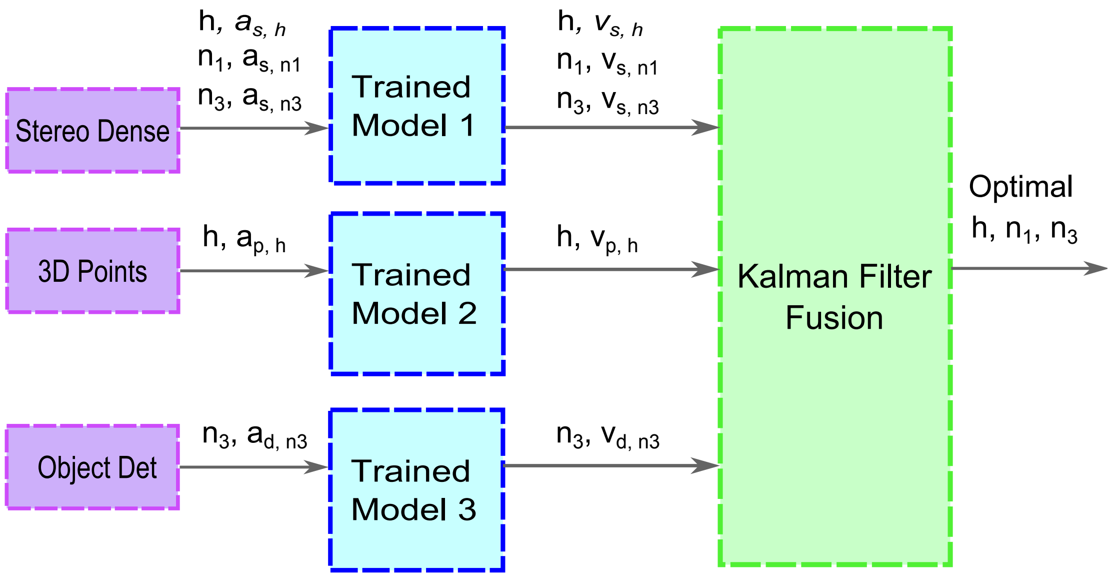


Figure 5.11: Adaptive cue combination framework with specific details. For each frame, $(h, a_{s,h})$ (n_1, a_{s,n_1}) and (n_3, a_{s,n_1}) are estimated for the dense stereo method, $(h, a_{p,h})$ for the 3D points method and $(n_3, a_{d,h})$ for the objection detection method. In each tuple, the first element is one of the variables influencing the estimation, while the other is a quantitative representation of the error behavior due to that variable. The trained models in Figures 5.7, 5.8 and 5.10 are used to get the observation covariances $v_{s,h}$, v_{s,n_1}, v_{s,n_3}, $v_{p,h}$ and v_{d,n_3} from $a_{s,h}$, a_{s,n_1}, a_{s,n_1}, $a_{p,h}$ and a_{d,n_3}. Finally, a Kalman filter fuses the observations and their covariances to yield optimal estimations.

Chapter 6

Results of Monocular Visual Odometry

We present extensive evaluation on the state-of-art KITTI dataset [GLU12], which consists of nearly 50 km of real-world driving in 22 sequences, covering urban, residential, country and highway roads. Speeds varying from 0 to 90 kmph, a low frame rate of 10 Hz and frequent presence of other cars pose additional challenges. We also show results on the publicly available Hague dataset [DG09], which is challenging due to low resolution and multiple objects.

The evaluation metrics are provided by Geiger et al. [GLU12], based on an extension of those proposed by Kümmerle et al. in [KSD$^+$09]. Rotation and translation errors are reported as averages of all relative transformations at a fixed distance, as well as functions of various subsequence lengths and vehicle speeds. For timings, our experiments are performed on a laptop with Intel Core i7 2.40 GHz processor with 8GB DDR3 RAM and 6M cache. The main modules occupy three parallel threads as depicted in Sections 4.1–4.2, while ground detection for scale correction occupies two threads of its own.

In consideration of real-time performance, only the dense stereo and 3D points cues are used for monocular SFM. Detection bounding box cues are used for the object localization application where they are available. Note that, when detection bounding box cues are available, it only improves ground orientation, it can not harm SFM. 3D object localization is demonstrated using object detection and tracked bounding boxes computed offline using [PRF11].

6.1 Benchmark Monocular Visual Odometry on KITTI

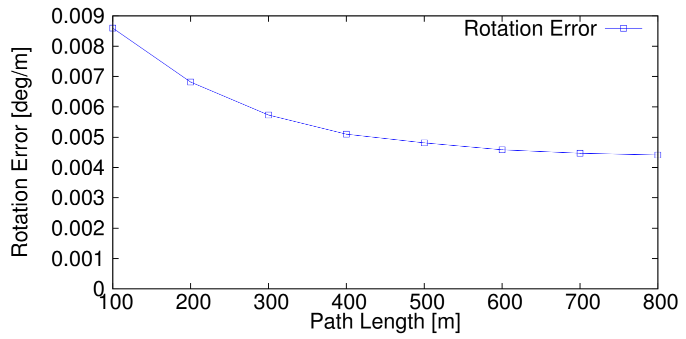
The visual odometry evaluation sequences in KITTI are 11–21, for which ground truth is not public. Our system's performance for these sequences is accessible from the KITTI evaluation webpage [1], under the name **MLM-SFM**. Figure 6.1 shows the performance of our system, with average rotation and translation errors reported over various subsequence lengths from 5 to 400 meters and speeds from 5 to 90 kmph. Note that all the other systems, with the exception of VISO2-M [GZS11], are stereo, yet we achieve close to the best rotation accuracy. Our translation errors are lower than several stereo systems and far lower than VISO2-M.

6.2 Accuracy and Robustness of Monocular SFM

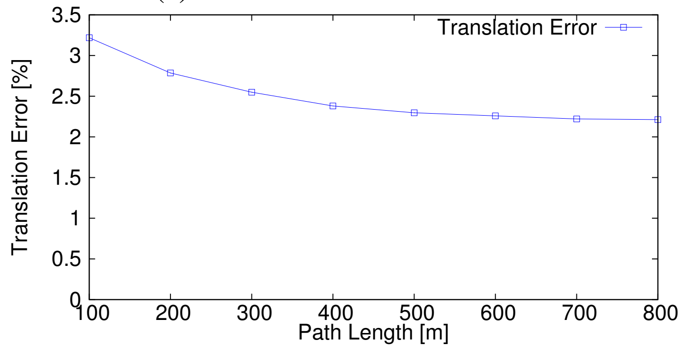
Another important benefit of our ground plane estimation is enhanced robustness. As demonstration, we run 50 trials of our system on Seq $0 - 10$, as well as stereo and monocular systems associated with the dataset, VISO2-S and VISO2-M [GZS11]. Errors relative to ground truth are computed using the metrics in [GLU12]. Average errors over Seq 0–10 are shown in Table 6.1. Note our vast performance improvement over VISO2-M, a rotation error better than VISO2-S and a low translation error also slightly better than stereo. The only sequences where we encounter high errors are 1 and 7. The former is an extended highway sequence at speeds of 90 kmph with repeated textures, while the latter has a segment where a large truck occludes over 70% of the image (see Figure 6.2). Our monocular SFM system currently relies only on low-level features, however, our future work integrates lane and object detection which can allow handling such scenarios.

In Figure 6.3 and 6.4, we show the reconstructed trajectories from our monocular SFM with adaptive ground plane estimation, the monocular system of VISO2-M that only uses 3D points and the stereo system VISO2-S [GZS11], for eight other sequences from the KITTI dataset, besides the two shown in Figure 1.5. All the trajectories are shown in blue, compared to ground truth shown in red. Note the high accuracy of our monocular system relative to ground truth, comparable to a stereo system and far more

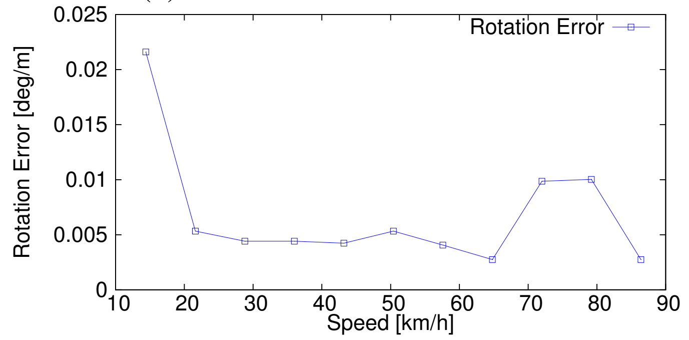
[1]`www.cvlibs.net/datasets/kitti/eval_odometry.php`



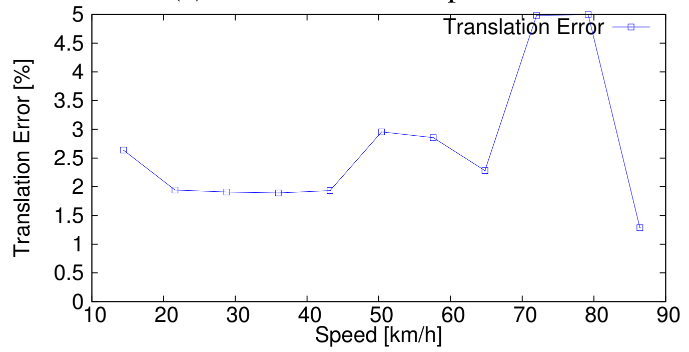
(a) Rot. error across distances



(b) Trans. error across distances



(c) Rot. error across speeds



(d) Trans. error across speeds

Figure 6.1: Evaluation results on the KITTI benchmark, for rotation and translation errors over various distances and speeds.

Table 6.1: Comparison of rotation and translation errors for our system versus other state-of-the-art stereo and monocular systems. The values reported are statistics over 50 trials and demonstrate the robustness of our system. Note that our translation and rotation errors are lower than stereo VISO2-S, and much better than VISO2-M.

Seq	Frms	Rot (deg/m)	σ_R^2	Trans (%)	σ_T^2
		VISO2-S (Stereo)			
0	4540	0.0109	7.06E-08	2.32	0.00323
2	4660	0.00736	3.27E-08	2.01	0.00174
3	800	0.0107	2.3E-07	2.32	0.0125
4	270	0.00807	8.75E-07	0.99	0.00295
5	2760	0.00976	3.77E-08	1.78	0.0019
6	1100	0.00716	1.56E-07	1.17	0.00466
8	4070	0.0104	6.58E-08	2.35	0.00375
9	1590	0.00938	1.56E-07	2.36	0.00719
10	1200	0.00857	4.35E-07	1.37	0.0112
Avg		0.009379		2.057332	
		VISO2-M (Monocular)			
Seq	Frms	Rot (deg/m)	σ_R^2	Trans (%)	σ_T^2
0	4540	0.0209	3.03E-06	11.91	0.0888
2	4660	0.0114	3.16E-07	3.33	0.0155
3	800	0.0197	6.79E-06	10.66	0.515
4	270	0.00927	3.08E-06	7.4	0.00957
5	2760	0.0328	3.2E-06	12.67	0.106
6	1100	0.0157	1.26E-06	4.74	0.0998
8	4070	0.0203	1.05E-06	13.94	0.102
9	1590	0.0143	2.29E-06	4.04	0.0842
10	1200	0.0388	1.33E-05	25.2	3.22
Avg		0.020295		10.18093	
		Our Results (Monocular)			
Seq	Frms	Rot (deg/m)	σ_R^2	Trans (%)	σ_T^2
0	4540	0.00476	1.08E-06	2.04	0.114
2	4660	0.00354	5.68E-08	1.5	0.0135
3	800	0.00213	1.05E-07	3.37	0.285
4	270	0.00234	4.93E-07	1.43	0.372
5	2760	0.00378	6.56E-06	2.19	0.195
6	1100	0.00813	1.85E-05	2.09	0.687
8	4070	0.00442	3.81E-07	2.37	0.0534
9	1590	0.00466	5.78E-07	1.76	0.0326
10	1200	0.00848	0.000102	2.12	1.27
Avg		0.004545		2.032654	

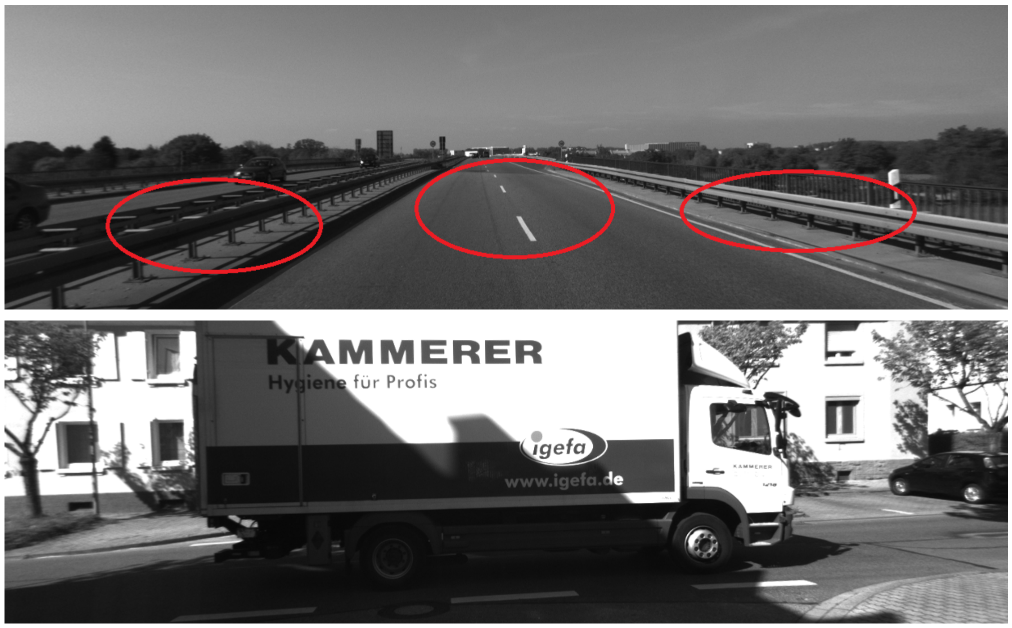


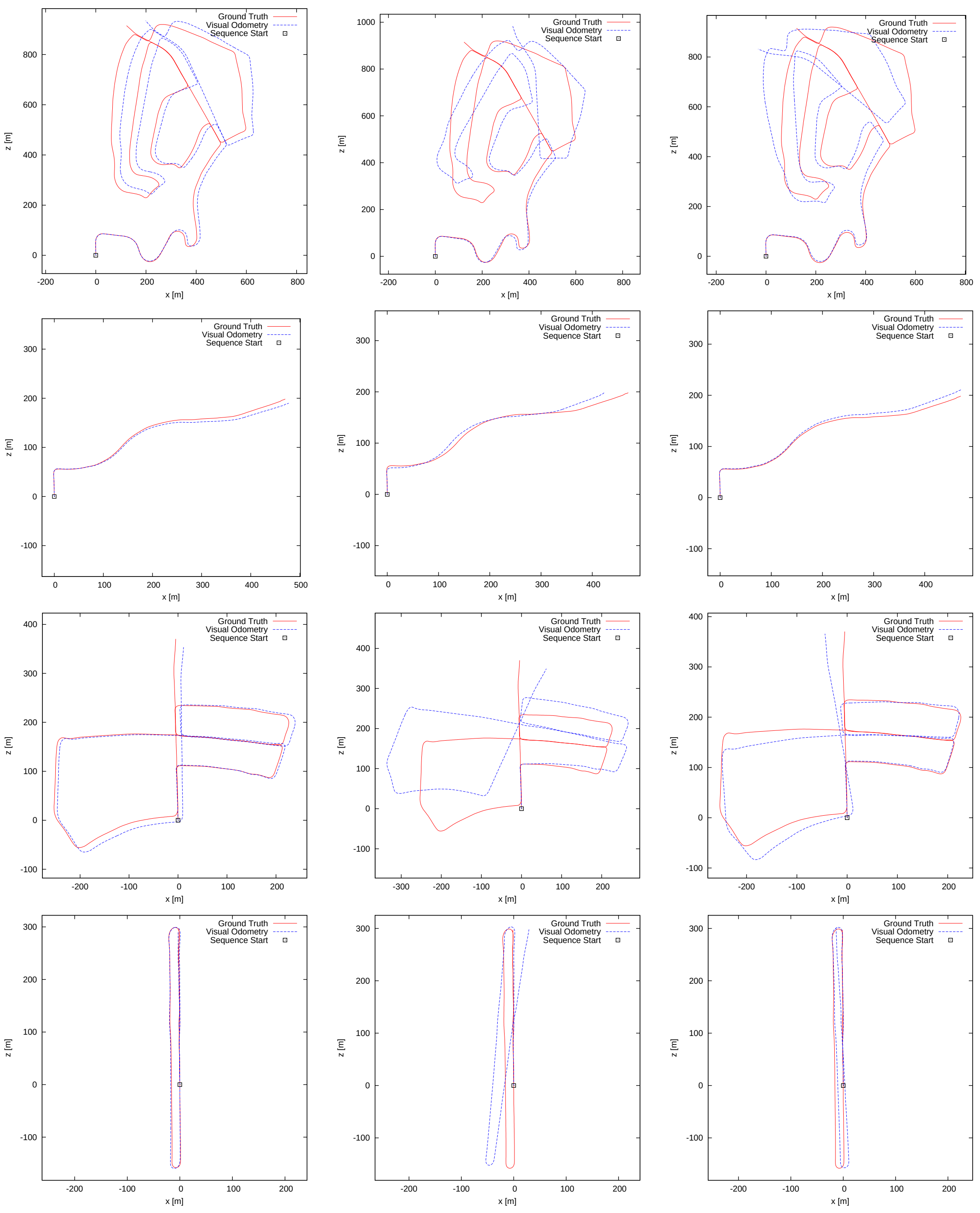
Figure 6.2: Example of failure scenarios. Example frames from Seq 01 and 07 with repeated features and serious interference from obstacles. These are situations that our system, which relies purely on SFM, is not designed to handle.

accurate than the prior monocular works. Also notice that our rotation error is lower than stereo, which has significant impact on long-range location error. This performance is enabled by our system architectural innovations and ground plane estimation, which combines multiple cues and adapts their relative weights to reflect per-frame uncertainties in visual data using models learned from training data.

6.3 Accuracy of Ground Plane Estimation

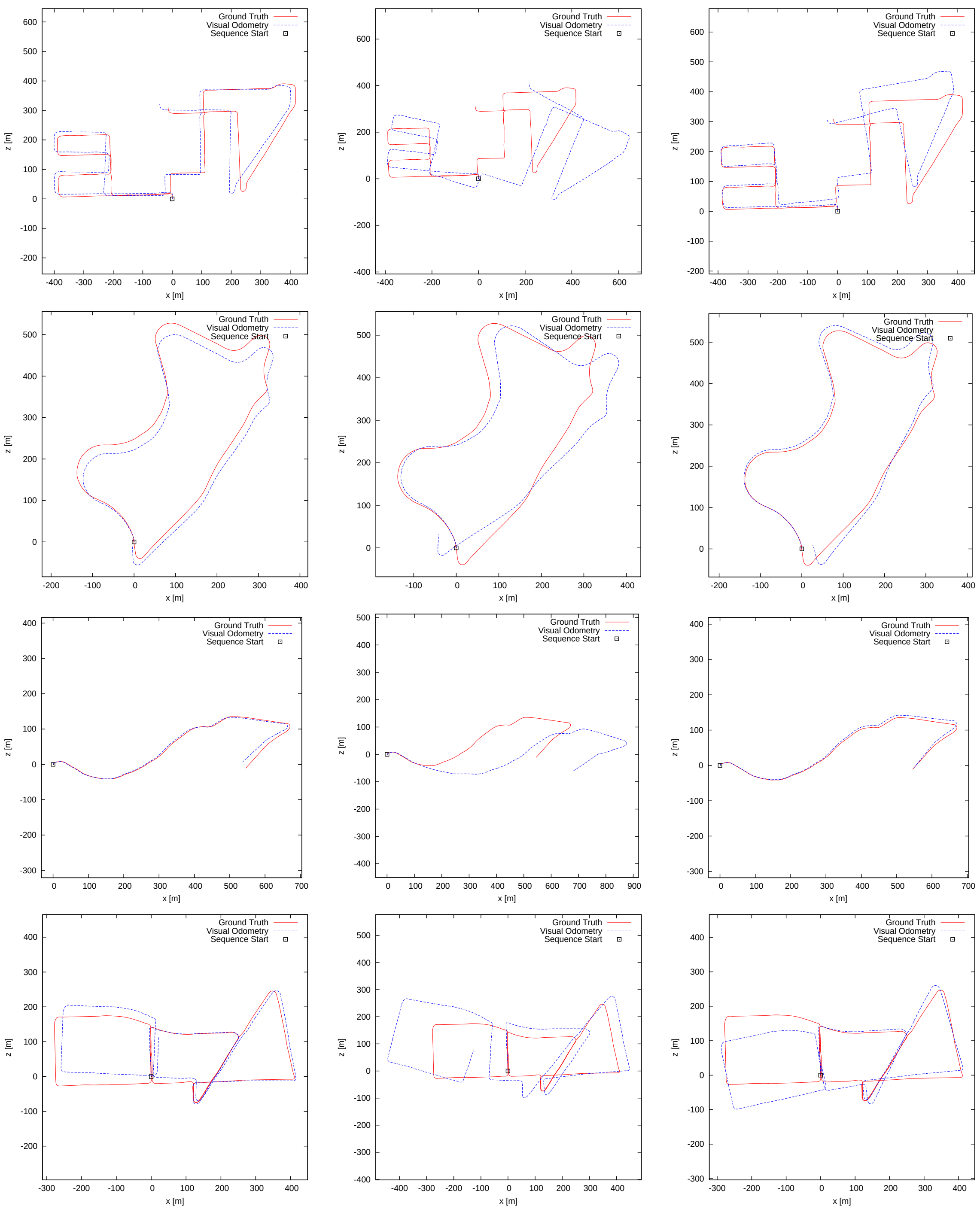
The performance of our monocular SFM is significantly better than [SCG13], since we successfully complete KITTI test sequences 11–21. Ground plane estimation that combines cues in a rigorous Kalman filter and adaptively computes fusion covariances is key to achieving our robust performance.

Fig. 6.5 shows examples of error in ground plane height relative to ground truth using 3D points and stereo cues individually, as well as the output of our combination. Each cue and the combination are implemented within a Kalman filter. Note that while



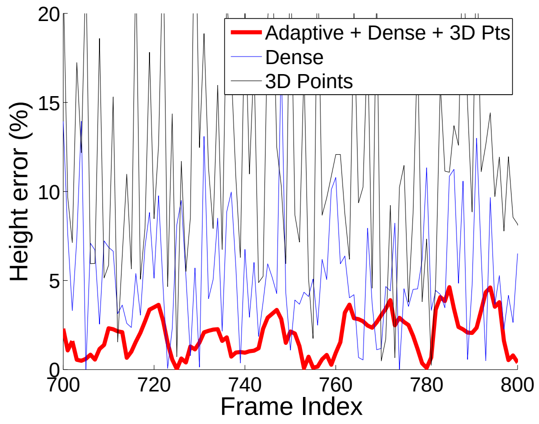
(a) Our System (b) VISO2-Mono [GZS11] (c) VISO2-Stereo [GZS11]

Figure 6.3: Reconstructed trajectories from sequences in the KITTI training dataset (with ground truth). Also see trajectories in Figure 1.5. (a) Our monocular SFM yields camera trajectories close to the ground truth over several kilometers of real-world driving. (b) Our monocular SFM significantly outperforms prior works that also use the ground plane for scale correction. (c) Our performance is comparable to stereo SFM.



(a) Our System (b) VISO2-Mono [GZS11] (c) VISO2-Stereo [GZS11]

Figure 6.4: Reconstructed trajectories from sequences in the KITTI training dataset (with ground truth). Also see trajectories in Figure 1.5. (a) Our monocular SFM yields camera trajectories are close to the ground truth over several kilometers of real-world driving. (b) Our monocular SFM significantly outperforms prior works that also use the ground plane for scale correction. (c) Our performance is comparable to stereo SFM.



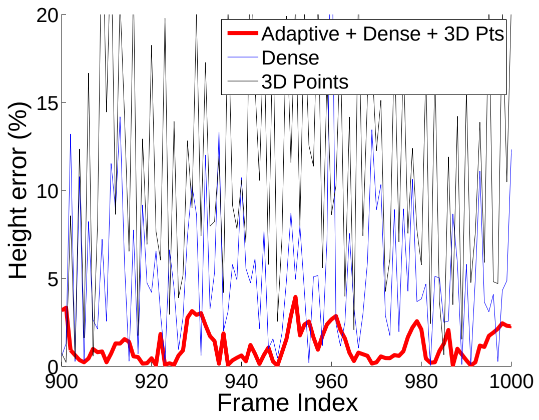


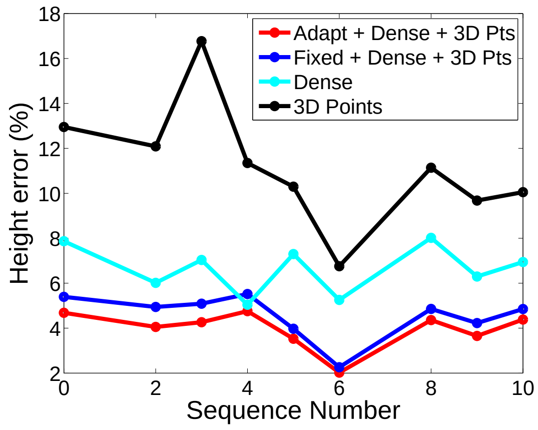
Figure 6.5: Height error relative to ground truth over (left) Seq 2 and (right) Seq 5. The effectiveness of our data fusion is shown by less spikiness in the filter output and a far lower error.

individual methods are very noisy, our cue combination allows a much more accurate estimation than either.

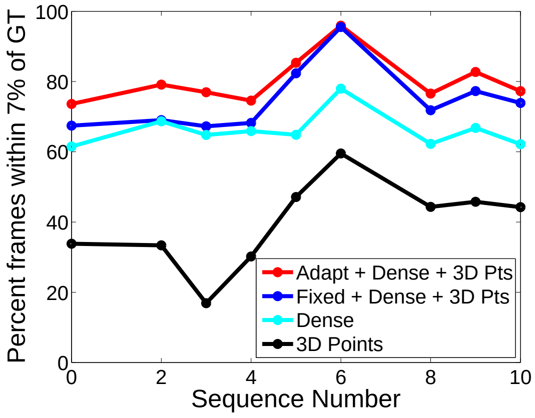
Next, we demonstrate the advantage of cue combination using the data-driven framework of Sec. 5.3 that uses adaptive covariances, as opposed to a traditional Kalman filter with fixed covariances. For this experiment, the fixed covariance for the Kalman filter is determined by the error variances of each variable over the entire training set (we verify by cross-validation that this is a good choice).

In Fig. 6.6, using only sparse feature matches causes clearly poor performance (black curve). The dense stereo performs better (cyan curve). Including the additional dense stereo cue within a Kalman filter with fixed covariances leads to an improvement (blue curve). However, using the training mechanism of Sec. 5.3 to adjust per-frame observation covariances in accordance with the relative confidence of each cue leads to a further reduction in error by nearly 1% (red curve). Fig. 6.6(b) shows that we achieve the correct scale at a rate of 75 – 100% across all sequences, far higher than the other methods.

In particular, compare our output (red curves) to that of only 3D points (black curves). This represents the improvement by this approach over prior works such as [SCG13, GZS11, SS08] that use only sparse feature matches from the road surface.



(a) Height error



(b) Success rate

Figure 6.6: Error and robustness of our ground plane estimation. (a) Average error in ground plane estimation across Seq 0-10. (b) Percent number of frames where height error is less than 7%. Note that the error in our method is far lower and the robustness far higher than achievable by either method on its own.

6.4 Effectiveness of Ground Plane Estimation

In this section, we demonstrate the effectiveness of our ground plane estimation by integrating with another publicly available monocular SFM system, VISO2-M. It relies on computing relative pose between all consecutive pairs of frames through a fundamental matrix estimation and uses continuous scale correction against a locally planar ground. This system architecture has the advantage of simplicity and robustness. Theoretically, it can not break down, since it does not intend to build long feature tracks, but the resulting disadvantage of low accuracy has been shown in Section 6.2. In this section, we show that our ground plane estimation can significantly improve accuracy of VISO2-M, demonstrating our potential to improve other monocular SFM systems.

We replaced VISO2-M's ground plane estimation with ours (Chapter 5), keeping everything else the same. KITTI training dataset are used for testing. Again, errors relative to ground truth are computed using the metrics in [GLU12]. Error rates are shown in Table 6.2. The method replacing VISO2-M's ground plane estimation with ours is under the name "VISO2-M + Our GP" (right column). Note the translation error is improved by over 4% from 12.4% to 8.15%. Comparing the error numbers of

Table 6.2: The effectiveness of our ground plane estimation is demonstrated by replacing VISO2-M's ground plane estimation module with ours. The new method "VISO2-M + Our GP" achieves over 4% better translation error.

		VISO-M		VISO-M + Our GP	
Seq	Frms	Rot (deg/m)	Trans (%)	Rot (deg/m)	Trans (%)
0	4540	0.0209	11.9	0.0206	6.57
1	1100	0.0735	37.9	0.0746	30.4
2	4660	0.0114	3.33	0.0114	2.73
3	800	0.0197	10.7	0.0192	5.67
4	270	0.0093	7.40	0.0087	1.49
5	2760	0.0328	12.7	0.0333	7.63
7	1100	0.1021	28.2	0.1020	22.6
6	1100	0.0157	4.74	0.0156	4.47
8	4070	0.0203	13.9	0.0203	6.64
9	1590	0.0143	4.04	0.0145	3.04
10	1200	0.0388	25.2	0.0379	21.3
Avg		0.0267	12.4	0.0267	8.15

"VISO2-M + Our GP" with our system's from Table 6.1 demonstrates the effectiveness of our monocular system architecture as well.

The performance of "VISO2-M + Our GP" for KITTI testing dataset is also accessible from the KITTI evaluation webpage[2], under the name **VISO2-M + GP**. Note that the translation error improves by over 4% from 11.94% to 7.46%, compared to the original method **VISO2-M**.

6.5 Effectiveness of Our SFM Architecture

To further demonstrate the effectiveness of our monocular SFM architecture discussed in Section 4.1 and 4.2, we compare our raw SFM performance (without the scale correction of our ground plane estimation) with another well-known SFM system, EKFMonoSLAM [CGDM10]. KITTI odometry dataset Seq 00 - 10 and the metrics in [GLU12] are again used. However, EKFMonoSLAM only successfully finishes three relatively short sequences 03, 04 and 06. The error rates are shown in Table 6.3. The middle column "Our SFM + no GP" shows the error numbers of our system without

[2]`www.cvlibs.net/datasets/kitti/eval_odometry.php`

Table 6.3: The effectiveness of our monocular SFM architecture is demonstrated by comparing the raw SFM performance (without the scale correction of our ground plane estimation) with the state-of-the-art SFM system, EKFMonoSLAM [CGDM10]. Our raw translation error is 10% better than EKFMonoSLAM.

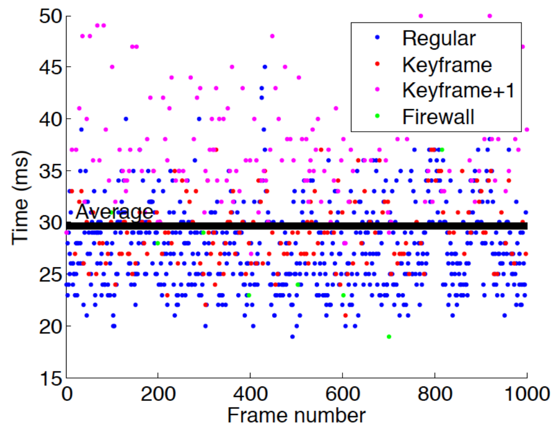
	EKFMonoSLAM		Our SFM wo GP		Our System	
Seq	Rot (deg/m)	Trans (%)	Rot (deg/m)	Trans (%)	Rot (deg/m)	Trans (%)
3	0.014	16.4	0.002	9.66	0.002	3.37
4	0.010	11.6	0.003	2.40	0.002	1.43
6	0.040	27.0	0.013	14.4	0.008	2.09
Avg	0.027	21.2	0.008	11.2	0.005	2.48

enabling the scale drift correction based on the ground plane estimation of Chapter 5. The translation error is higher compared to our full system in the third column, but it is still 10% better than EKFMonoSLAM.

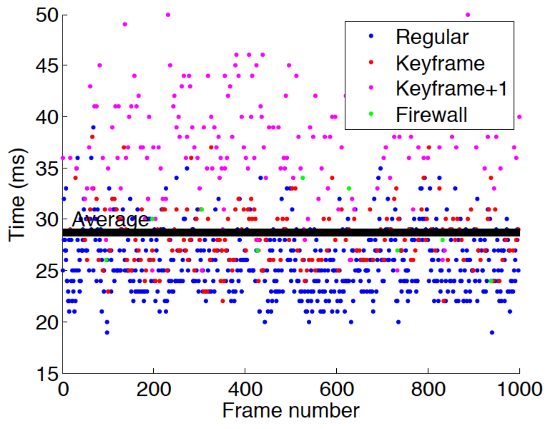
6.6 Real-time Performance

To illustrate our assertion that the system returns real-time pose at an average of 30 fps and a worst-case timing of 50 ms per frame, Figure 6.7 provides the timing graphs of the system on two sequences. In particular, note that the insertion of keyframes, triggering bundle adjustments or error-correcting mechanisms do not result in significant spikes in our timings, which is in contrast to several contemporary real-time systems.

We also observe that keyframes are inserted once in about 5 and 6 frames for sequences 08 and 05, respectively. This is expected since a fast moving vehicle will demand new 3D points from the epipolar update module at frequent intervals. It does not affect the performance of our system since the global bundle adjustment triggered after a keyframe finishes before the next frame's pose computation and runs in parallel to it. In fact, keyframe insertion is an opportunity to introduce long-range constraints in the optimization (so long as the epipolar update module can return long enough tracks). Thus, to ensure speed and accuracy, it is crucial for a multithreaded visual odometry system to not only have a well-designed keyframe architecture, but also to have its various modules such as pose estimation, epipolar search and various bundle adjustments operating in optimal conjunction with each other.



(a) Sequence 05



(b) Sequence 08

Figure 6.7: The runtimes of our system for various types of frames. Blue denotes a steady state frame, red denotes a keyframe, magenta the frame after a keyframe and green denotes a firewall insertion. The black line is the average time per frame, which corresponds to 33.7 fps for sequence 05 and 34.9 fps for sequence 08.

6.7 Monocular SFM on an Additional Public Dataset

We show additional results on the publicly available “The Hague” dataset [DG09]. It consists of three sequences of varying lengths, from 600 m to 5 km. The dataset is challenging due to low resolution 640×480 images, as well as several obstacles due to crowded scenes and moving vehicles close to the camera.

Accurate scale drift correction using the adaptive ground plane estimation of Section 5.2 trained by KITTI dataset allows us to successfully and accurately complete such challenging sequences, in contrast to prior state-of-the-art in monocular SFM. For instance, there are many crowded scenes or long stops in Hague 3 when the system may encounter scale drifts. However, our scale correction that combines cues from sparse 3D points and dense stereo, adapting observation covariances at every frame to reflect relative uncertainties, allows correct scale to be maintained even through such events.

In the absence of ground truth, Table 6.4 reports figures for loop closure or end-point error. Just like our results on the KITTI dataset, these error rates represent a significant improvement over those attainable by prior monocular state-of-the-art and are comparable to state-of-the-art stereo systems.

The reconstructed trajectories for some of the sequences, along with ground truth

Table 6.4: End-point errors in The Hague dataset. These error rates represent a significant improvement over those attainable by prior monocular state-of-the-art and are comparable to state-of-the-art stereo systems.

Sequence	Images	Length (m)	Error (m)	Error (%)
1	2500	609.34	32.72	5.37
2	3000	834.39	16.60	1.99
3	19000	5045.45	244.57	4.85

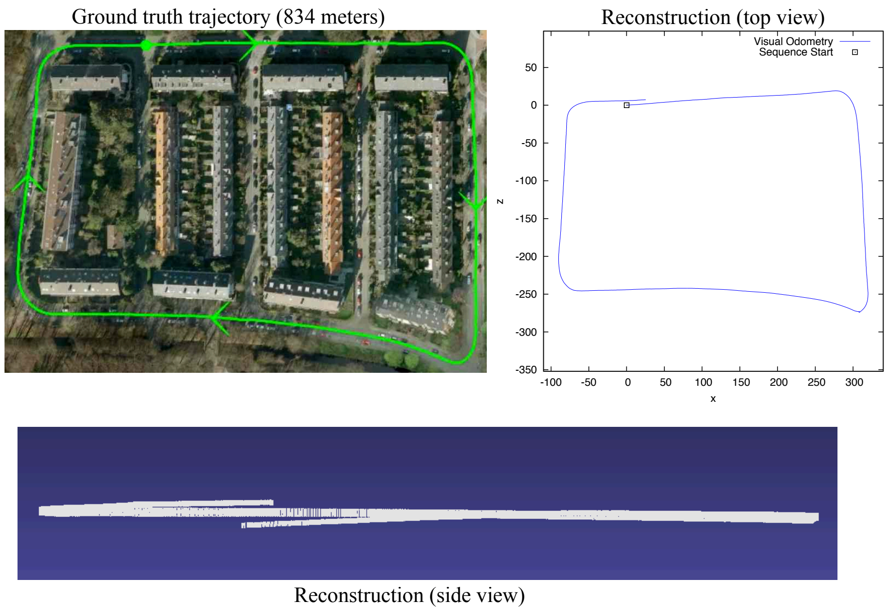
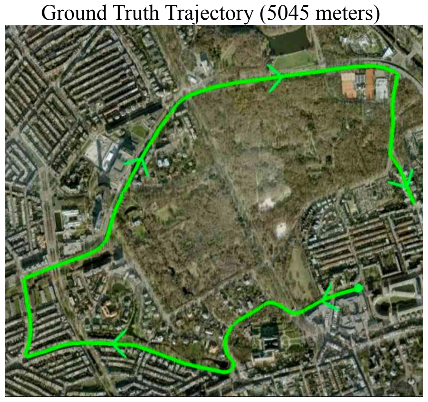


Figure 6.8: The reconstructed trajectories for Seq 2 in Hague dataset, along with ground truth overlays on an aerial map

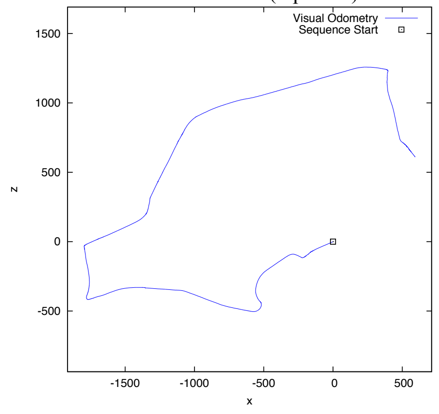
overlays on an aerial map, are shown in Figures 6.8 and 6.9. Note the excellent rotation handling of our system, as well as the ability to maintain correct scale through long sequences.

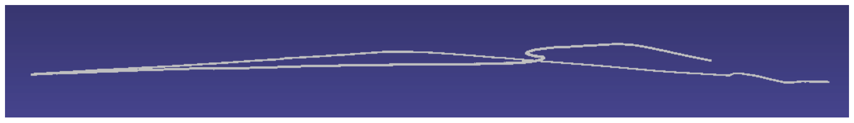
This chapter in part, has been submitted for publication, as it may appear in “High Accuracy Monocular SFM and Scale Correction for Autonomous Driving”, by Shiyu Song, Manmohan Chandraker, Clark C. Guest, in IEEE Transactions on Pattern Analysis and Machine Intelligence.

Ground Truth Trajectory (5045 meters)



Reconstruction (top view)





Reconstruction (side view)

Figure 6.9: The reconstructed trajectories for Seq 3 in Hague dataset, along with ground truth overlays on an aerial map

Chapter 7

Monocular Object Localization

The novel ground plane estimation in Chapter 5 not only improves the SFM system by providing a correction in scale, but also can significantly benefits the localization accuracy for those monocular systems that localizes the objects through a ground plane. We present a framework for highly accurate 3D localization of objects like cars in autonomous driving applications, using a single camera. Our localization framework jointly uses information from complementary modalities like structure from motion (SFM) and object detection, combined through an adaptively estimated ground plane, to achieve high localization accuracy in both near and far fields. This is in contrast to prior works that rely on triangulating detector outputs against a fixed ground plane, or motion segmentation based on sparse feature tracks. To extract SFM cues, we demonstrate the need for and an efficient implementation of dense tracking to handle fast-moving objects like cars whose images are small and low in texture. Rather than completely commit to tracklets generated by a 2D tracker, we make novel use of raw detection scores to allow our 3D bounding boxes to adapt to better quality 3D cues. Our formulation for 3D localization is efficient and can be regarded as an extension of bundle adjustment to incorporate object detection cues. Experiments on the KITTI dataset show the efficacy of each of our cues (3D points, adaptive ground plane and object detection bounding boxes), as well as the accuracy and robustness of our 3D object localization relative to ground truth.

7.1 Introduction

The rapid advent of autonomous driving technologies has introduced the need for highly accurate 3D localization of objects like cars in real-world driving scenarios. The applications of real-time object localization range from driver safety, to danger prediction, to better understanding of traffic scenes. This chapter presents a framework for 3D object localization that combines cues from structure from motion (SFM), object detection and ground plane estimation to achieve excellent performance in real-world driving, using only monocular video as input.

The key to our accurate 3D object localization is the novel joint optimization framework of Section 7.4 that explicitly accounts for SFM and object detection being complementary modalities for scene understanding. Monocular SFM cues consist of 3D points on the object and a per-frame estimate of the ground plane, while detection cues include bounding boxes and detector scores. The mutual interactions of these cues is judiciously governed by our joint optimization to exploit their relative strengths.

Intuitively, SFM can estimate accurate 3D points on nearby objects, but suffers due to the low resolution of those far away. On the other hand, bounding boxes from object detection are obtainable for distant objects, but are often inconsistent with the 3D scene in the near field. Thus, we seek 3D bounding boxes that are most consistent with 2D tracked ones, while also maximizing the alignment of estimated object pose with tracked 3D points (Figure 7.1).

Another example of such interaction is between the object bounding boxes and the ground plane. Each independently moving object in monocular SFM may at best be estimated up to an unknown scale factor. The contact of a 2D bounding box with the ground plane provides a cue to resolve this scale. On the other hand, the fact that 3D bounding boxes lie on the ground can in turn be used to improve the accuracy of the ground plane estimation. Our ground plane is estimated at every frame, as opposed to prior works that use a fixed one for 3D localization [CS10, ELSVG09, WWR+13]. In contrast, we note that both object position and size are closely related to the ground plane, and so must be jointly estimated. This is especially important for far objects, since small errors in ground orientation can lead to large 3D errors at greater distances. Thus, incorporation of SFM cues in the form of an estimated ground plane is also beneficial for

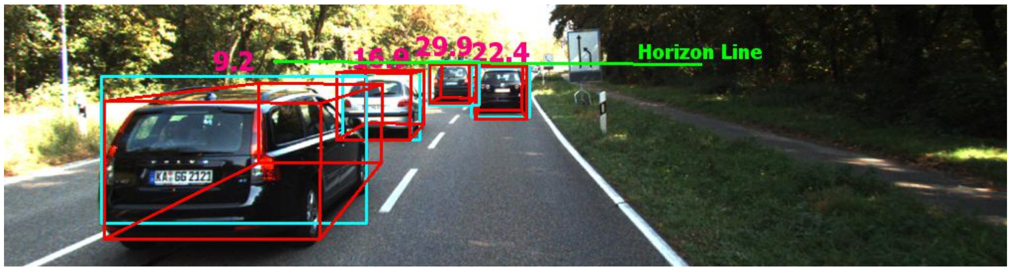


Figure 7.1: Sample output of the 3D object localization framework. We demonstrate a 3D object localization framework that combines cues from SFM, detection bounding boxes and an adaptively estimated ground plane. Cyan denotes 2D bounding boxes, the green line is the horizon from estimated ground plane, red denotes estimated 3D localization for far and near objects, with distances in magenta. Notice that for the closest object, the 3D bounding box is accurate even though the 2D one is not. This is an example of the effectiveness of our joint optimization, that incorporates SFM cues and raw detection scores.

far objects.

Our system is designed for real-time application, and so must eschew complex scene understanding approaches that seek joint solutions for SFM, object tracking, and localization. Rather, those are sequential operations, so inaccuracies in earlier stages must be compensated. The input to 3D object localization are 2D tracklets from tracking-by-detection, which are often noisy and poorly localized. To handle such tracks, Section 7.4.3 proposes a method to incorporate raw detection scores in our joint optimization, while avoiding the prohibitive cost of evaluating the detector model for every possible 3D bounding box and object pose configuration. This allows us to efficiently find detection bounding boxes with high scores that are more consistent with 3D geometry, rather than being confined to estimating the best fit to the output of a 2D tracker. Thus, our framework can be considered an extension of traditional SFM bundle adjustment to also incorporate object cues such as 2D bounding boxes and detection scores.

An important aspect of our object localization is the use of 3D points as SFM cues. In order to obtain reliable 3D points on objects such as cars that occupy a small window in the image and are low in texture, we propose a dense tracking mechanism in Section 7.5. Rather than rely on few and unstable sparse feature matches, we use an intensity alignment approach for pose estimation. The computed pose can provide epipolar constraints to guide a TV-L1 optical flow [WPB$^+$08] and reduce the 2D search to 1D, which allows gains in both speed and accuracy. Further, accurate dense tracks added through this mechanism prevent the system from catastrophic breakdown, as would happen if too few sparse features are available for a traditional PnP-based pose estimation.

We show in Section 7.6 that combining cues from SFM and object detection through the adaptive ground plane significantly improves 3D localization for both near and distant objects. The benefit of our cue combination is available even for more comprehensive monocular scene understanding frameworks such as [CS10, WWR+13]. We demonstrate this in Section 7.6 by using the object tracking of [CS10] within our joint localization framework, to achieve over 5% improvement in 3D localization (evaluated relative to ground truth on KITTI).

To summarize, our principal contributions in this chapter are:

1. A joint optimization framework for 3D object localization that combines cues from SFM cues like 3D points and ground plane, with object cues like bounding boxes and detection scores, to achieve accuracy in both near and far fields.

2. A dense tracking mechanism that can reliably estimate pose and 3D points even for challenging objects like cars in driving scenarios.

3. Incorporation of raw detection scores to allow 3D bounding boxes to “undo” tracking errors, that is, achieve consistency with both 3D geometry as well as detection scores.

7.2 Related Work

Multibody SFM has been proposed in the past to simlutaneously localize a moving camera and moving objects [LKSV07, OSVG07]. In addition, Schindler et al. [SUW06] propose a model selection for segmentation. However, multibody SFM for moving object localization has been demonstrated only for short sequences. Indeed, in real driving scenarios, it is challenging to obtain reliable feature tracks in sufficient numbers for multibody SFM to be robust, due to small object size, fast speeds and lack of texture.

To localize moving objects, Ozden et al. [OSVG07] and Kundu et al. [KKJ11] use joint motion segmentation and SFM. Brox et al. [BRGC10] use a combination of sparse SFM and dense optical flow for joint tracking and segmentation. In practice, it is difficult to obtain stable feature tracks on low-textured objects such as cars when they are not close and segmentation is challenging in actual driving videos where camera and various object motions are often correlated. A different approach is that of multi-target tracking frameworks that combine object detection with stereo [ELSVG09] or monocular SFM [CS10, WWR+13]. Detection can handle more distant objects, decouples feature tracking for individual objects, and together with the ground plane, provides a cue to estimate object scales that are difficult to resolve for traditional monocular SFM even with multiple segmented motions [OSG10].

While multi-target tracking frameworks such as [CS10, ELSVG09, WWR+13] do not show 3D localization results, it is an inherent part of their frameworks. However, they effectively only triangulate detection bounding boxes against a fixed ground plane. Our 3D localization goes much further by using a dynamic ground plane, incorporating SFM cues and making novel use of detection cues. The benefits of each of these is shown in our experiments. Similarly, systems such as [GLW+14] use a stereo setup to infer the layout of urban traffic junctions. All such scene understanding and multi-target tracking frameworks can benefit from our novel use of the ground plane, SFM and detection cues.

Our input is monocular video, so a robust mechanism is needed to estimate 3D points despite the above challenges. This is crucial and our experiments show that traditional sparse SFM cannot handle the needs of autonomous driving. Instead, we propose a combination of dense intensity-aligned pose estimation, an epipolar-guided extension to TV-L1 optical flow of [ZPB07] and consistency checks in the spirit of sparse SFM, to extract 3D points for our joint optimization framework. Unlike [WPB+08], we cannot use a fundamental matrix to constrain flow vectors, since there are not enough sparse feature matches. While more accurate optical flow methods are available [BBM09, YMU13], our focus is on a real-time system and we choose TV-L1 for its balance of accuracy and speed.

7.3 Background

In this section, we define our notation and describe the relevant background concepts.

Notation A vector in $\mathbb{R}^n$ is denoted as $\mathbf{x} = (x_1, \cdots, x_n)^\top$. A matrix is denoted as $\mathbf{X}$. The homogeneous representation of vector $\mathbf{x}$ is denoted as $\widetilde{\mathbf{x}} = (\mathbf{x}^\top, 1)^\top$. A variable x in frame t of a sequence is denoted as x_t or $x(t)$.

Ground plane geometry As shown in Fig. 7.2, the camera height (also called ground height) h is defined as the distance from the principal center of the camera to the ground plane. Usually, the camera is not perfectly parallel to the ground plane and there exists a non-zero pitch angle θ. The ground height h and the unit normal vector $\mathbf{n} = (n_1, n_2, n_3)^\top$ define the ground plane. For a 3D point $(x, y, z)^T$ on the ground plane:

$$h = y \cos\theta - z \sin\theta. \tag{7.1}$$

Object localization through ground plane Accurate estimation of both ground height and orientation is crucial for 3D object localization. Let $\mathbf{K}$ be the camera intrinsic calibration matrix. As [CS10, ELSVG09, WWR+13], the bottom of a 2D bounding box, $\mathbf{b} = (x, y, 1)^\top$ in homogeneous coordinates, can be back-projected to 3D $\mathbf{c} = (c_x, c_y, c_z)^\top$ through the ground plane $\{h, \mathbf{n}\}$:

$$\mathbf{c} = (c_x, c_y, c_z)^\top = -\frac{h\mathbf{K}^{-1}\mathbf{b}}{\mathbf{n}^\top \mathbf{K}^{-1}\mathbf{b}}, \tag{7.2}$$

Similarly, the object height can also be obtained using the estimated ground plane and the 2D bounding box height.

Monocular SFM and ground plane estimation The ground plane is estimated at every frame by adaptively combining cues from sparse SFM, dense inter-frame stereo and object detection, in accordance with their per-frame relative confidences using the method in Chapter 5. This is an important choice – as shown in our experiments, using

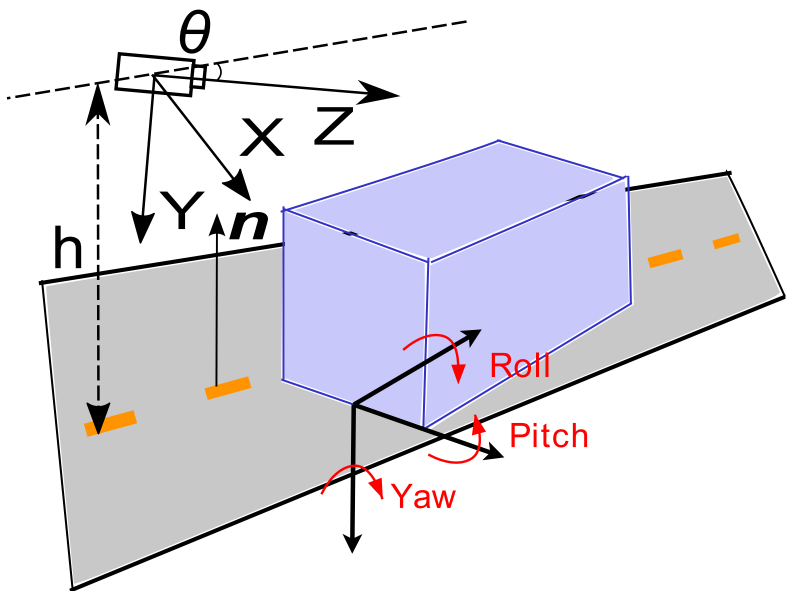


Figure 7.2: Geometry of coordinate system definitions for object localization. The camera height h is the distance from the camera principal point to the ground plane. The pitch angle is θ and $\mathbf{n}$ is the ground plane normal. Thus, the ground plane is defined by $(\mathbf{n}, h)^\top$.

an inaccurate or fixed ground plane from calibration cannot be an option for reliable 3D localization over long sequences.

7.4 Joint Use of SFM and Detection for 3D Object Localization

As discussed in Section 7.1, SFM and 2D object bounding boxes offer inherently complementary cues for scene understanding. SFM produces reliable tracks for nearby objects, but suffers from low resolution in the far field. On the other hand, detection or tracking bounding boxes tend to be consistent with the 3D scene for far objects, but may be inaccurately aligned with the near scene due to perspective challenges. In this section, we propose a framework that combines SFM and 2D object bounding boxes, through an accurate ground plane, to localize both near and far objects in 3D.

We formulate the 3D object localization in an energy minimization framework.

The energy function, $\mathscr{E}$, is a linear combination of SFM and object costs, with additional terms to enforce consistency with prior knowledge:

$$\mathscr{E} = \mathscr{E}_{sfm} + \lambda_o \mathscr{E}_{obj} + \lambda_p \mathscr{E}_{prior}. \tag{7.3}$$

In the following, we explain each cue used for 3D object localization. First, we define the 3D coordinate system and our representation of 3D object pose.

7.4.1 3D Coordinate System

Consider camera coordinate system $\mathscr{C}$ with orthonormal axes $(\boldsymbol{\alpha}_c, \boldsymbol{\beta}_c, \boldsymbol{\gamma}_c)$ and an object coordinate $\mathscr{O}$ with axes $(\boldsymbol{\alpha}_o, \boldsymbol{\beta}_o, \boldsymbol{\gamma}_o)$. Let the origin of object coordinates be the 3D point $\mathbf{c}_o = (x_c, y_c, z_c)^\top$, expressed in camera coordinates, corresponding to center of the line segment where the back plane of the object intersects the ground. Let the ground plane be parameterized as $\mathbf{g} = (\mathbf{n}^\top, h)^\top$, where $\mathbf{n} = (\cos\theta\cos\phi, \cos\theta\sin\phi, \sin\theta)^\top = (n_1, n_2, n_3)^\top$ and $h = -\mathbf{c}_c^T\mathbf{n}$. In $\mathbf{n} = (\cos\theta\cos\phi, \cos\theta\sin\phi, \sin\theta)^\top$, θ is the camera pitch angle and $90^o + \phi$ is the roll angle, where $\phi \in (-180^o, 0^o)$. We assume that the object lies on the ground plane and is free to rotate in-plane with yaw angle ψ. Thus, the object pose is completely determined by a six-parameter vector $\boldsymbol{\Omega} = (x_c, z_c, \psi, \theta, \phi, h)^\top$. The coordinate system definitions are visualized in Figure 7.2.

With the above definitions, one may transform between object and camera systems using the ground plane, object yaw angle and object position. Define $\mathbf{N} = [\mathbf{n}_\alpha, \mathbf{n}_\beta, \mathbf{n}_\gamma]$, where $\mathbf{n}_\gamma = (-n_1, n_3, -n_2)^\top$, $\mathbf{n}_\beta = -\mathbf{n}$ and $\mathbf{n}_\alpha = \mathbf{n}_\beta \times \mathbf{n}_\gamma$. Then, given a homogeneous 3D point $\widetilde{\mathbf{x}}_o$ in the object coordinate system, the transformation from object to camera coordinates is given by:

$$\widetilde{\mathbf{x}}_c = \mathbf{P}_\pi \mathbf{P}_\psi \widetilde{\mathbf{x}}_o, \text{ with } \mathbf{P}_\pi = \begin{bmatrix} \mathbf{N} & \mathbf{c}_o \\ \mathbf{0}^\top & 1 \end{bmatrix}, \quad \mathbf{P}_\psi = \begin{bmatrix} \exp([\boldsymbol{\omega}_\psi]_\times) & \mathbf{0} \\ \mathbf{0}^\top & 1 \end{bmatrix}, \tag{7.4}$$

where $\boldsymbol{\omega}_\psi = (0, \psi, 0)^\top$ and $[\cdot]_\times$ is the cross product matrix.

Finally, the projection function for a 3D point $\mathbf{x}_o$ in object coordinates to the 2D point $\mathbf{u}$ image plane is denoted $\mathbf{u} = \pi_\Omega(\mathbf{x}_o)$. It is simply the inhomogeneous version of

$$\lambda\widetilde{\mathbf{u}} = \mathbf{K}[\mathbf{I}\,|\,\mathbf{0}]\mathbf{P}_\pi \mathbf{P}_\psi \widetilde{\mathbf{x}}_o. \tag{7.5}$$

7.4.2 SFM Cues

Tracked 3D points Let N objects be tracked in the scene over T frames, with object i being tracked from frames s_i to e_i. During this interval, suppose M_i feature points are triangulated by the object SFM mechanism. In the object coordinates, we denote this set of 3D points as $\mathbf{X}_o^i = [\mathbf{x}_1^i, \cdots, \mathbf{x}_{M_i}^i]$. Since the object is rigid, note that the location of each $\mathbf{x}_j^i$ does not depend on time. Let $\mathbf{u}_j^i(t) = (u_j^i(t), v_j^i(t))$ be the 2D point corresponding to $\mathbf{x}_j^i$ in frame t. Then, the first component of the SFM cost favors the object poses $\{\boldsymbol{\Omega}^i(t)\}$, for $i = 1, \cdots, N$ and $t = s_i, \cdots, e_i$ for each object i, that minimize the reprojection error:

$$\mathscr{E}_{reproj}\left(\{\boldsymbol{\Omega}^i(t)\}\right) = \sum_{i=1}^{N} \sum_{t=s_i}^{e_i} \sum_{j=1}^{M_i} \|\mathbf{u}_j^i(t) - \pi_{\boldsymbol{\Omega}^i(t)}(\mathbf{x}_j^i)\|^2. \tag{7.6}$$

Note that there is an overall ambiguity in the origin of $\mathscr{O}$ with respect to $\mathscr{C}$ that cannot be resolved by SFM alone. To do so, we require input from object bounding boxes.

Ground plane Also recall that the background monocular SFM system outputs a ground plane estimate $\bar{\mathbf{g}}(t)$ at every frame t. The object pose as defined in Section 7.4.1 also depends on the ground plane, which is shared across all objects. Note that a shared ground plane is only imposed for physical plausibility and is not a hard requirement for our system. In fact, lower energies can be attained by letting each individual object reside on its own ground plane (and might even be useful in some situations where ground plane variation is high among objects). Thus, an additional SFM cue that informs object pose is the ground plane estimate from the background SFM:

$$\mathscr{E}_{ground}\left(\{\boldsymbol{\Omega}^i(t)\}\right) = \sum_{t=1}^{T} \|\mathbf{g}(t) - \bar{\mathbf{g}}(t)\|^2. \tag{7.7}$$

where $i = 1, \cdots, N$ and $t = s_i, \cdots, e_i$ for each object i, as before.

The combined cost from the SFM cues is now a weighted sum:

$$\mathscr{E}_{sfm} = \mathscr{E}_{reproj} + \lambda_g \mathscr{E}_{ground}. \tag{7.8}$$

7.4.3 Object Cues

Object bounding box Let the dimensions of the object 3D bounding box (to be estimated) be $l_\alpha, l_\beta, l_\gamma$ along the $\boldsymbol{\alpha}_o, \boldsymbol{\beta}_o, \boldsymbol{\gamma}_o$ axes. Then, locations of the 3D bounding box vertices, in object coordinates, are $\mathbf{B} = [\mathbf{v}_1, \cdots, \mathbf{v}_8]$, where we have by definition $\mathbf{v}_1 = (-l_\alpha/2, 0, 0)^\top, \cdots, \mathbf{v}_8 = \left(l_\alpha/2, -l_\beta, l_\gamma\right)^\top$. Note that the $\mathbf{B}$ is in object coordinates and does not vary over time. The projection of the 3D bounding box is

$$\mathbf{q}_k(t) = (p_k(t), q_k(t))^\top = \pi_{\boldsymbol{\Omega}(t)}(\mathbf{v}_k), \text{ for } k = 1, \cdots, 8. \tag{7.9}$$

Let the projected edges of the 3D bounding box at frame t be $\mathbf{b}(t) \in \mathbb{R}^4$, which are found as the extrema of the projected vertices along both the image axes:

$$b_1(t) = \min_k p_k(t),\ b_2(t) = \max_k p_k(t),\ b_3(t) = \min_k q_k(t),\ b_4(t) = \max_k q_k(t), \tag{7.10}$$

for $k \in \{1, \cdots, 8\}$. With a mild abuse of notation, we will denote $\mathbf{b}(t) = \pi_{\boldsymbol{\Omega}(t)}(\mathbf{B})$. Let $\mathbf{d}(t)$ be the corresponding four sides of the tracked 2D bounding box in frame t. Then, we define an object bounding box error:

$$\mathscr{E}_{box}\left(\{\boldsymbol{\Omega}^i(t)\}, \mathbf{B}^1, \cdots, \mathbf{B}^N\right) = \sum_{i=1}^{N} \sum_{t=s_i}^{e_i} \|\pi_{\boldsymbol{\Omega}^i(t)}(\mathbf{B}^i) - \mathbf{d}^i(t)\|^2. \tag{7.11}$$

Object detection scores While we use the output of an object tracker, we must also be aware that tracked bounding boxes are not always accurate. There are two issues that 3D localization must address:

- A 2D tracker might not always pick the detection bounding box with the highest score, rather it can pick one with a high score that is most consistent with any other priors like smoothness and track length. However, these constraints are imposed in 2D by most trackers, so our 3D localization must provide an opportunity for the 3D bounding box to undo any suboptimal choices made by the 2D tracker.
- Further, many tracking-by-detection frameworks work with the detector output after nonmaximal suppression, which results in a discrete set of 2D bounding boxes.

Our localization framework considers a continuous model of the raw detection output, which is crucial for 3D consistency.

To address the first issue, a straightforward approach would be to simply attempt to find the 3D bounding box **B** whose projection maximizes the detection score. However, note that this requires the detector model to be evaluated at every function evaluation of the 3D localization, which can be too expensive for real-time operation. So, we adopt an alternative approach, as described next.

At every frame t, we assume there are M objects and that the detector outputs N candidate bounding boxes denoted $\mathbf{b}_n$ and their respective scores δ_n, for $n = 1, \cdots, N$. We model the distribution of detector scores as a mixture of Gaussians. That is, at each frame t, we estimate 4×4 full-rank covariance matrices $\mathbf{\Sigma}_m$, centered at $\boldsymbol{\mu}_m$, as:

$$\min_{A_m, \mu_m, \Sigma_m} \sum_{n=1}^{N} \left(\sum_{m=1}^{M} A_m e^{-\frac{1}{2}\varepsilon_{mn} \Sigma_m^{-1} \varepsilon_{mn}} - \delta_n \right)^2, \tag{7.12}$$

where $\varepsilon_{mn} = \mathbf{b}_n - \boldsymbol{\mu}_m$ (dependence on t is suppressed for clarity). A Levenberg-Marquardt routine is used for minimization. Example fitting results are shown Fig. 7.3.

Let the model estimated by (7.12) at time t be denoted $\mathscr{S}_t$, which yields a score $\mathscr{S}_t(\mathbf{b})$ for a 2D bounding box $\mathbf{b}$ at frame t. As a result of the above modeling, during every function evaluation for 3D localization, for each putative 3D bounding box $\mathbf{B}$ and object pose $\mathbf{\Omega}_t$, we can estimate the detector score $\mathscr{S}_t(\pi_{\mathbf{\Omega}}(\mathbf{B}))$ without having to evaluate the detection model. We are now in a position to propose an efficient detection cost for 3D localization, which simply attempts to find the 3D bounding box and object pose that achieves the best detection score:

$$\mathscr{E}_{det}\left(\{\mathbf{\Omega}^i(t)\}, \mathbf{B}^1, \cdots, \mathbf{B}^N\right) = \sum_{i=1}^{N} \sum_{t=s_i}^{e_i} \left(\frac{1}{\mathscr{S}_t(\pi_{\mathbf{\Omega}^i(t)}(\mathbf{B}^i))} \right)^2. \tag{7.13}$$

Thus, we use the raw detection output to allow the estimated 3D bounding box to overcome any suboptimal choices made by the 2D tracker in assignment of bounding boxes, while avoiding the cost of too many detector evaluations for every putative 3D bounding box and object pose.

The total object cost is a weighted sum of the bounding box and detection costs:

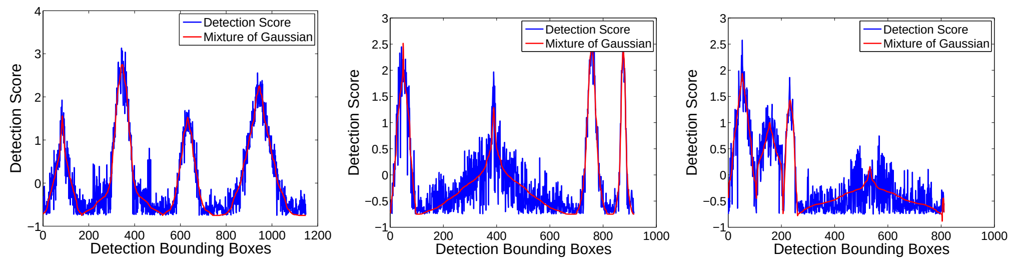


Figure 7.3: Examples of mixture of Gaussians fits to detection scores. Note that our fitting (red) closely reflects the variation in noisy detection scores (blue). Each peak corresponds to an object.

$$\mathcal{E}_{obj} = \mathcal{E}_{box} + \lambda_d \mathcal{E}_{det}. \tag{7.14}$$

7.4.4 Priors

We impose two priors for 3D localization: object size and trajectory smoothness. Let $\mathbf{x}_c(t)$ be the 3D position of the object at frame t in *camera* coordinates (obtained by applying (7.4) on the 3D bounding box $\mathbf{B}$). Further, let $\mathbf{g}(t)$ be the ground plane estimate at frame t. Then the trajectory smoothness prior constitutes a cost given by

$$E_{smooth} = \sum_{t=s_i}^{e_i} \|\mathbf{g}(t) - 2\mathbf{g}(t+1) + \mathbf{g}(t+2)\|^2 + \sum_{i=1}^{N} \sum_{t=s_i}^{e_i} \|\mathbf{x}_c(t) - 2\mathbf{x}_c(t+1) + \mathbf{x}_c(t+2)\|^2.$$

Let $\{\bar{l_\alpha}, \bar{l_\beta}, \bar{l_\gamma}\}$ be the priors on the object dimensions. Then, the size prior is

$$\mathcal{E}_{size} = \sum_{i=1}^{N} \left[(l_\alpha^i - \bar{l_\alpha})^2 + (l_\beta^i - \bar{l_\beta})^2 + (l_\gamma^i - \bar{l_\gamma})^2 \right]. \tag{7.15}$$

The total cost for imposing priors on 3D localization is then given by a weighted sum:

$$\mathcal{E}_{prior} = \mathcal{E}_{smooth} + \lambda_s \mathcal{E}_{size}. \tag{7.16}$$

7.4.5 Joint Optimization

With the above definitions of the various cues, we now define the combined energy function that must be minimized over the set of object poses $\{\mathbf{\Omega}^i(t)\}$, 3D bounding box dimensions $\mathbf{B}^i$ and the set of tracked 3D points on each object $\mathbf{X}_o^i$, for objects $i = 1, \cdots, N$, each of which is visible in frames s_i to e_i:

$$\mathscr{E}\left(\{\mathbf{\Omega}^i(t)\}, \mathbf{B}_1, \cdots, \mathbf{B}^N, \mathbf{X}_o^1, \cdots, \mathbf{X}_o^N\right) = \mathscr{E}_{sfm} + \lambda_o \mathscr{E}_{obj} + \lambda_p \mathscr{E}_{prior}, \tag{7.17}$$

where $\mathscr{E}_{sfm}$, $\mathscr{E}_{obj}$ and $\mathscr{E}_{prior}$ are defined in (7.8), (7.14) and (7.16), respectively. The optimization defined by (7.17) may be regarded as an extension of traditional bundle adjustment to incorporate object cues, in the sense that it is defined over a set of variables $\{\mathbf{\Omega}^i(t)\}$ that constitute "poses" and another set given by $\{\mathbf{B}^i, \mathbf{X}_o^i\}$ that constitutes "3D points". Thus, it can be solved efficiently using a sparse Levenberg-Marquardt algorithm and is fast enough to match the real-time monocular SFM.

To maintain computational efficiency over long sequences, we perform the above joint optimization over a window of $W = 50$ frames. In all our experiments, the parameter values are empirically set to the following fixed values: $\lambda_0 = 0.7$, $\lambda_p = 2.7$, $\lambda_g = 2.7$, $\lambda_d = 0.03$ and $\lambda_s = 0.03$.

7.4.6 Initialization

The success of a local minimization framework as defined in (7.17) is contingent on a good initialization. We again rely on an accurate ground plane estimation along with cues from both 2D bounding boxes and SFM to initialize the variables in (7.17).

Object Poses, $\{\mathbf{\Omega}^i\}$: The initial position of an object, $\widehat{\mathbf{c}}_o = (\widehat{x}_c, \widehat{y}_c, \widehat{z}_c)^\top$, is computed from the object bounding box and ground plane using (7.2). The initial yaw can be estimated from initial object positions in two frames $\widehat{\mathbf{c}}_o^{t-1}$ and $\widehat{\mathbf{c}}_o^{t+1}$. The object's ground height and orientation are initialized to the SFM ground plane. The initial pose of the object is now available as $\widehat{\mathbf{\Omega}} = (\widehat{x}_c, \widehat{z}_c, \widehat{\psi}, \theta, \phi, h)^\top$.

3D Bounding Boxes, $\{\mathbf{B}^i\}$: The initial object dimensions $(\widehat{l_\alpha}, \widehat{l_\beta}, \widehat{l_\gamma})$ are computed as the optimal alignment to the 2D bounding box, by fixing $\widehat{l_\gamma} = \eta \widehat{l_\alpha}$ and minimizing the cost $\mathscr{E}_{box}$ over $\widehat{l_\alpha}$ and $\widehat{l_\beta}$, with a prior that encourages the ratio of bounding box sizes along γ_o and α_o to be η. The practical reason for this regularization is that the camera motion is largely forward and most other vehicles in the scene are similarly oriented, thus, the localization uncertainty along γ_o is expected to be higher. By training on ground truth 3D bounding boxes in the KITTI dataset, we set $\eta = 2.5$ empirically.

3D Points, $\{\mathbf{X}^i\}$: For initialization, each tracked 2D feature point $\mathbf{u}$ is assumed to lie on the plane $\mathbf{n}_\gamma$, orthogonal to the ground. Its position in camera coordinates is

$$\mathbf{x}_c = -(\mathbf{n}_\gamma^\top \mathbf{c}_c)(\mathbf{n}_\gamma^\top \mathbf{K}^{-1}\widetilde{\mathbf{u}})^{-1}\mathbf{K}^{-1}\widetilde{\mathbf{u}}, \tag{7.18}$$

thus, the initial 3D point in object coordinates $\mathscr{O}$ is $\widetilde{\mathbf{x}}_o = \mathbf{P}_\pi^{-1}\mathbf{P}_\psi^{-1}\widetilde{\mathbf{x}}_c$.

7.5 Details of SFM Cues

In this section, we describe some of the challenges for extracting SFM cues on objects such as cars and how we overcome them. Since the background SFM relies on sparse feature tracking, it is natural to initially consider a similar mechanism to extract SFM cues for objects. However, in our autonomous driving application, objects can move fast, have small size and are low in texture. Thus, we propose a dense tracking mechanism to augment SFM cues for tracked objects.

7.5.1 Sparse Feature Tracking

The sparse feature tracking follows similar principles as the background SFM [SCG13, SC14]. The essential task for the primary pipeline is to maintain a set of high quality 3D points, which is achieved by ORB or NCC feature matching and robust PnP based pose estimations in a RANSAC framework. For autonomous driving applications, points rapidly move out of field of view, or suffer from blurs and illumination changes. So a secondary pipeline is tasked with replenishing 3D points – that have already undergone extensive validations in parallel – to the primary pipeline. It has been shown that this architecture achieves very high accuracy for the background SFM process in visual odometry benchmarks relevant to autonomous driving [SCG13, SC14].

The key to the utility of SFM cues proposed in Section 7.4 is the quality of feature matches and 3D points. Due to the small size of many objects of interest, lack of texture and high speeds relative to the camera, it is hard in practice to obtain enough high quality sparse matches for stable PnP-based pose estimation, which also affects the quality of triangulated 3D points. Further, note that having too few sparse matches and bad pose estimates in effect leads to a breakdown in the object tracking framework, and so must be avoided. Thus, in Section 7.5.3, we propose a dense tracking mechanism to extract SFM cues for object localization.

7.5.2 Pose Estimation by Intensity Alignment

As we will see in Section 7.5.3, the knowledge of camera pose is crucial for both speed and accuracy of dense tracking. However, the PnP-based pose computation of the sparse pipeline requires prior knowledge of feature tracks. In this section, we propose to use a dense pose estimation based on image intensity alignment [KSC13]. This has two benefits – first, it does not rely on feature matching, so the epipolar constraints introduced by this pose estimation can instead be used to improve the quality of feature tracking. Second, we show in experiments that the quality of pose estimated by intensity alignment is better in our application, since the number of sparse matches may be too low for effective PnP-based pose estimation.

Recall that object pose is defined in Section 7.4.1 as $\mathbf{\Omega} = (x_c, z_c, \psi, \theta, \phi, h)^\top$. Supposing the object pose $\mathbf{\Omega}(t)$ in frame t is known, along with a set of reliable 3D points, $\{\mathbf{x}\}$. Then the object pose $\mathbf{\Omega}(t+1)$ can be estimated by minimizing the intensity difference between the projections of the 3D points in two neighboring frames:

$$\min_{\mathbf{\Omega}(t+1)} \sum_{i=1}^{N} \left[\mathbf{I}_t\left(\pi_{\mathbf{\Omega}(t)}(\mathbf{x}_i)\right) - \mathbf{I}_{t+1}\left(\pi_{\mathbf{\Omega}(t+1)}(\mathbf{x}_i)\right)\right]^2 \tag{7.19}$$

As this intensity alignment is only valid for a small motion between $\mathbf{\Omega}(t)$ and $\mathbf{\Omega}(t+1)$, the above optimization is embedded into an iterative warping approach to handle large motions. After the pose estimation, outliers in feature matches are removed by checking the reprojection error with the estimated object pose. This estimated object pose is refined by the joint optimization framework of Section 7.4, by taking into account all the cues including 3D points and object bounding boxes. With all individual cues including 3D points and 2D matches in SFM cues, we are now in a position to use this pose for improving the quality of 3D points obtained by dense tracking.

7.5.3 Dense Feature Tracking

For normal SFM applications, the sparse features are sufficient. However, due to low texture, specularity and small size of the object, we found that the sparse features are not sufficient for us to have a robust feature tracking. Therefore, we introduce the dense feature matching.

For dense feature tracking, we rely on TV-L1 optical flow between neighboring frames t and $t+1$. To maximize efficiency, we only compute optical flow within the small sub-image defined by the object bounding box. In practice, optical flow tracks can be very noisy, while high quality 3D points are crucial for our SFM cues. So, besides the feature selection mechanism of [SBK10], we also divide each object region into 8×8 buckets and only the pixels with the highest Harris corner responses are selected to be tracked.

Epipolar Guided TV-L1 Optical Flow Having access to the pose computed from intensity alignement, rather than from feature tracks, provides the luxury to use the epipolar constraints from the computed pose to guide the optical flow that generates features tracks. This has two benefits – first, the optical flow is much faster since a 2D search on the image plane is reduced to a 1D search on the epipolar line. Second, the accuracy of feature tracks also improves since optical flow vectors are now constrained to be consistent with epipolar geometry.

The epipolar line is given by a point $\mathbf{p}_0$, along with a direction vector $\mathbf{p}$, $\mathbf{p}_0$ is the closest point to the point at the source image on the epipolar line in the destination image. Points on the epipolar line can be described by a single parameter u, as $\mathbf{p}_0 + u\mathbf{p}$. Given images I_t and I_{t+1} at frames t and $t+1$, let Γ be the region of image I_t under consideration. For pixels $\mathbf{x} \in \Gamma$ with corresponding epipolar lines $\{\mathbf{p}_0, \mathbf{p}\}$, the TV-L1 energy function is

$$E = \int_{\Gamma} \lambda |I_t(\mathbf{x}) - I_{t+1}(\mathbf{p}_0 + u\mathbf{p})| + |\partial u| \, d\mathbf{x} \tag{7.20}$$

Next, we linearize the image I_1 around $\mathbf{p}_0 + u_0\mathbf{p}$, where u_0 is the putative epipolar guided match:

$$I_{t+1}(\mathbf{p}_0 + u\mathbf{p}) = I_{t+1}(\mathbf{p}_0 + u_0\mathbf{p}) + (u - u_0)\nabla_{\mathbf{p}} I_{t+1}(\mathbf{p}_0 + u_0\mathbf{p}), \tag{7.21}$$

with $\nabla_{\mathbf{p}} I_{t+1}$ denoting the derivative of I_{t+1} along the direction $\mathbf{p}$. The TV-L1 cost function now becomes:

$$E = \int_{\Gamma} \lambda |u\nabla_{\mathbf{p}} I_{t+1} + I_{t+1}(\mathbf{p}_0 + u_0\mathbf{p}) - u_0\nabla_{\mathbf{p}} I_{t+1} - I_t| + |\partial u| d\mathbf{x} \tag{7.22}$$

Denoting $\rho(u) = (u - u_0)\nabla_{\mathbf{p}} I_{t+1} + I_{t+1}(\mathbf{p}_0 + u_0\mathbf{p}) - I_t$ and by introducing an auxiliary variable v, we can minimize a convex approximation of (7.22):

$$E_\theta = \int_{\Gamma} |\partial u| + \frac{1}{2\theta}(u - v)^2 + \lambda |\rho(v)| \, d\mathbf{x}, \tag{7.23}$$

where θ is a small constant that ensures v is a close approximation of u. This is now in the same form as the TV-L1 optical flow in [ZPB07], but with u a 1D variable instead of 2D. Thus, the same approach as [ZPB07] can be used to solve (7.23) much faster.

Dense Tracking Framework The tracks obtained by the epipolar-guided optical flow are triangulated to obtain 3D points. However, optical flow based tracking is noisier than descriptor-based matching, so the 3D points must be validated before use in object localization. We use a mechanism similar to the sparse matching pipeline for this validation. The triangulated 3D point is reprojected into the past few images where it is visible. Only those points are retained for whom the NCC scores corresponding to all the reprojections are above a threshold. Now we have a new set of 3D points ready to be added to the master thread when required. These 3D points are also refined by the joint optimization framework of Section 7.4 that incorporates other cues too.

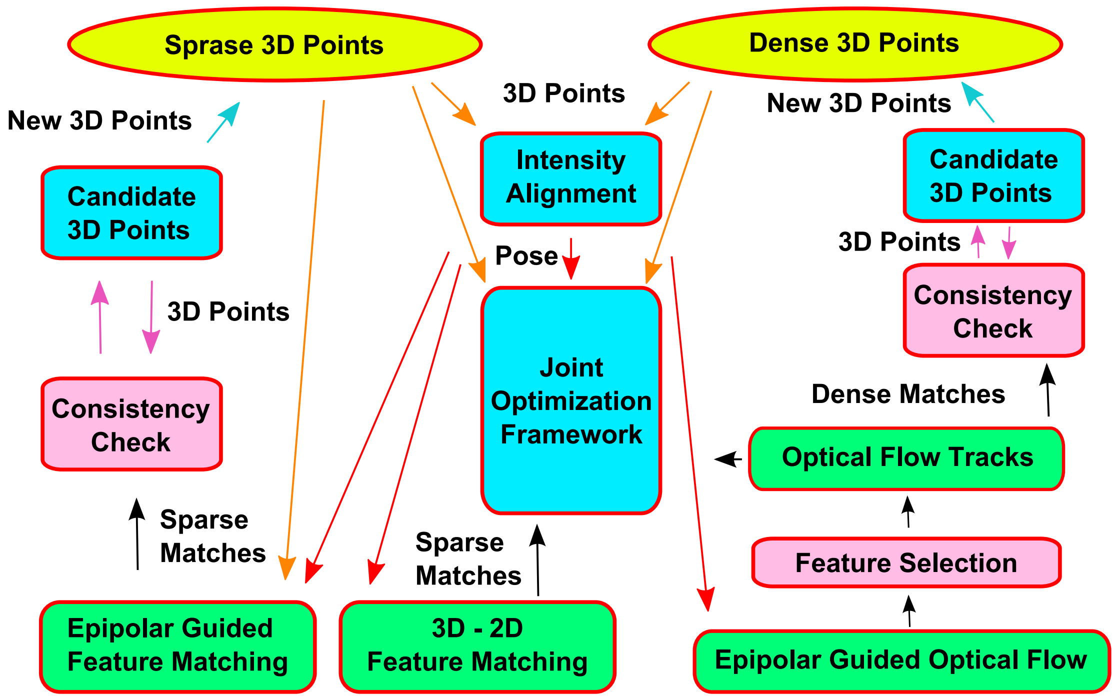


Figure 7.4: Workflow for 3D points cue in 3D object localization framework. We maintain two sets of 3D points, sparse and dense. 3D-2D matching maintains the sparse set. These points together with the dense ones are used by the intensity alignment pose estimation. Epipolar guided feature matching replenishes sparse 3D points through a series of validation mechanisms. For dense matches, we track corner-like features from an epipolar-guided optical flow. Those tracks either provide new 3D points again through a series of validations or provide feature matches for dense 3D points. The estimated pose and the entire set of 3D points are used by the joint optimization framework as part of the SFM cue.

7.6 Experiments

We present evaluation on the state-of-art KITTI dataset [GLU12], which contains real-world driving sequences. Since KITTI doesn't provide a benchmark for object localization and the ground truth 3D labels are not public for the testing dataset, we evaluate our 3D localization with the training dataset of the tracking benchmark. We demonstrate 3D localization on tracked bounding boxes computed using [GLW$^+$14], [CS10] and also the ground truth. Our system has been extensively tested on real-world driving scenarios to demonstrate its excellent performance.

The joint framework for 3D localization presented in Section 7.4 uses several cues to achieve high accuracy. First, it adaptively estimates the ground plane at every frame, instead of relying on a fixed one. Second, it estimates and tracks 3D bounding boxes that are consistent with 2D tracks. Next, it uses raw detection cues to allow the 3D localization to recover from possibly suboptimal choices made by the 2D tracker. Finally, it incorporates SFM cues in the form of epipolar-guided dense feature tracks. Our experiments show the benefit of each of these cues.

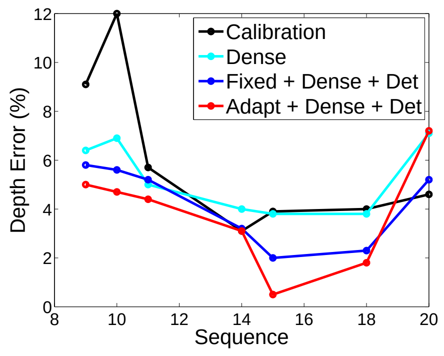
7.6.1 Localization with Different Ground Plane Estimations

Object localization is an important application of our ground plane estimation. Accurate estimation of both ground height and orientation is crucial for 3D object localization. Now we demonstrate the benefit of the adaptive ground plane estimation of Chapter 5 for 3D object localization.

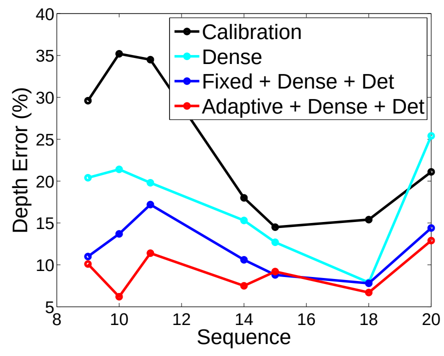
KITTI does not provide a localization benchmark, so we instead use the tracking training dataset to evaluate against ground truth. We use Seq 1-8 for training and Seq 9-20 for testing. The metric we use for evaluation is percentage error in object position. For illustration, we consider only the vehicle objects and divide them into “close” and “distant”, where distant objects are farther than 10m. We discard any objects that are not on the road. Candidate bounding boxes for training the object detection cue are obtained from [FGMR10], called DP here).

Fig. 7.5 compares object localization using a ground plane from our data-driven cue combination (red curve), as opposed to one estimated using fixed covariances (blue), or one that is fixed from calibration (black). The top row uses ground truth object tracks, while the bottom row uses tracks from the state-of-the-art tracker of [PRF11]. For each case, observe the significant improvement in localization using our cue combination. Also, from Figs. 7.5(b),(d), observe the significant reduction in localization error by incorporating the detection cue for ground plane estimation for distant objects.

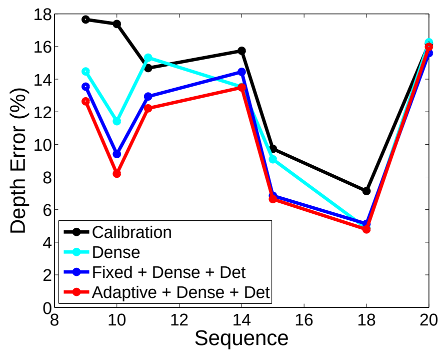
Fig. 7.1 shows an example from our localization output. Note the accuracy of our 3D bounding boxes (red), even when the 2D tracking-by-detection output (cyan) is not accurate.



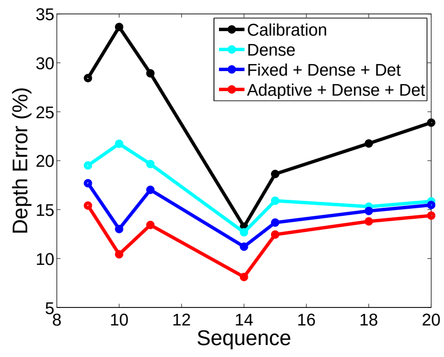
(a) Close objects (GT)



(b) Distant objects (GT)



(c) Close objects (DP)



(d) Distant objects (DP)

Figure 7.5: Comparison of 3D object localization errors for calibrated ground, stereo cue only, fixed covariance fusion and adaptive covariance fusion of stereo and detection cues. (Top row) Using object tracks from ground truth (Bottom row) Using object tracks from [PRF11]. Rrrors reduce significantly for adaptive cue fusion, especially for distant object where detection cue is more useful.

7.6.2 Effectiveness of Different Cues

To demonstrate the effectiveness of each of the above contributions, we show the object localization accuracy with different methods in Table 7.1 and 7.2 using 2D bounding boxes from ground truth, as well as the tracking output of [GLW$^+$14]. For each table, the left column lists different methods as various cues are added in the localization framework. The most important evaluation metric for our autonomous driving application is the distance accuracy in depth, $Z(\%)$. We also list the horizontal localization accuracy, $X(\%)$.

We differentiate between near and far objects in evaluating the results (although our localization method does not make any such distinction). This is to show the effectiveness of different cues at various distance ranges. For instance, we expect SFM cues to be more effective in the near range, while we expect the ground plane estimation to have a significant impact on far objects. We consider objects up to 15 meters away to be close. Although SFM cues are effective up to about 30 meters for the camera resolutions in the KITTI dataset, we find that SFM cues are most stable within 15 to 20 meters.

`CalibGround` denotes the baseline method where localization is performed by directly back-projecting the bottom of the tracked 2D bounding box to 3D using (7.2), with a fixed ground plane. The fixed ground plane is obtained from the calibration, which is $(\mathbf{n}^\top, h)^\top = (0, -\cos(0.03), -\sin(0.03), 1.7)^T$ for the KITTI dataset. Note that this is essentially how the ground plane is used for localization in several prior works [CS10, ELSVG09, WWR$^+$13]. Observe that the localization errors are very high – clearly this is not suitable for autonomous driving applications.

`AdaptiveGround` uses the same back-projection of (7.2) to estimate the location of the 3D bounding box, however, the ground plane used is adaptively estimated at every frame, replicating the method of [SC14]. It is observed that the localization accuracy is improved significantly, especially for far objects, since small errors in ground plane orientation can have a large impact on 3D error over longer distances. We emphasize that a good ground plane is not only crucial for localization accuracy, but also has an important role in the stability of the joint optimization framework. We have the global object size variables across all the frames and there are object size prior terms and object bounding box terms in the joint optimization framework. Note that both object size and distance are variables in our optimization framework. The object size, the ground plane and the object distance are highly correlated entities, so poor ground plane estimation in a few frames may cause the optimization to break down. Thus, these results demonstrate the high quality of the ground plane estimation as well.

In `Ground+Opt`, besides using the adaptive ground plane, we also estimate 3D bounding boxes that best fit the tracked 2D bounding boxes. Priors are also enabled for 3D trajectory smoothness and 3D size constancy. As we can see, injection of further 3D cues causes the errors to decrease, especially for near objects.

Next, we add object detection cues in the joint optimization framework to incorpo-

Table 7.1: Comparison of 3D object localization errors for various cues used in our joint optimization framework, with bounding boxes from ground truth. The benefits of each of adaptive ground plane, object bounding boxes, detection scores and 3D points are clearly visible, as is the performance benefit from our dense tracking.

Method	Ground Truth			
	Close Objects		Far Objects	
	Z(%)	X(m)	Z(%)	X(m)
Calib Ground	10.15	0.526	25.27	0.79
Adaptive Ground	8.97	0.379	9.81	0.35
Ground+Opt	6.42	0.261	8.90	0.35
Ground+Opt+Det	6.08	0.246	8.63	0.33
Ground+Opt+Det+PnP	5.87	0.243	8.54	0.34
Ground+Opt+Det+Align	5.49	0.240	8.26	0.33

Table 7.2: Comparison of 3D object localization errors for various cues used in our joint optimization framework, with the tracking output of [GLW+14]. The benefits of each of adaptive ground plane, object bounding boxes, detection scores and 3D points are clearly visible, as is the performance benefit from our dense tracking.

Method	TBD [GLW+14]			
	Close Objects		Far Objects	
	Z(%)	X(m)	Z(%)	X(m)
Calib Ground	13.29	0.58	26.90	0.75
Adaptive Ground	13.89	0.50	10.17	0.33
Ground+Opt	9.49	0.33	9.42	0.34
Ground+Opt+Det	9.43	0.32	9.48	0.33
Ground+Opt+Det+PnP	9.37	0.30	11.19	0.37
Ground+Opt+Det+Align	8.29	0.28	10.43	0.36

rate raw detection scores. The results are shown in the row labeled `Ground+Opt+Det`. It is clear that the error decreases further, since the system can now search for 2D detection bounding boxes that have high scores and are more consistent with 3D geometry.

In `Ground+Opt+Det+PnP`, we incorporate SFM cues in the joint optimization framework, but with a PnP based pose estimation. A full optical flow must be used now instead of an epipolar-guided one, since feature tracks are precursors to PnP. The remaining validation mechanisms are the same as the description in Section 7.5. However, due to the challenging size and texture of the objects under consideration, PnP is limited by outliers from unstable tracking. As we see, the improvement from adding PnP based SFM cues is quite limited. (We also attempted a PnP based object pose estimation based on sparse SIFT feature matches, however, the number of matches are too few, which

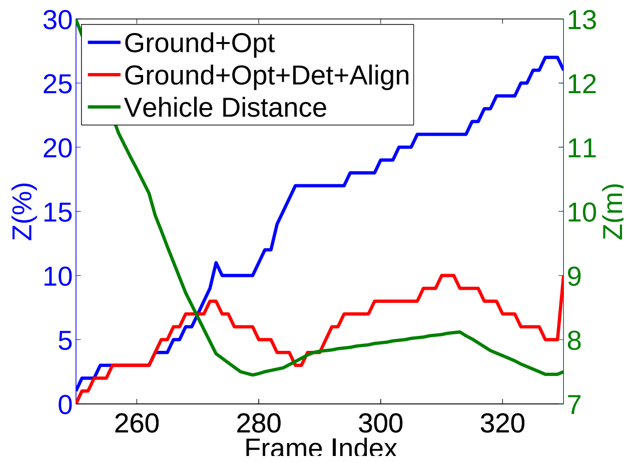


Figure 7.6: Benefit of SFM cues for 3D object localization. The green curve plotted against the right axis shows distance of an object as it approaches the camera. On the left axis, the blue curve shows object depth error when only object bounding box cues are used for localization, while the red curve incorporates SFM cues. SFM cues have a significant impact on localization accuracy in the near field.

causes breakdowns due to highly inaccurate poses.)

Finally, in `Ground+Opt+Det+Align`, we use dense tracking based on epipolar-guided optical flow, along with the intensity alignment based pose estimation to replace the PnP. It is seen that errors decrease for the ground truth bounding boxes in Table 7.1 and 7.2, but even more so for the actual detection bounding boxes. This clearly demonstrates that SFM cues can help 3D object localization to account for unstable detection and tracking inputs. Intuitively, SFM cues are expected to be more helpful for close objects, for which better quality 3D points can be estimated. Our results are consistent with this intuition.

Table 7.3: Our 3D object localization can improve the performance of existing scene understanding frameworks such as [CS10]. This is due to our joint optimization that makes judicious use of an adaptive ground plane, 3D points, object bounding boxes and detection scores. Note that [SC14] uses a global optimization, while we use a windowed one and yet perform better in many instances.

Seq. No.		0004				0047				0056
Object ID		1	2	3	6	0	4	9	12	0
No. of Frames		91	251	284	169	170	96	94	637	293
Z (%)	MTT [CS10]	14.4	17.6	12.9	12.3	16.2	18.1	13.8	11.6	13.9
	MLMSFM [SC14]	4.1	6.8	5.3	7.3	9.6	11.4	7.1	10.5	5.5
	Ours	6.04	5.64	4.93	5.92	5.86	12.46	7.04	8.24	6.01

7.6.3 Effectiveness of SFM Cues

To further illustrate the relative benefits accrued by SFM cues, Figure 7.6 shows a sample output of the system from a few frames in the KITTI dataset. The green curve corresponds to distance of an object as it approaches the camera (right axis). The left axis shows the error in depth estimate for the methods `Ground+Opt` (blue cirve) and `Ground+Opt+Det+Align` (red curve). The latter includes SFM cues, while the former does not. It can be seen that SFM cues are inactive when the object is further, but keep the error rate low when the object is near. On the other hand, ignoring SFM cues and relying only on object bounding boxes impacts performance in the near field.

7.6.4 Comparison with [CS10]

Finally, we show the improvement in 3D localization that our use of SFM and detection cues affords for other scene understanding frameworks. We use the tracking output provided by [CS10] on a few KITTI sequences, along with its raw detection output based on [FGMR10]. The localization error is compared to the method of [SC14] that only uses an adaptive ground plane and detection bounding boxes, as well as our method that additionally incorporates 3D points and detection scores. Note that [SC14] performs a global optimization over all the frames, while we use only a windowed optimization. It is evident from Table 7.3 that our use of SFM and detection cues can also benefit other scene understanding frameworks.

7.7 Discussion and Future Work

We have presented a joint optimization framework for 3D object localization, designed for autonomous driving applications. It recognizes and exploits the complementary strengths of SFM cues (3D points and ground plane) and object cues (bounding boxes and detection scores), to achieve good localization accuracy in both near and far fields. Our system is efficient and can operate in real-time – it can be considered an extension of traditional bundle adjustment with object cues. The generality of our framework means it can be used to readily improve the performance of most 3D scene understanding systems that rely on object tracking and a ground plane. Our system uses object detection as input and a challenge for future work is to obtain this input in real-time. Our future work also explores the use of our 3D object localization in autonomous driving applications that involve comprehensive scene understanding.

This chapter has been submitted for publication, as it may appear in "High Accuracy Monocular 3D Object Localization for Autonomous Driving", by Shiyu Song, Manmohan Chandraker, in proceedings of European Conference on Computer Vision, Zurich, September 6-12, 2014.

Chapter 8

Monocular Lane Detection

The problem of lane detection is an important component for many intelligent vehicle applications, such as lane departure warning, lane keeping, turn assist, autonomous driving, traffic scene understanding, and so on. The structure from motion (SFM) system in Chapter 4 and the ground plane estimation in Chapter 5 can significantly improve the performance of a lane detector by incorporating SFM and ground plane cues. For example, a common technique used in lane detection, the bird-eye-view transform, requires a homography plane (the ground plane). A more accurate ground plane can improve the quality of the bird-eye-view transform. Another example is that a common problem to solve in a video-based lane detection is line segment matching. The epipolar constraint from computed SFM poses enables us to solve this problem more effectively. In this chapter, we present a lane detection system incorporating our SFM poses and ground planes from our SFM system.

8.1 System architecture

In contrast to prior lane detection algorithms, our system incorporates SFM pose cues and ground plane cues. The system workflow proceeds in a similar way as our SFM system.

At steady state, the system has access to a set of 3D lanes. A 3D lane is represented by a parabolic curve in 3D, on the ground. In each new frame, we extract line segments, try to find supporting line segments for each existing 3D lane, and trim the existing 3D lanes according to them, so that we track these 3D lanes. To distinguish from the step "line segment tracking" defined later, we call this step "lane tracking".

When the new frames contain new lanes that have not been detected, we add these lanes in a keyframe. To find the candidate lanes to add in a keyframe, we track extracted line segments over $L = 3$ frames. If a line segment can be consistently tracked over L frames, we believe that there is a good lane associated with this line segment to be added. This step is called "line segment tracking". After a series of steps, such as non-maximum suppression, lane selection, lane fitting and so on, the new lane is selected, created and added to the set of 3D lanes maintained by our system. In the following sections, we describe the details of each step.

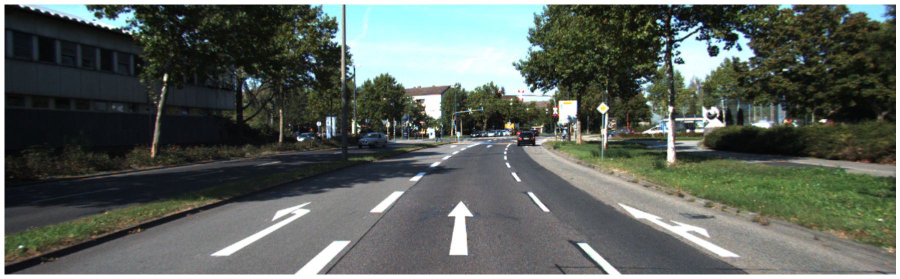
8.2 bird-eye-view Transform

To extract line segments, we find that a bird-eye-view transform of the original image can significantly boost the performance, because of the perspective distortion. Given an estimated ground plane from Chapter 5, we can construct a homography matrix to transform the image from a normal view to a bird-eye view as shown in Figure 8.1. As we can see, obviously the perspective in the bird-eye-view is distorted. It is expected that the line segment algorithms will work significantly better, which we'll see in the next section.

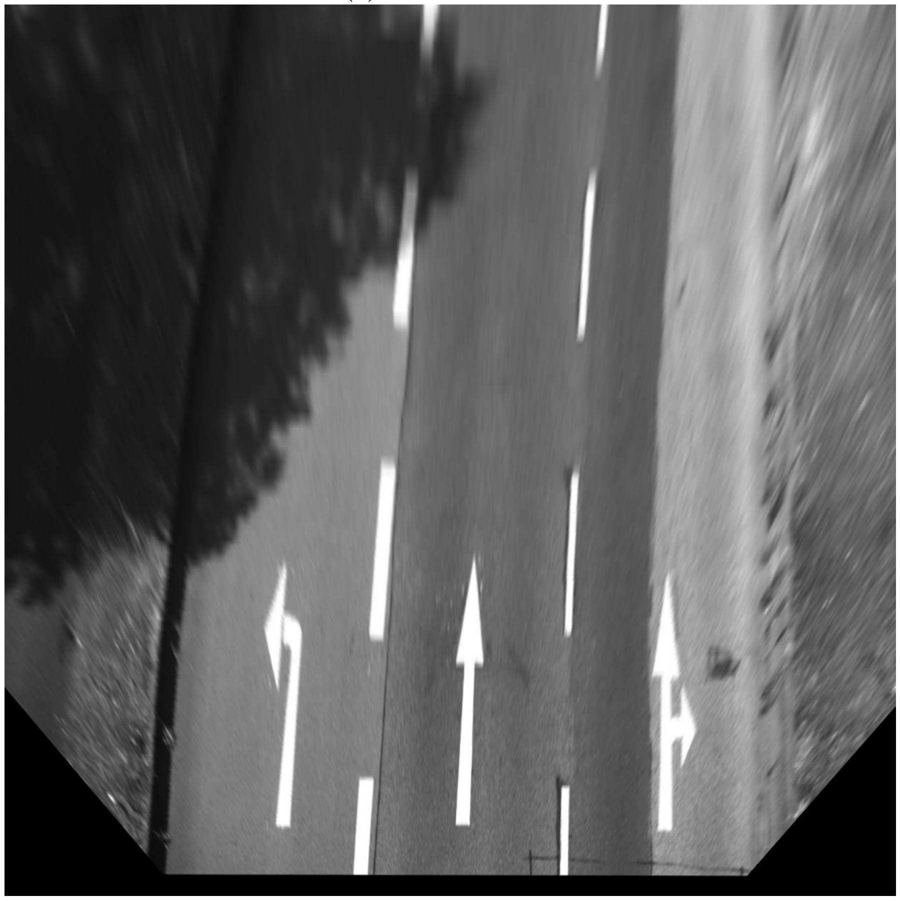
8.3 Line Extraction

To extract line segments, we first compute the gradient map of the input image. The gradient map is computed by using OpenCV's Sobel function with first order derivative in the x and y direction and kernel size 5. The computed gradient map is shown in Figure 8.2.

The computed gradient map is converted to a binary map by applying a response threshold (120 for bird-eye-view and 170 for normal view). The binary map is used as the input to the commonly used Hough transform algorithm and its variation [Hou59, DH72, MGK00] to extract line segments, as shown in Figure 8.3. Note some line segments

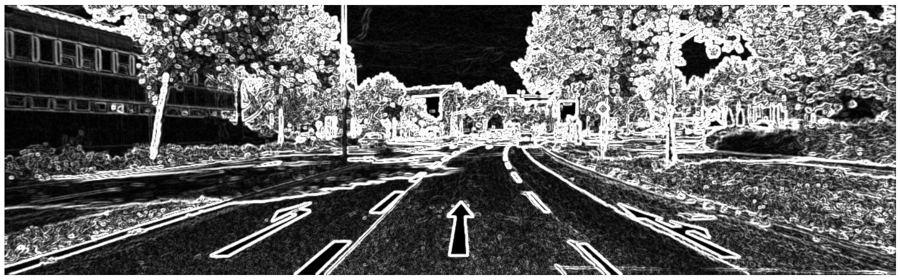


(a) Normal View

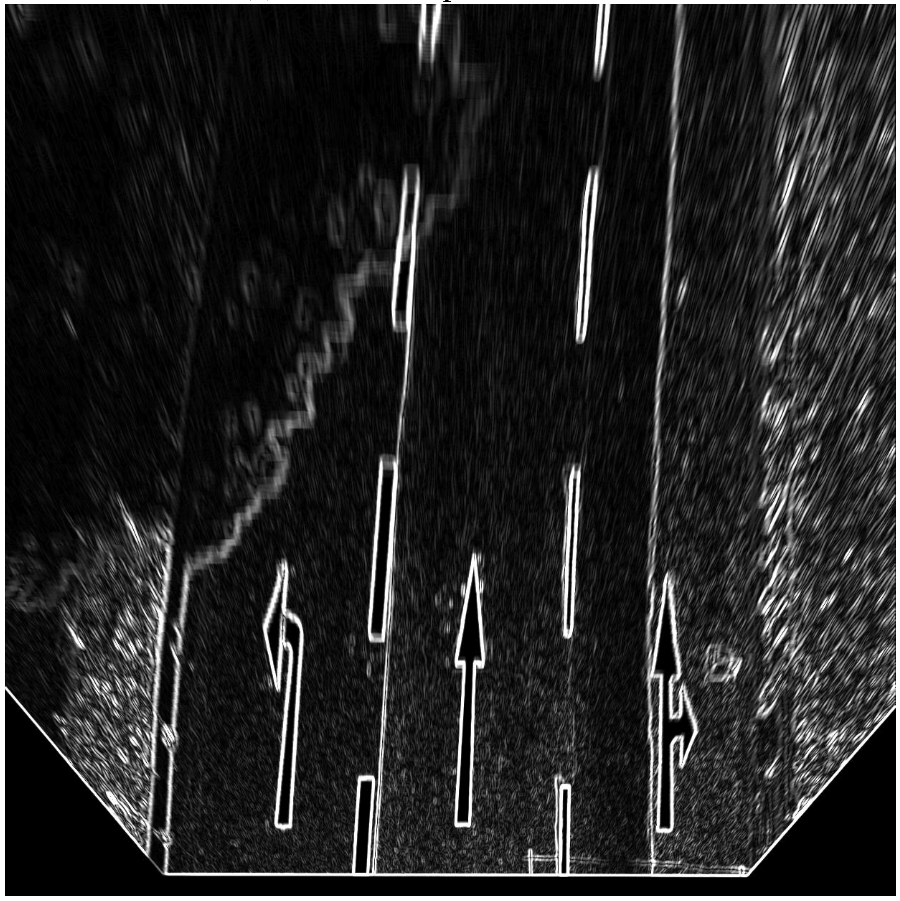


(b) Bird-eye-view

Figure 8.1: Bird-eye-view transform of an image in KITTI dataset. (a) The normal view (b) The bird-eye-view transformed based on the ground plane estimation in Chapter 5.



(a) Gradient map in normal view



(b) Gradient map in bird-eye-view

Figure 8.2: Gradient map of an image and its bird-eye-view transform in KITTI dataset. (a) Gradient map in normal view (b) Gradient map in bird-eye-view.

above the ground are also extracted. These line segments can be eliminated easily by taking the horizon computed from the ground plane into consideration. To achieve the best robustness, our system extracts line segments from both the normal view and the bird-eye-view, and saves them for line segment tracking and lane tracking.

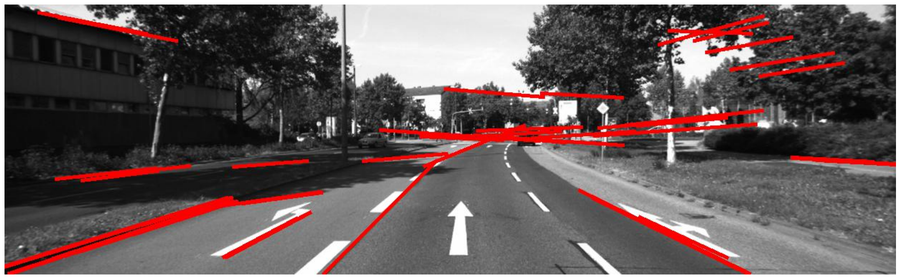
8.4 Lane Tracking and Temporal Integration

At steady state, the system has access to a set of 3D lanes. The lane tracking step tracks the existing 3D lanes by finding supporting line segments in the new frame, trims and updates the 3D lanes according to them.

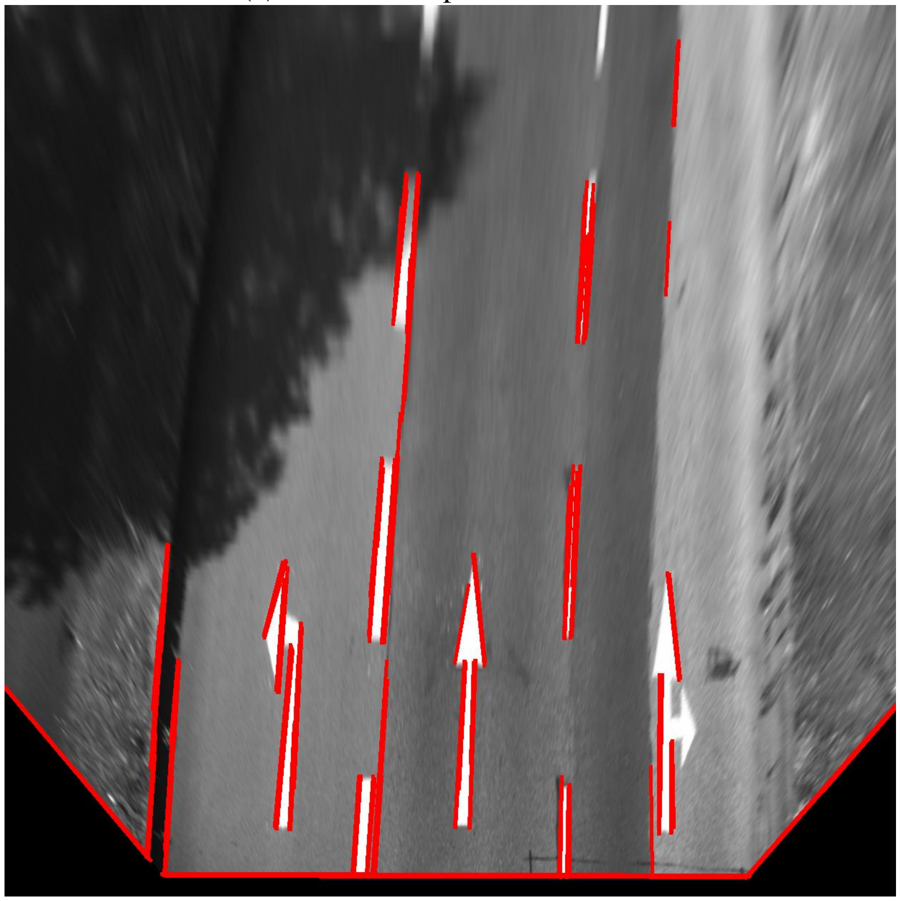
Given the camera intrinsic matrix and SFM poses, we can project the 3D lanes to 2D, and then convert them to bird-eye-view. Given each 2D lane in bird-eye-view, we try to find all the similar extracted line segments in the view. If the orientation difference between the lane and the line segment is smaller than 5 degree and the distance between the end points of the line segment and the lane is smaller than 35 pixels, we consider the line segment and the lane are *similar*. After finding all the similar line segments to a lane, we add the end points of these similar line segments to the lane as new control points. Recall that a lane is represented by a fitted parabola curve. A parabola curve is fitted by a set of control points. By updating these control points, the lane is updated. If a lane can't find any supporting (similar) line segments, we consider that the lane has disappeared and eliminate it. The 2D control points will be triangulated to 3D against the ground plane by using Equation 7.2 in Section 5.1 of Chapter 7. In this way, the 3D lanes are updated as well.

The procedure is illustrated in Figure 8.4. In the figure, a lane is represented as a line segment with only two control points for simplicity. In frame $t-k$ and t, we find the supporting line segments and triangulate the control points to get the 3D lane. The 3D lane is then projected to the new frame $t+1$. In frame $t+1$, we found the supporting line segment (in green), and then updated the 3D lane according to it.

So far, we can consistently tracking existing 3D lanes. To add new 3D lanes into the system, we need a mechanism for line segment tracking and lane adding in a keyframe.



(a) Gradient map in normal view



(b) Gradient map in bird-eye-view

Figure 8.3: Line segment extraction of an image and its bird-eye-view transform in KITTI dataset. (a) Line segment extraction in normal view (b) Line segment extraction in bird-eye-view.

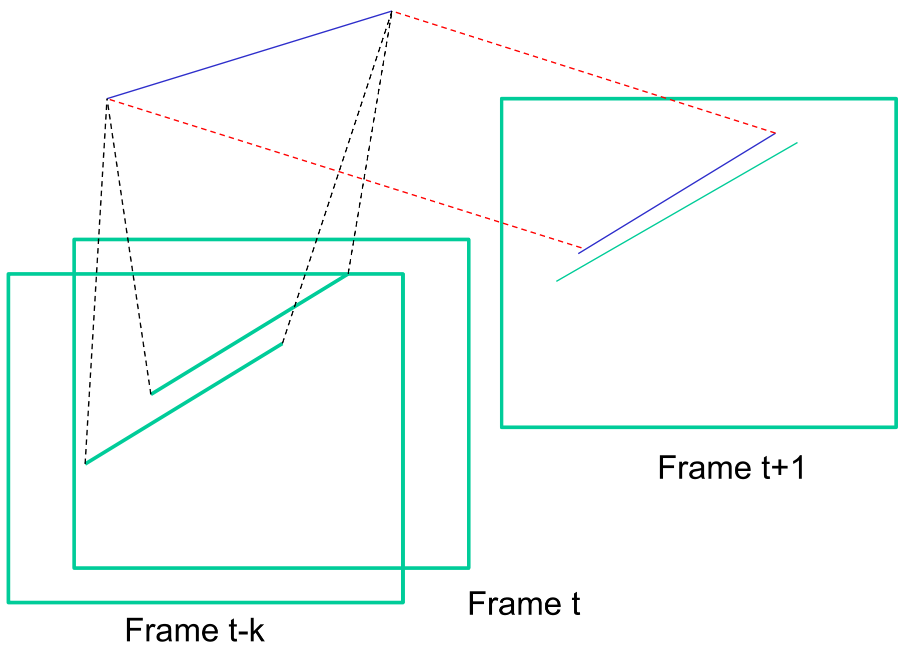


Figure 8.4: Lane triangulation and lane tracking. A lane is represented as a line segment with only two control points for simplicity. In frame $t-k$ and t, we find the supporting line segments and triangulate the control points to get the 3D lane. The 3D lane is then projected to the new frame $t+1$. In frame $t+1$, we found the supporting line segment (in green), and then updated the 3D lane according to it.

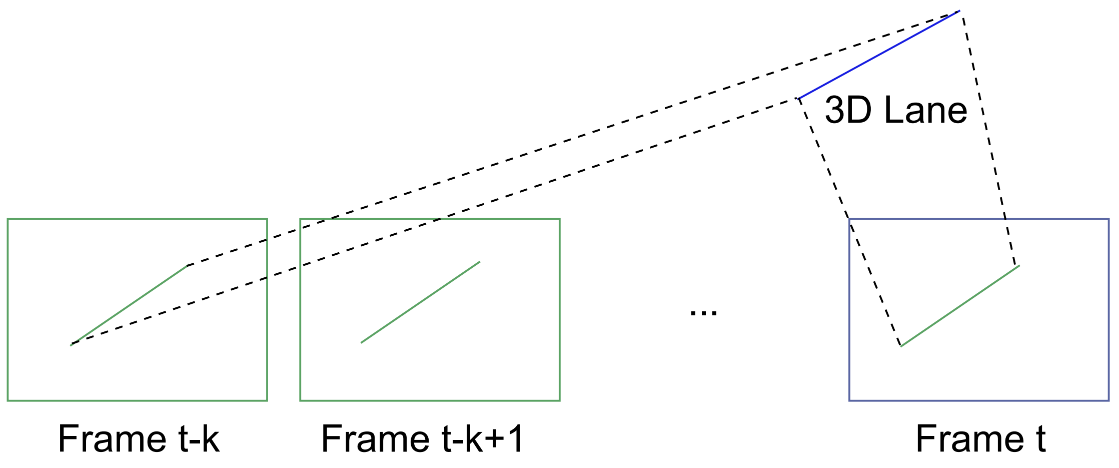


Figure 8.5: Line segment tracking and new lane adding. Starting from frame $t-k$, we track a line segment until a new key frame, frame t. In the frame t, we consider this line segment as a good candidate, since it has already been consistently tracked over k frames. A new 3D lane based on it is then added to the system.

8.5 Line Segment Tracking

When the new frames contain new lanes that have not been detected, we add these lanes in a keyframe. For robustness, we first track the line segments over $L = 3$ frames. If a line segment can be consistently tracked over L frames, we believe that there is a good lane associated with this line segment to be added.

The procedure is illustrated in Figure 8.5. Starting from frame $t-k$, we track a line segment until a new key frame, frame t. In the frame t, we consider this line segment as a good candidate, since it has already been consistently tracked over k frames. A new 3D lane based on it is then added to the system.

To track a line segment, in every new frame, we try to find the best matching line segment. This procedure is done by using epipolar geometry and computing NCC scores of the points from the source and the destination line segments, shown in Figure 8.6. Starting from a point within a set of points evenly distributed at the source line segment at frame $t-1$, we compute an epipolar line (in black) for it in frame t. Suppose there are some destination line segments in frame t that have intersections with the computed epipolar line. NCC scores are computed between the start point and these intersections. This step is repeated for every point from the set of points at the source

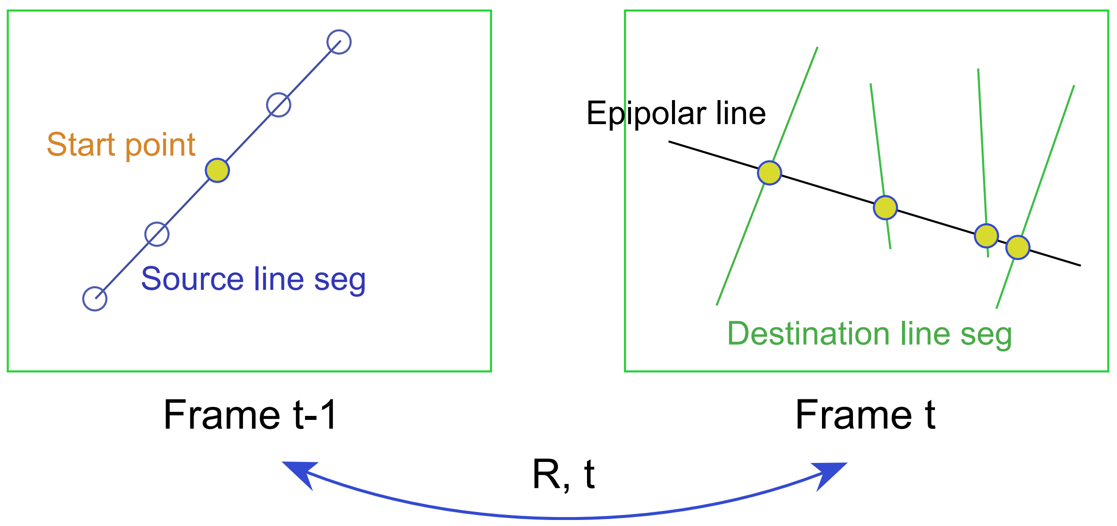


Figure 8.6: Line segment matching. Starting from a point (called the start point) within a set of points evenly distributed at the source line segment at frame $t-1$, we compute an epipolar line (in black) of it in frame t. Suppose there are some destination line segments in frame t that have intersections with the computed epipolar line. NCC scores are computed between the start point and these intersections. This step is repeated for every point from the set of points at the source line segment. NCC scores are accumulated for each destination line segment. At the end, the destination line segment that has the highest sum of the NCC scores is considered as the best match.

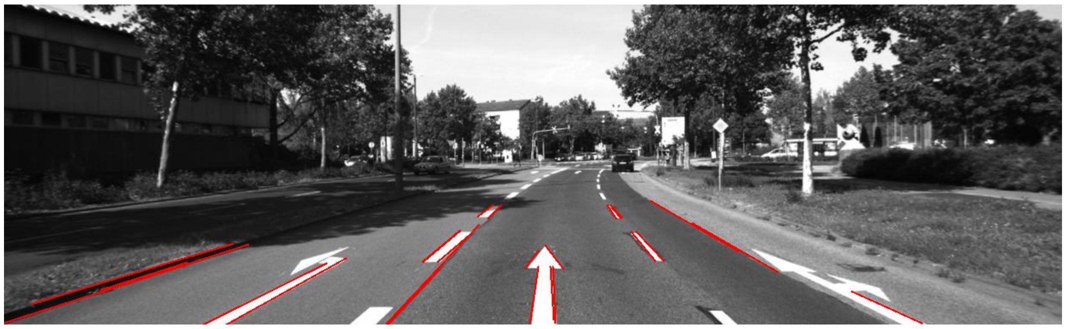
line segment. NCC scores are accumulated for each destination line segment. At the end, the destination line segment that has the highest sum of the NCC scores is considered the best match. In this matching step, the epipolar constraint is used, so the SFM pose cues are incorporated.

We continue tracking the line segment in the new frames. When a new keyframe comes, we add new lanes based on these well tracked line segments.

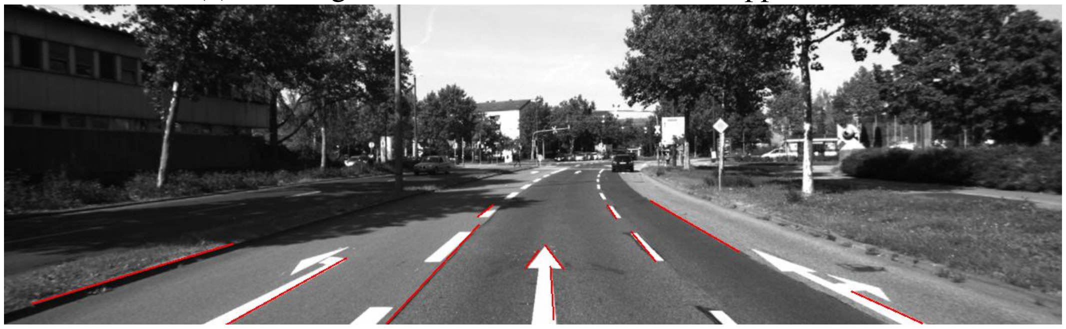
8.6 Lane Adding in a Keyframe

Line segments are tracked over L frames, then new lanes are created and added based on these tracks at a keyframe. In this section, we discuss the details of the adding step.

The first step is non-maximum suppression. This step reviews all the tracked



(a) Line segments before non-maximum suppression.



(b) Line segments after non-maximum suppression.

Figure 8.7: Line segment non-maximum suppression. This step reviews all the tracked line segments in the key frame, and combines similar line segments. After non-maximum suppression, we have only one longest line segment in the local region with similar orientation. (a) Line segments before non-maximum suppression. (b) Line segments after non-maximum suppression.

line segments in the key frame, and combines similar line segments. Here "similar" means two parallel and overlapped line segments. Note that the definition of "similar" is different from before. After non-maximum suppression, we have only one longest line segment in the local region with similar orientation, as shown in Figure 8.7.

The second step is lane selection and creation. In this step, we process all the line segments after non-maximum suppression, and create lanes based on them. We also try to link the two line segments, if their orientation difference is smaller than 5 degree and the distance between the end points of the line segments is smaller than 35 pixels. For these line segments, we consider them *similar* line segments. We create the new lane by adding all the end points of these line segments as control points. These control points are triangulated to 3D against the ground, then the 3D parabola curve is fitted to these 3D control points to represent a 3D lane.

The system continues maintaining these new added 3D lanes together with existing lanes as discussed in Section 8.4. In the next section, we discuss our lane detection results.

8.7 Experiments and Results

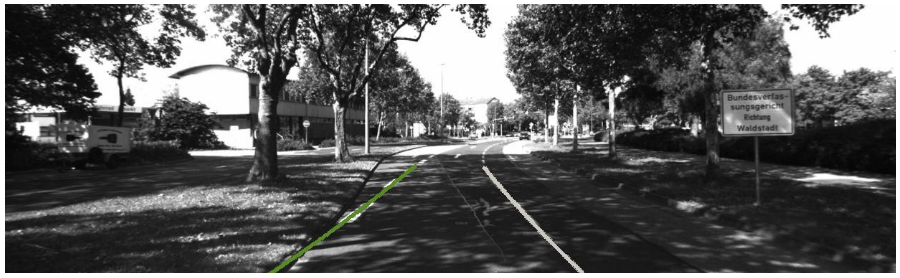
We present our lane detection results on the KITTI dataset. The results of Seq 0002 are shown visually in the Figure 8.8. As we can see, (a) Frame 11: Our system has good performance to confront interference, such as shadows. (b) Frame 47 and (c) Frame 79: Our system works well for normal road conditions. (d) Frame 131: Our system doesn't have the constraint that lanes have to be aligned with the direction of vehicle motion. Horizontal lanes are detected as well.

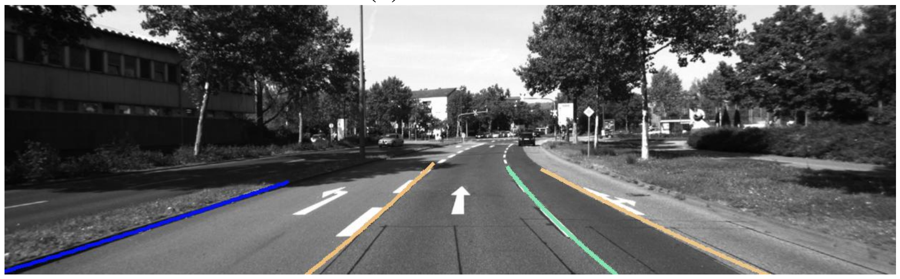
The results of Seq 0003 are shown in the Figure 8.9. (a) Frame 26: Our system has good performance to confront interference, such as other cars. (b) Frame 52: Our system works well for normal road conditions. (c) Frame 93 and (d) Frame 139: Our system can detect some false positives, but these false positives are not hard to eliminate by incorporating more cues, such as road/lane color and possible road region priors.

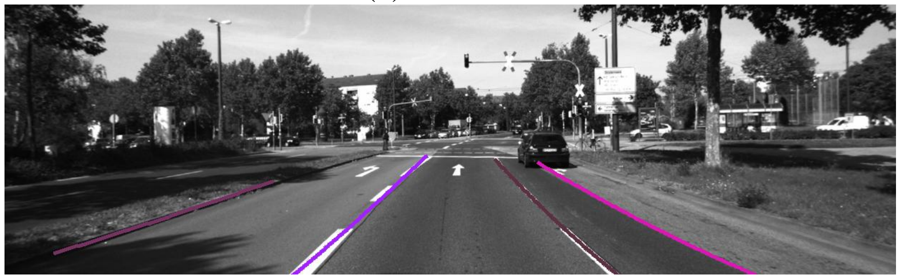
The results of Seq 0004 are shown in the Figure 8.10. (a) Frame 14: Our system doesn't have the constraint that a lane has to be aligned with the direction of the vehicle motion. Horizontal lanes are detected as well. (b) Frame 53: Our system works well in normal road conditions. (c) Frame 122 and (d) Frame 227: Our system has good performance to confront interference, such as shadows.

8.8 Conclusions

In this chapter, we presented a lane detection system incorporating our SFM poses and ground planes. It has been shown that accurate ground planes can improve the results of the bird-eye-view transform, thereby boosting the performance of the line segment extraction step accordingly. By incorporating the epipolar constraints computed from SFM poses, a line segment matching algorithm is introduced, which plays an important


(a) Frame 11.


(b) Frame 47.


(c) Frame 79.

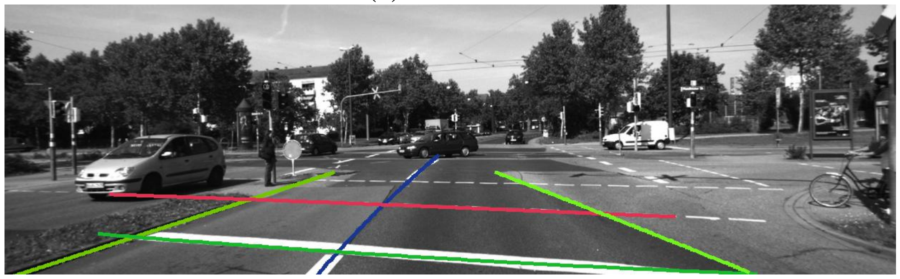
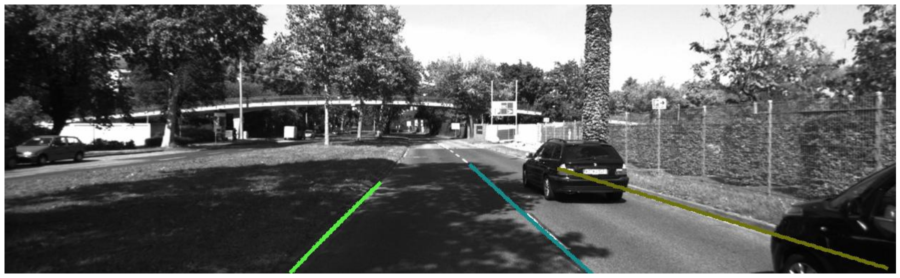
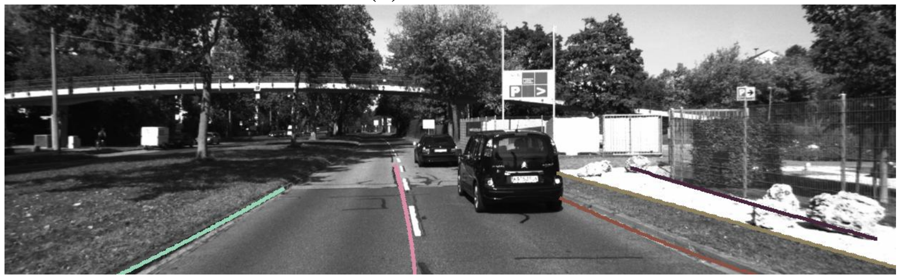

(d) Frame 131.

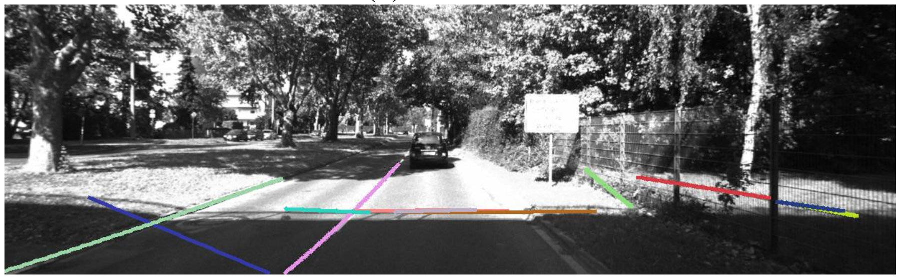
Figure 8.8: Lane detection results of KITTI Seq 0002. (a) Frame 11: Our system has good performance to confront interference, such as shadows. (b) Frame 47 and (c) Frame 79: Our system works well in normal road conditions. (d) Frame 131: Our system doesn't have the constraint that the lane has to be aligned with the direction of the vehicle motion. Horizontal lanes are detected as well.



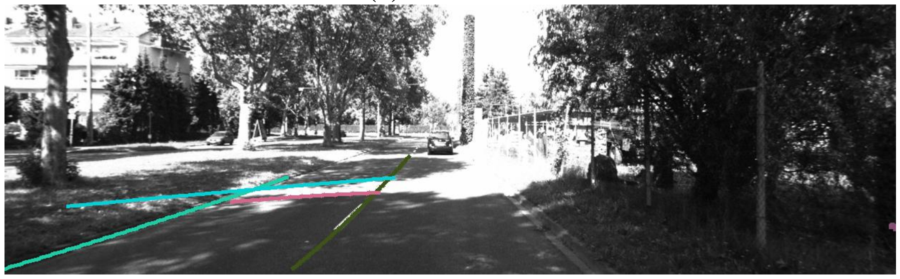
(a) Frame 26.



(b) Frame 52.

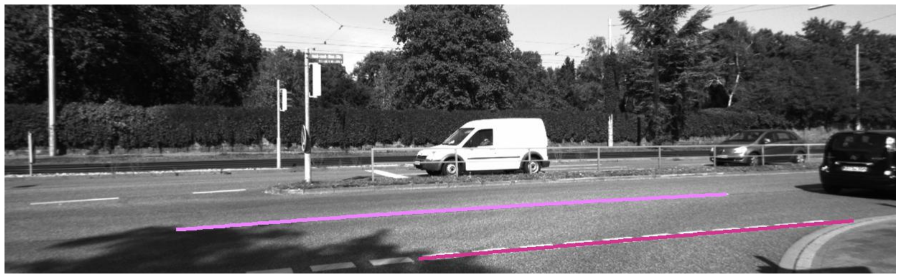


(c) Frame 93.

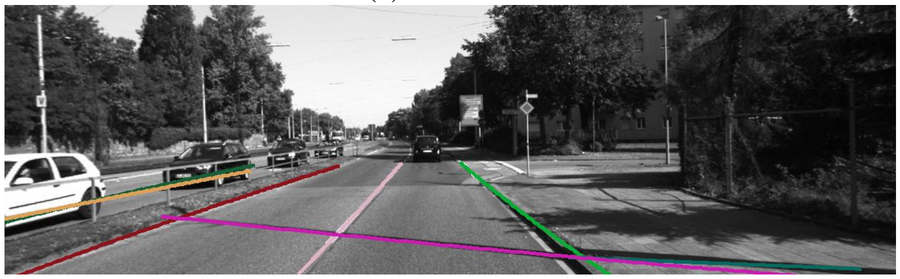


(d) Frame 139.

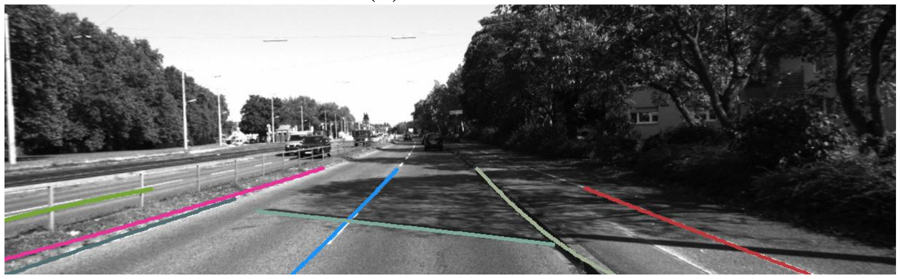
Figure 8.9: Lane detection results of KITTI Seq 0003. (a) Frame 26: Our system has good performance to confront interference, such as other cars. (b) Frame 52: Our system works well in normal road conditions. (c) Frame 93 and (d) Frame 139: Our system can detect some false positives, but these false positives are not hard to eliminate by incorporating more cues, such as road/lane color and possible road region priors.



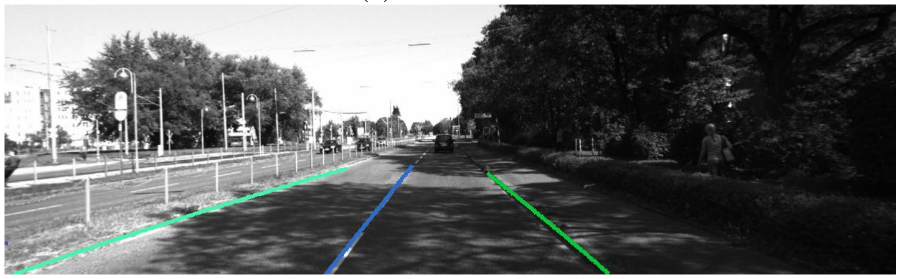
(a) Frame 14.



(b) Frame 53.



(c) Frame 122.



(d) Frame 227.

Figure 8.10: Lane detection results of KITTI Seq 0004. (a) Frame 14: Our system doesn't have the constraint that the lane has to be aligned with the direction of the vehicle motion. Horizontal lanes are detected as well. (b) Frame 53: Our system works well in normal road conditions. (c) Frame 122 and (d) Frame 227: Our system has good performance to confront interference, such as shadows.

role in the robustness of a lane detection system. Experimental results are shown using the KITTI benchmark dataset. Our system works well in normal road conditions, and also has good performance to confront interference, such as shadows, other cars and so on.

Chapter 9

Conclusions

Autonomous driving has become a hot topic in both academia and industry. The progress of autonomous car development in Google or other companies has become more and more promising. Although they are still far away from real consumer markets, people are warmly hoping autonomous driving techniques can become practical and affordable in the coming 10 years. Besides autonomous driving, there are also other related applications with lower barriers, such as collision avoidance, driver warning, and so on. All these applications can benefit from this work. In this section, we summarize our contributions and discuss future directions.

9.1 Discussion

This thesis has investigated structure from motion techniques for the application of autonomous driving. We have demonstrated that judicious multithreading can boost both the speed and accuracy of handling challenging road conditions. This is achieved through a novel per-frame epipolar search mechanism that generates redundantly validated 3D points persistent across long tracks and an efficient keyframe architecture to perform online thread-safe global bundle adjustment in parallel with pose computation. The novel epipolar search thread avoids delay, improves the quality of 3D points compared to prior approaches that only search and add 3D points when necessary, and also overcomes the challenge of fast moving imagery in the application of autonomous driving by searching in every frame. Additionally, the epipolar search works together with local bundle adjustment as a replacement for a more accurate, but expensive, multiview triangulation, which significantly improves the robustness and the accuracy. We have adaptive mechanisms for all types of typical SFM components, such as feature extraction, feature matching, keyframe or firewall adding. All these characteristics allow the system to outperform other state-of-the-art systems by large margins. The system is optimized to provide pose output in real-time at every frame, without delays for keyframe insertion or refinement. Our monocular visual odometry system performs nearly as well as stereo systems, which is attributable to robust correction of scale drift.

Our robust and accurate scale correction is a significant step in bridging the gap between monocular and stereo SFM. We believe this has great benefits for autonomous driving applications. We have demonstrated that the performance of real-time monocular SFM that uses our ground plane estimation is comparable to stereo on real-world driving sequences. In particular, we have shown that it is beneficial to include cues such as dense stereo and object bounding boxes for ground estimation, besides the traditional sparse features used in prior works. Further, we proposed a mechanism to combine those cues in a principled framework that reflects their per-frame relative confidences, as well as prior knowledge from training data. Significant improvement of the accuracy of ground plane estimation over prior works has been demonstrated. Not only does the ground plane estimate help scale correction of a monocular SFM system, but also it is a crucial component of a more ambitious traffic scene understanding system. Our accurate ground plane easily benefits existing 3D localization frameworks and vision-based lane detection for those applications, as also demonstrated by our experiments.

9.2 Future Directions

Our monocular SFM system provides accurate self-localization functionalities. But since all SFM systems localize incrementally, accumulation of error is inevitable. A clear direction to solve this issue is to integrate sensors, such as the global positioning system (GPS), or prior information, such as maps, into the system. GPS has already become standard equipment in modern cars. Map information is also accessible through Wi-Fi or 4G mobile network or local storage. A self-localization system combining a vision based SFM system with GPS or map information should be a practical solution for the autonomous driving application.

Based on state-of-the-art detection and tracking techniques, we demonstrated our monocular object localization framework through a highly accurate ground plane. This suppests a direction for a traffic scene understanding system. A traffic scene understanding system can detect and track cars, pedestrians, bicycles, trucks, buses and so on. Given their localizations, trajectory prediction becomes a trivial further step. Ever further, a scene understanding system could built upon these to accomplish all types of applications, such as safety and non-safety awareness, collision prediction and avoidance, and so on.

Last but not least, our final goal is to achieve a vision-based autonomous driving system, as opposed to the currently developed autonomous driving systems based on LIDAR (laser radar) that are hundreds of times more expensive . We believe a vision-based system is the final cost friendly solution for consumers in the future.

Appendix A

Appendix A

A.1 Essential Matrix Estimation

A essential matrix can be estimated from a set of corresponding image points by using the 8-point algorithm. Let $E = \hat{T}R$ be the essential matrix.

$$E = \begin{bmatrix} e_{11} & e_{12} & e_{13} \\ e_{21} & e_{22} & e_{23} \\ e_{31} & e_{32} & e_{33} \end{bmatrix}. \tag{A.1}$$

We can stack the 3×3 entries of a vector $E^s \in \mathbb{R}^9$:

$$E^s = [e_{11}, e_{21}, e_{31}, e_{12}, e_{22}, e_{32}, e_{13}, e_{23}, e_{33}]^T \in \mathbb{R}^9. \tag{A.2}$$

We further denote the *Kronecker product* $\otimes$ as:

$$A \otimes B \doteq \begin{bmatrix} a_{11}B & a_{12}B & \cdots & a_{1n}B \\ a_{21}B & a_{22}B & \cdots & a_{2n}B \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1}B & a_{m2}B & \cdots & a_{mn}B \end{bmatrix} \in \mathbb{R}^{mk \times nl}, \tag{A.3}$$

where $A \in \mathbb{R}^{m \times n}$ and $B \in \mathbb{R}^{k \times l}$.

Let $(\tilde{\mathbf{x}}_1^j, \tilde{\mathbf{x}}_2^j), j = 1, 2, \cdots, n$ be a set of corresponding image points in the homogeneous representation. Define:

$$\mathbf{a}^j \doteq \tilde{\mathbf{x}}_1^j \otimes \tilde{\mathbf{x}}_2^j. \tag{A.4}$$

Stack $\mathbf{a}^j$ as:

$$\chi \doteq [\mathbf{a}^1, \mathbf{a}^2, \cdots, \mathbf{a}^n]^T. \tag{A.5}$$

Ideally, the vector E^s satisfies

$$\chi E^s = 0. \tag{A.6}$$

At least 8 correspondences are needed to estimate an essential matrix. To compute the E^s that minimizes the least-squares error $\|\chi E^s\|^2$, Singular Value Decomposition (SVD) is used, and the E^s is the eigenvector of $\chi^T \chi$ that corresponds to its smallest eigenvalue.

In detail, let the SVD of χ be $\chi = U_\chi \Sigma_\chi V_\chi^T$. Define E^s to be the ninth column of V_χ, and unstack the entries of E^s into a 3×3 matrix, called E'_{est}. In general, this E'_{est} may not be in the essential space.

To project the estimated essential matrix into the essential space, one could compute the SVD of E'_{est} to be

$$E'_{\text{est}} = U \Sigma V^T. \tag{A.7}$$

In the ideal case, one of the diagonal elements of Σ should be zero, or at least small compared to the other two, which should be equal. So we set

$$\Sigma' = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}. \tag{A.8}$$

The optimized essential matrix $E_{\text{est}} = U \Sigma' V^T$.

A.2 Homography Matrix Estimation

A homography matrix can be estimated from a set of corresponding image points in a plane. Let $H = R + \frac{1}{d} T N^T \in \mathbb{R}^{3\times 3}$ be the planar homography matrix. We can stack the entries of H as a vector:

$$H^s = [H_{11}, H_{21}, H_{31}, H_{12}, H_{22}, H_{32}, H_{13}, H_{23}, H_{33}]^T \in \mathbb{R}^9. \tag{A.9}$$

Define

$$\chi = [\mathbf{a}^1, \mathbf{a}^2, \mathbf{a}^3, \cdots, \mathbf{a}^n]^T \in \mathbb{R}^{3n\times 9}, \tag{A.10}$$

where the matrix $\mathbf{a^j} = \tilde{\mathbf{x}}_1^j \otimes \hat{\tilde{\mathbf{x}}}_2^j$. $\tilde{\mathbf{x}}_1$ and $\tilde{\mathbf{x}}_2$ are the homogeneous representation of a pair of correspondences at the same plane in two images. In total, we have n correspondences. $\hat{\tilde{\mathbf{x}}}$ is the *cross product* matrix of the vector defined in Section 3.1.1. $\otimes$ is the *Kronecker product* defined in Section A.1.

Ideally, the vector H^s satisfies

$$\chi H^s = 0. \tag{A.11}$$

At least four correspondences are needed to compute a homography matrix. Let the SVD of χ be $\chi = U_\chi \Sigma_\chi V_\chi^T$. Define H^s to be the ninth column of V_χ, and unstack the entries of H^s into a 3×3 matrix, called H'_{est}.

Compute the SVD of H'_{est} to be

$$H'_{\text{est}} = U\Sigma V^T = U \begin{bmatrix} \sigma_1 & 0 & 0 \\ 0 & \sigma_2 & 0 \\ 0 & 0 & \sigma_3 \end{bmatrix} V^T. \tag{A.12}$$

We can normalize H'_{est} as $H_{\text{est}} = H'_{\text{est}}/\sigma_2$.

Bibliography

[AAWS11] M. Achtelik, M. Achtelik, S. Weiss, and R. Siegwart. Onboard imu and monocular vision based control for mavs in unknown in- and outdoor environments. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 3056–3063, May 2011.

[AFDM08] A. Angeli, D. Filliat, S. Doncieux, and J.-A. Meyer. Fast and incremental method for loop-closure detection using bags of visual words. *Robotics, IEEE Transactions on*, 24(5):1027–1037, Oct 2008.

[BBM09] T. Brox, C. Bregler, and J. Malik. Large displacement optical flow. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 41–48, June 2009.

[BDW06] Tim Bailey and H. Durrant-Whyte. Simultaneous localization and mapping (slam): part ii. *IEEE Robotics Automation Magazine*, 13(3):108–117, Sept 2006.

[BM92] P.J. Besl and Neil D. McKay. A method for registration of 3-d shapes. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 14(2):239–256, Feb 1992.

[BRGC10] T. Brox, B. Rosenhahn, J. Gall, and D. Cremers. Combined region and motion-based 3D tracking of rigid and articulated objects. *PAMI*, 32(3):402–415, March 2010.

[CFJS00] Alessandro Chiuso, Paolo Favaro, Hailin Jin, and Stefano Soatto. 3-d motion and structure from 2-d motion causally integrated over time: Implementation. In *Robotics and Automation, IEEE Transactions on*, pages 734–750, 2000.

[CGDM09] J. Civera, O.G. Grasa, A.J. Davison, and J. M M Montiel. 1-point ransac for EKF-based structure from motion. In *Intelligent Robots and Systems, IEEE International Conference on*, pages 3498–3504, Oct 2009.

[CGDM10] Javier Civera, Oscar G. Grasa, Andrew J. Davison, and J. M. M. Montiel. 1-point ransac for extended Kalman filtering: Application to real-time structure from motion and visual odometry. *Journal of Field Robot*, 27(5):609–631, September 2010.

[CLFP10] Brian Clipp, Jongwoo Lim, Jan-Michael Frahm, and Marc Pollefeys. Parallel, real-time visual SLAM. In *IROS*, pages 3961–3968, 2010.

[CMM05] Yang Cheng, M. Maimone, and L. Matthies. Visual odometry on the mars exploration rovers. In *Systems, Man and Cybernetics, 2005 IEEE International Conference on*, volume 1, pages 903–910 Vol. 1, Oct 2005.

[CN08] Mark Cummins and Paul Newman. Fab-map: Probabilistic localization and mapping in the space of appearance. *The International Journal of Robotics Research*, 27(6):647–665, 2008.

[CS10] Wongun Choi and Silvio Savarese. Multi-target tracking in world coordinate with single, minimally calibrated camera. In *ECCV*, pages 553–567, 2010.

[Dav03] A.J. Davison. Real-time simultaneous localisation and mapping with a single camera. In *ICCV*, pages 1403–1410, Oct 2003.

[DG09] Gijs Dubbelman and Frans Groen. Bias reduction for stereo motion estimation with applications to large scale visual odometry. In *CVPR*, 2009.

[DH72] Richard O. Duda and Peter E. Hart. Use of the hough transformation to detect lines and curves in pictures. *Communications of the ACM*, 15(1):11–15, January 1972.

[DM98] Leonardo Dagum and Ramesh Menon. Openmp: an industry standard api for shared-memory programming. *Computational Science Engineering, IEEE*, 5(1):46–55, 1998.

[DRMS07] Andrew J. Davison, Ian D. Reid, Nicholas D. Molton, and Olivier Stasse. MonoSLAM: Real-time single camera SLAM. *PAMI*, 29(6):1052–1067, 2007.

[DWB06] H. Durrant-Whyte and Tim Bailey. Simultaneous localization and mapping: part i. *IEEE Robotics Automation Magazine*, 13(2):99–110, June 2006.

[ELSVG09] A. Ess, B. Leibe, K. Schindler, and L. Van Gool. Robust multiperson tracking from a mobile platform. *PAMI*, 31(10):1831–1846, 2009.

[FB81a] 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.

[FB81b] Martin A. Fischler and Robert C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, June 1981.

[FEN07] F. Fraundorfer, C. Engels, and D. Nister. Topological mapping, localization and navigation using image collections. In *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, pages 3872–3877, Oct 2007.

[FGMR10] Pedro F. Felzenszwalb, Ross B. Girshick, David McAllester, and Deva Ramanan. Object detection with discriminatively trained part-based models. *PAMI*, 32(9):1627–1645, 2010.

[FL88] O. D. Faugeras and F. Lustman. Motion and Structure From Motion in a Piecewise Planar Environment. *Pat. Rec. AI*, 2(3):485–508, 1988.

[FS12] F. Fraundorfer and D. Scaramuzza. Visual odometry: Part ii: Matching, robustness, optimization, and applications. *IEEE Robotics Automation Magazine*, 19(2):78–90, June 2012.

[FWmFP08] Friedrich Fraundorfer, Changchang Wu, Jan michael Frahm, and Marc Pollefeys. Visual word based location recognition in 3d models using distance augmented weighting. In *In Fourth International Symposium on 3D Data Processing, Visualization and Transmission*, 2008.

[GKS$^+$10] G. Grisetti, R. Kummerle, C. Stachniss, U. Frese, and C. Hertzberg. Hierarchical optimization on manifolds for online 2d and 3d mapping. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 273–278, May 2010.

[GLU12] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? The KITTI vision benchmark suite. In *CVPR*, 2012.

[GLW$^+$14] A. Geiger, M. Lauer, C. Wojek, C. Stiller, and R. Urtasun. 3D traffic scene understanding from movable platforms. *PAMI*, 2014.

[GZS11] Andreas Geiger, Julius Ziegler, and Christoph Stiller. StereoScan: Dense 3D reconstruction in real-time. In *IEEE Int. Veh. Symp.*, 2011.

[Har97] Richard Hartley. In defense of the eight-point algorithm. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 19(6):580–593, Jun 1997.

[HARB11] P. Hansen, H. Alismail, P. Rander, and B. Browning. Monocular visual odometry for robot localization in lng pipes. In *Proceedings of Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 3111–3116, May 2011.

[HCSD10] A. Handa, M. Chli, H. Strasdat, and A.J. Davison. Scalable active matching. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 1546–1553, June 2010.

[HLON91] R.M. Haralick, D. Lee, K. Ottenburg, and M. Nolle. Analysis and solutions of the three point perspective pose estimation problem. In *Computer Vision and Pattern Recognition, 1991. Proceedings CVPR '91., IEEE Computer Society Conference on*, pages 592–598, Jun 1991.

[Hou59] P.V.C. Hough. Machine Analysis Of Bubble Chamber Pictures. In *2nd International Conference On High-Energy Accelerators*, 1959.

[How08] A. Howard. Real-time stereo visual odometry for autonomous ground vehicles. In *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pages 3946–3952, Sept 2008.

[HP88] C. G. Harris and J. M. Pike. 3d positional integration from image sequences. *Image Vision Comput.*, 6(2):87–90, May 1988.

[HS88] Chris Harris and Mike Stephens. A combined corner and edge detector. In *In Proc. of Fourth Alvey Vision Conference*, pages 147–151, 1988.

[Jaz70] Andrew H. Jazwinski. *Stochastic Processes and Filtering Theory*. Academic Press, April 1970.

[JL00] M. Jogan and A. Leonardis. Robust localization using panoramic view-based recognition. In *Pattern Recognition, 2000. Proceedings. 15th International Conference on*, volume 4, pages 136–139 vol.4, 2000.

[KKJ11] Abhijit Kundu, K Madhava Krishna, and C. V. Jawahar. Realtime multibody visual slam with a smoothly moving monocular camera. In *ICCV*, pages 2080–2087, 2011.

[KM07] Georg Klein and David Murray. Parallel tracking and mapping for small AR workspaces. In *ISMAR*, 2007.

[KM08] Georg Klein and David Murray. Improving the agility of keyframe-based SLAM. In *ECCV*, 2008.

[KRC$^+$11] Bernd Manfred Kitt, Joern Rehder, Andrew D Chambers, Miriam Schonbein, Henning Lategahn, and Sanjiv Singh. Monocular visual odometry using a planar road model to solve scale ambiguity. In *Proceedings of European Conference on Mobile Robots*, Sep 2011.

[KRE60] Kalman, Rudolph, and Emil. A New Approach to Linear Filtering and Prediction Problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):35–45, 1960.

[KSC13] C. Kerl, J. Sturm, and D. Cremers. Robust odometry estimation for RGB-D cameras. In *ICRA*, pages 3748–3754, 2013.

[KSD$^+$09] Rainer Kümmerle, Bastian Steder, Christian Dornhege, Michael Ruhnke, Giorgio Grisetti, Cyrill Stachniss, and Alexander Kleiner. On measuring the accuracy of SLAM algorithms. *Autonomous Robots*, 27(4):387–407, 2009.

[LA09] M.I. A. Lourakis and A.A. Argyros. SBA: A Software Package for Generic Sparse Bundle Adjustment. *ACM Trans. Math. Soft.*, 36(1):1–30, 2009.

[LH87] H. C. Longuet-Higgins. A computer algorithm for reconstructing a scene from two projections. In M. A. Fischler and O. Firschein, editors, *Readings in Computer Vision: Issues, Problems, Principles, and Paradigms*, pages 61–62. Kaufmann, Los Altos, CA., 1987.

[LK81] Bruce D. Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of the 7th International Joint Conference on Artificial Intelligence*, IJCAI'81, pages 674–679, San Francisco, CA, USA, 1981. Morgan Kaufmann Publishers Inc.

[LKSV07] Ting Li, V. Kallem, D. Singaraju, and R. Vidal. Projective factorization of multiple rigid-body motions. In *CVPR*, pages 1–6, June 2007.

[LMCG99] S. Lacroix, A. Mallet, R. Chatila, and L. Gallo. Rover Self Localization in Planetary-Like Environments. In M. Perry, editor, *Artificial Intelligence, Robotics and Automation in Space*, volume 440 of *ESA Special Publication*, page 433, August 1999.

[LMNF09] 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.

[Low04] David G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2):91–110, 2004.

[MCM07] Mark Maimone, Yang Cheng, and Larry Matthies. Two years of visual odometry on the mars exploration rovers. *Journal of Field Robotics*, 24(3):169–186, 2007.

[MGK00] J. Matas, C. Galambos, and J. Kittler. Robust detection of lines using the progressive probabilistic hough transform. *Computer Vision and Image Understanding*, 78(1):119 – 137, 2000.

[ML09] Marius Muja and David G. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. In *VISSAPP*, pages 331–340. INSTICC Press, 2009.

[ML12] Marius Muja and David G. Lowe. Fast matching of binary features. In *Computer and Robot Vision (CRV)*, pages 404–410, 2012.

[MLD$^+$06] E. Mouragnon, M. Lhuillier, M. Dhome, F. Dekeyser, and P. Sayd. Real time localization and 3d reconstruction. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 1, pages 363–370, 2006.

[Mor80] Hans Peter Moravec. *Obstacle Avoidance and Navigation in the Real World by a Seeing Robot Rover*. PhD thesis, Stanford University, Stanford, CA, USA, 1980. AAI8024717.

[MS87] L. Matthies and S.A. Shafer. Error modeling in stereo navigation. *Robotics and Automation, IEEE Journal of*, 3(3):239–248, 1987.

[MS06] A. Milella and R. Siegwart. Stereo-based ego-motion estimation using pixel tracking and iterative closest point. In *Computer Vision Systems. IEEE International Conference on*, pages 21–21, Jan 2006.

[MSKS03] Yi Ma, Stefano Soatto, Jana Kosecka, and S. Shankar Sastry. *An Invitation to 3-D Vision: From Images to Geometric Models*. SpringerVerlag, 2003.

[NCH06] P. Newman, D. Cole, and K. Ho. Outdoor slam using visual appearance and laser ranging. In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pages 1180–1187, May 2006.

[Nis04] David Nistér. An efficient solution to the five-point relative pose problem. *PAMI*, 26(6):756–777, 2004.

[NNB04] David Nistér, Oleg Naroditsky, and James Bergen. Visual odometry. In *CVPR*, pages 652–659, 2004.

[NNB06] David Nistér, Oleg Naroditsky, and James Bergen. Visual odometry for ground vehicle applications. *Journal of Field Robotics*, 23(1):3–20, 2006.

[NS06] David Nister and Henrik Stewenius. Scalable recognition with a vocabulary tree. In *Proceedings of Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, CVPR, pages 2161–2168, Washington, DC, USA, 2006. IEEE Computer Society.

[Oli05] John Oliensis. The least-squares error for structure from infinitesimal motion. *IJCV*, 61(3):259–299, 2005.

[OLT06] E. Olson, J. Leonard, and S. Teller. Fast iterative alignment of pose graphs with poor initial estimates. In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pages 2262–2269, May 2006.

[OMSM00] C.F. Olson, L.H. Matthies, M. Schoppers, and M.W. Maimone. Robust stereo ego-motion for long distance navigation. In *Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on*, volume 2, pages 453–458 vol.2, 2000.

[OMSM01] C.F. Olson, L.H. Matthies, M. Schoppers, and M.W. Maimone. Stereo ego-motion improvements for robust rover navigation. In *ICRA*, volume 2, pages 1099–1104, 2001.

[OMSM03] Clark F. Olson, Larry H. Matthies, Marcel Schoppers, and Mark W. Maimone. Rover navigation using stereo ego-motion. *Robotics and Autonomous Systems*, 43(4):215 – 229, 2003.

[OSG10] Kemal Egemen Ozden, Konrad Schindler, and Luc Van Gool. Multibody structure-from-motion in practice. *PAMI*, 32(6):1134–1141, 2010.

[OSVG07] K.E. Ozden, K. Schindler, and L. Van Gool. Simultaneous segmentation and 3D reconstruction of monocular image sequences. In *ICCV*, pages 1–8, 2007.

[PMP11] A. Pretto, E. Menegatti, and E. Pagello. Omnidirectional dense large-scale mapping and navigation based on meaningful triangulation. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 3289–3296, May 2011.

[PRF11] H. Pirsiavash, D. Ramanan, and C.C. Fowlkes. Globally-optimal greedy algorithms for tracking a variable number of objects. In *CVPR*, 2011.

[QCG$^+$09] Morgan Quigley, Ken Conley, Brian P. Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y. Ng. Ros: an open-source robot operating system. In *ICRA Workshop on Open Source Software*, 2009.

[RPD10] Edward Rosten, Reid Porter, and Tom Drummond. FASTER and better: A machine learning approach to corner detection. *PAMI*, 32:105–119, 2010.

[RRKB11a] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. Orb: An efficient alternative to sift or surf. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2564–2571, Nov 2011.

[RRKB11b] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. ORB: An efficient alternative to SIFT or SURF. In *ICCV*, pages 2564 –2571, 2011.

[SBK10] Narayanan Sundaram, Thomas Brox, and Kurt Keutzer. Dense point trajectories by GPU-accelerated large displacement optical flow. In *ECCV*, pages 438–451, 2010.

[SC14] Shiyu Song and Manmohan Chandraker. Robust scale estimation in real-time monocular sfm for autonomous driving. In *Computer Vision and Pattern Recognition, IEEE International Conference on*, Columbus, Ohio, USA, June 2014.

[SCG13] Shiyu Song, Manmohan Chandraker, and Clark C. Guest. Parallel, real-time monocular visual odometry. In *ICRA*, 2013.

[SF11] D. Scaramuzza and F. Fraundorfer. Visual odometry: Part i - the first 30 years and fundamentals. *IEEE Robotics Automation Magazine*, 18(4):80–92, 2011.

[SFPS09] D. Scaramuzza, F. Fraundorfer, M. Pollefeys, and R. Siegwart. Absolute scale in structure from motion from a single vehicle mounted camera by exploiting nonholonomic constraints. In *ICCV*, pages 1413–1419, 2009.

[SFS09] D. Scaramuzza, F. Fraundorfer, and R. Siegwart. Real-time monocular visual odometry for on-road vehicles with 1-point ransac. In *Robotics and Automation, 2009. ICRA '09. IEEE International Conference on*, pages 4293–4299, May 2009.

[SMD10a] H. Strasdat, J. M. M. Montiel, and A. Davison. Scale drift-aware large scale monocular slam. In *Proceedings of Robotics: Science and Systems*, Zaragoza, Spain, June 2010.

[SMD10b] H. Strasdat, J. M M Montiel, and A.J. Davison. Real-time monocular slam: Why filter? In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 2657–2664, May 2010.

[SMD12] Hauke Strasdat, J.M.M. Montiel, and Andrew J. Davison. Visual slam: Why filter? *Image and Vision Computing*, 30(2):65 – 77, 2012.

[SS08] D. Scaramuzza and R. Siegwart. Appearance-guided monocular omnidirectional visual odometry for outdoor ground vehicles. *Robotics, IEEE Transactions on*, 24(5):1015–1026, 2008.

[ST94a] Jianbo Shi and Carlo Tomasi. Good features to track. In *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR '94., 1994 IEEE Computer Society Conference on*, pages 593–600, Jun 1994.

[ST94b] Jianbo Shi and Carlo Tomasi. Good features to track. In *CVPR*, pages 593–600, 1994.

[SUW06] Konrad Schindler, James U, and Hanzi Wang. Perspective n-view multi-body structure-and-motion through model selection. In *ECCV*, volume 3951, pages 606–619, 2006.

[TMD+06] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann, K. Lau, C. Oakley, M. Palatucci, V. Pratt, P. Stang, S. Strohband, C. Dupont, L.-E. Jendrossek, C. Koelen, C. Markey, C. Rummel, J. van Niekerk, E. Jensen, P. Alessandrini, G. Bradski, B. Davies, S. Ettinger, A. Kaehler, A. Nefian, and P. Mahoney. Winning the darpa grand challenge. *Journal of Field Robotics*, 2006.

[TMHF00] Bill Triggs, Philip Mclauchlan, Richard Hartley, and Andrew Fitzgibbon. Bundle adjustment - a modern synthesis. In *Vision Algorithms: Theory and Practice, LNCS*, pages 298–375. Springer Verlag, 2000.

[TPD08] Jean-Philippe Tardif, Yanis Pavlidis, and Kostas Daniilidis. Monocular visual odometry in urban environments using an omnidirectional camera. In *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pages 2531–2538, Sept 2008.

[UN00] I. Ulrich and I. Nourbakhsh. Appearance-based place recognition for topological localization. In *Robotics and Automation, 2000. Proceedings. ICRA '00. IEEE International Conference on*, volume 2, pages 1023–1029 vol.2, 2000.

[WACS11] Changchang Wu, S. Agarwal, B. Curless, and S.M. Seitz. Multicore bundle adjustment. In *CVPR*, pages 3057–3064, June 2011.

[WPB+08] A. Wedel, T. Pock, J. Braun, U. Franke, and D. Cremers. Duality tv-l1 flow with fundamental matrix prior. In *IVCNZ*, pages 1–6, Nov 2008.

[WSS11] Stephan Weiss, Davide Scaramuzza, and Roland Siegwart. Monocular-slam-based navigation for autonomous micro helicopters in gps-denied environments. *Journal of Field Robotics*, 28(6):854–874, 2011.

[WWR+13] Christian Wojek, Stefan Walk, Stefan Roth, Konrad Schindler, and Bernt Schiele. Monocular visual scene understanding: Understanding multi-object traffic scenes. *PAMI*, 35(4):882–897, 2013.

[YMU13] K. Yamaguchi, D. McAllester, and R. Urtasun. Robust monocular epipolar flow estimation. In *CVPR*, pages 1862–1869, June 2013.

[ZPB07] C. Zach, T. Pock, and H. Bischof. A duality based approach for realtime TV-L1 optical flow. In *DAGM on Pattern Recognition*, pages 214–223, 2007.