Revisiting Monocular Visual Odometry from Downward Facing Cameras

Thesis

Presented in Partial Fulfillment of the Requirements for the Degree Master of Science in the Graduate School of The Ohio State University

By

Daniel Hicks Walton III, B.S.

Graduate Program in Electrical and Computer Engineering

The Ohio State University

2023

Thesis Committee

Prof. Keith A. Redmill, Advisor

Prof. Umit A. Ozguner, Committee Member

Abstract

Visual Odometry has been a popular research topic for the last 40 years. Odometry alone has shown robustness across different sensor modalities including LiDAR and vision. In the automated vehicle setting, VO can provide an alternative to sensors like GPS that are prone to large errors in the presence of urban canyons and poor weather. Recent approaches focus on deep learning and the estimation of uncertainty from forward-facing cameras. In this research, Visual Odometry from downward-facing cameras is revisited to produce a simple yet robust alternative to forward-facing methods. Visual Odometry from downward-facing cameras has applications across many robotic platforms including UAVs, AUVs, and UGVs due to the relative location of prominent features in their respective environments. However, in the automated vehicle setting, this is particularly challenging due to severe motion blur, variable lighting conditions, and varying texture. In this thesis, a comprehensive review of odometry approaches is discussed. Several approaches for estimating a camera's pose from a downward-facing camera were explored. And finally, a geometric Visual Odometry pipeline from a downward-facing camera is proposed.

Dedication

This thesis is dedicated to my loved ones for their support throughout this process.

Acknowledgments

I would like to thank my family and loved ones for supporting me throughout this process and in my personal life. I also would like to thank my advisor, Professor Keith Redmill, for his guidance and support throughout this process. I would also like to thank Professor Umit Ozguner and Dr. Ekim Yurtsever for their support and feedback on my research. I would also like to thank the College of Engineering, and the numerous mentors I've had the privilege of interacting with over the last several years. Lastly, I would like to thank my lab mates for their consistent support and feedback over the last couple of years.

Vita

April 1997 ……... Born – Columbus, Ohio

May 2015 …… Westerville North High School

December 2020 …… B.S. Electrical and Computer Engineering,
The Ohio State University

January 2021 to January 2022 ……. Discovery Scholars Program Fellow,
The Ohio State University

January 2022 to Present ……... Graduate Research Associate,
The Center for Automotive Research,
The Ohio State University

Fields of Study

Major Field: Electrical and Computer Engineering

Table of Contents

Abstract ii
Dedication iii
Acknowledgments iv
Vita v
Table of Contents vi
List of Tables viii
List of Figures ix
Chapter 1. Introduction 1
1.1 Background 1
1.2 Motivation 2
1.3 Contributions 3
Chapter 2. Vehicle Localization Literature Review 4
2.1 Vehicle Localization 4
2.1.1 Wheel Odometry 5
2.1.2 Inertial Odometry 6
2.1.3 Laser Odometry 7
2.1.4 Radar Odometry 8
2.1.5 Visual Odometry 9
Chapter 3. Visual Odometry Fundamentals 10
3.1 Advantages and Disadvantages 12
3.2 Visual Odometry, Structure from Motion, and SLAM 16
3.3 Classical Visual Odometry Methods 19
3.3.1 Camera Model 20
3.3.2 Feature Detection and Matching 24
3.3.3 Outlier Rejection 30
3.3.4 Motion Estimation 32
3.3.5 Optimization 36
3.3.6 Notable Geometric Visual Odometry and SLAM Methods 37
3.4 Direct Visual Odometry Methods 41

3.5 Deep Learning-based Visual Odometry Methods 44
3.6 Downward-Facing Camera Methods 45
Chapter 4. Feature Extraction Experiments 51
4.1 Problem Setup 52
4.1.1 Carla Simulator Dataset 52
4.1.2 Oxford Robotcar Dataset 53
4.1.3 KITTI Dataset 54
4.2 Phase 1 Experiments 55
4.2.1 Experiment 1a: Feature Detection and Matching 56
4.2.1.1 Problem Formulation 56
4.2.1.2 Experiment Results 57
4.2.2 Experiment 1b: Sparse and Dense Optical Flow 60
4.2.2.1 Problem Formulation 60
4.2.2.2 Experiment Results 63
4.2.3 Experiment 1c: Power Spectrum Analysis 66
4.2.3.1 Problem Formulation 66
4.2.3.2 Experiment Results 68
4.2.4 Experiment 1d: End-to-End Deep Learning 70
4.2.4.1 Problem Formulation 70
4.2.4.2 Experiment Results 73
4.2.5 Experiment 1e: Motion from Blur 76
4.2.5.1 Problem Formulation 76
4.2.5.2 Experiment Results 79
Chapter 5. Deep Learning based Feature Detection and Matching 89
5.1 Problem Formulation 89
5.2 Experiment Results 93
Chapter 6. Conclusion and Future Work 96
6.1 Conclusion 96
6.2 Future Work 98
Bibliography 99

List of Tables

Table 1. The results for different feature extraction, description, and matching pipelines from classic and a more modern method. The left-hand side has the statistics for the unfiltered features and the right-hand side has the statistics for the inliers. BF: Brute Force Matcher. 59

Table 2. The final proposed network architecture. 74

Table 3. The set of Gaussian basis filters, G_{2x} and steerable kernels, k_x [118]..... 78

List of Figures

Figure 1. An image of the electric cart used by Hans Moravec to conduct his initial visual odometry experiments [31]. 11

Figure 2. (a) An image of the NAVLAB test vehicle at Carnegie Mellon for their work on ALVINN and (b) the proposed neural network architecture for the pose estimation [2]. 12

Figure 3. Different camera modalities for VO pipelines and examples of their respective data: (a) RGB, (b) RGB-D (c) Stereo, (d) Event, (e) Infrared, (f) Omnidirectional, and (g) Multi-View Stereo [34-40]. 14

Figure 4. An example of the data output for the (a) Structure from Motion, (b) Visual Odometry, and (c) Simultaneous Localization and Mapping problems [50, 18, 28]..... 18

Figure 5. A high-level view of classical VO pipelines. 19

Figure 6. A basic model of a modern camera with focal length (f), 3D observation (P), depth (z), projected point (p'), and distance from lens to projection plane (z'). The light rays observed from 3D observations converge at point F on the principal axis [17]. 21

Figure 7. The Pinhole Camera Model used in many computer vision tasks. [47]. 22

Figure 8. The local feature extraction approach of the well-known FAST feature detector [64]. 27

Figure 9. A basic depiction of the epipolar geometry used in classical VO pipelines [47]. 33

Figure 10. A high-level view of the ORB-SLAM architecture as seen in [18]. 39

Figure 11. A depiction of the bearing vectors, f in the Normal Epipolar Constraint (NEC) and Probabilistic Normal Epipolar Constraint (PNEC) work. As seen in [87]. 40

Figure 12. The high-level VSLAM architecture for LSD-SLAM as seen in [22]. 43
Figure 13. The vehicle setup in the Carla Simulator. [112]. 53
Figure 14. A diagram of the vehicle sensor setup in the Oxford Robotcar Dataset that is relevant for this research [35]. 54
Figure 15. A diagram of the sensor setup for the forward-facing camera in the KITTI Dataset and the virtually warped downward-facing camera [90,113]. 55
Figure 16. An example failure mode for feature extraction in the Carla Dataset. 60
Figure 17. The Lucas-Kanade Optical Flow method with Shi-Tomasi Corners: (a) Estimated Trajectory (b) Ground Truth on Carla Town 03. 64
Figure 18. A pair of images for the dense optical flow input. 65
Figure 19. The magnitude of the optical flow (left) and the angle (right) for the image. Dark blue represents small magnitude or angle. Bright green represents large magnitude or angle..... 66
Figure 20. The histogram for the magnitude (a) and angle (b) of the optical flow. 66
Figure 21. (a) A diagram of the drone and camera setup. (b) An image of the cross-power spectrum for an image patch pair [92]. 68
Figure 22. A sample image pair used from the Carla Dataset. 69
Figure 23. A sample image pair with power spectrum magnitude displayed. 69
Figure 24. (a) The expected inverse FFT of the power spectrum. (b) The output for many of the inverse FFTs. 70
Figure 25. A diagram of the architecture in [106] for their unsupervised camera pose network from a downward-facing camera. The EarlyBird network uses two image pairs

to regress camera pose, and the SlowBird network uses five image pairs to regress camera pose. 72
Figure 26. A diagram of DeepVO as outlined in [25]. 73
Figure 27. Qualitative results for the proposed camera pose network. On the left is the Carla Ground Truth, and on the right is the predicted model trajectory. 75
Figure 28. A diagram of the motion from blur baseline illustrated in [118]. 79
Figure 29. The sample image presented in [118] that is used for the baseline experiment. 80
Figure 30. Baseline experiment results for patch-wise and whole image optical flow estimation. Results are consistent with the results in [118]. 81
Figure 31. A sample image from the Oxford Robotcar Dataset warped downward. 82
Figure 32. Baseline experiment results for the Oxford Robotcar Dataset. 83
Figure 33. Artificially blurred Carla Images with the minimum of their cepstrum shown. 84
Figure 34. The results of synthetically blurring 1000 Carla images from a downward-facing camera and measuring the frame's optical flow from motion blur. (a) Magnitude Estimation and (b) Angle Estimation Results. 86
Figure 35. A sample subsequence ran on the Oxford Robotcar Dataset, ... 87
Figure 36. Basic attention layer for transformers [56,123]. 90
Figure 37. Overall architecture of the LoFTR framework proposed in [56]. This approach is used to match features in the geometric VO front end of this thesis research. 91

Figure 38. An estimated trajectory for the Oxford Robotcar Dataset, Sequence: 2015-10-30-13-52-14..... 95

Chapter 1. Introduction

1.1 Background

Over the last few decades, the robotics industry and automated vehicles have grown in popularity. Ideas of autonomous robots and self-driving cars have been ever present in science fiction mediums for nearly a century [1]. However, real world manifestations of these ideas have not been present until recently. In the early 2000s, the Defense Advanced Research Projects Agency (DARPA) initiated a sequence of challenges to bring self-driving technologies to the military [3]. The DARPA Grand Challenges of 2004 and 2005 (and later the DARPA Urban Challenge in 2007) led to several prominent business ventures for the development of autonomous vehicles in a consumer setting. However, today there remain challenges to bringing these technologies to market.

Autonomous vehicles from a systems perspective are typically designed under 3 major paradigms: Perception and Localization, Motion Planning and Decision Making, and Control. Vehicle perception systems seek to detect, track, and glean high level knowledge from obstacles and a traversable path in their environment. Localization pipelines often utilize high-definition maps and a sensor suite to localize the vehicle

globally and locally for vehicle control and motion planning actions. Poorly localized vehicles can be extremely dangerous for passengers and the public.

It is now commonplace for consumer vehicles to have GPS, cameras, and radar for Advanced Driver Assistance Systems (ADAS). For more advanced automated vehicles like robotaxis, these sensors can also include LiDAR and high-end inertial sensors. GPS technology is essential for self-driving technology. Even with the knowledge of a high-definition map vehicles still require global references to localize within the map and plan routes to a desired destination. However, consumer grade GPS suffers from large errors that make it unreliable for vehicle localization [4]. These errors can exceed several meters and in extreme scenarios hundreds of meters. These errors in large part are caused by urban canyons, poor weather, multipath, and other environmental challenges [4]. Consequently, sensor redundancy is essential for highly automated vehicles (HAVs).

1.2 Motivation

GPS Localization is essential for the safety and practicality of HAVs. However, they produce a localization signal prone to large errors. These errors can be caused by the presence of urban canyons, multipath, or severe weather. To ensure the safety of passengers in HAVs, and pedestrians, localization sensor redundancy is required. Visual Odometry is an increasingly important approach to localizing vehicles, due to the ubiquity of cameras, their inexpensiveness, and versatility for perception problems. The VO literature has been dominated by forward-facing VO pipelines that experience

challenges with uncertainty, brought on by dynamically moving obstacles, occlusions, severe weather, and varying sources of illumination changes. Downward-facing VO pipelines have been scarcely studied in the HAV literature. Other robotics platforms including UAVs, AUVs, and UGVs have shown that downward-facing cameras can provide localization signals for their respective use cases and are sometimes a requirement [92-94]. My research focuses on two major components:

- Given advances in visual odometry from higher compute resources, can visual odometry from downward facing cameras show significant improvements from earlier methods?
- And can a relatively simple model be used to give a robust odometry output where GPS localization fails?

1.3 Contributions

In this thesis research, a comprehensive review of Vehicle Localization (Chapter 2) and the Visual Odometry problem from forward-facing and downward-facing cameras (Chapter 3) is discussed. Several approaches for geometric, direct, and deep learning VO pipelines from downward-facing cameras are proposed that showcase the challenges of the problem. Experiments for classic feature extraction approaches and their results are in Chapter 4. Lastly, a VO pipeline that leverages the downward-facing view from a single monocular camera is proposed along with its experimental results (Chapter 5.) Chapter 6 concludes this thesis and discusses future work.

Chapter 2. Vehicle Localization Literature Review

There are two major fields of study that are relevant to this research. This chapter is a literature review on Vehicle Localization, and in Chapter 3 a Visual Odometry literature review and fundamentals are discussed. In this chapter, a general overview of vehicle localization methods will be discussed, motivating the need for sensor redundancy and accurate localization pipelines in vehicles, particularly self-driving and highly automated vehicles. In Chapter 3, major paradigms and advances in Visual Odometry will be discussed. This will underscore the need for cameras in localization pipelines for self-driving.

2.1 Vehicle Localization

Vehicle localization is a major paradigm for the development of automated vehicles. Localization is needed for a wide range of perception, control, and motion planning tasks. There are a multitude of approaches to both sensing methods, and the design of algorithms. Ego-localization was used as early as the 3rd century B.C. for surveying land with basic wheel odometry techniques as described by Vitruvius in his writings [5]. These techniques were also extended for ships to navigate large bodies of water [6]. For most of human history, the use of landmarks, stars, maps and basic odometry were used to estimate the relative location of a vehicle. And with the invention

of the car, not much had changed. However, with the invention of Radio Detection and Ranging (Radar) technology, large military vehicles on land and sea (and eventually air) began to localize targets. These observations were constrained relative to the vehicle and were primarily used for object detection and tracking. Although rarely used for ego-vehicle localization, the use of radar in localizing with respect to other objects and landmarks has been useful in automating vehicles, particularly with the integration of radar for Advanced Driver Assistance Systems (ADAS) [7]. In the 1950s, inertial navigation systems were invented for military use for the navigation of air vehicles [8]. In the early 1970's, the invention of GPS made it possible to give a vehicle's position within a global reference frame [9].

Reliable global localization is paramount for the widespread use of automated vehicles. Without a global reference point, vehicles can't reliably plan trips to different destinations. However, GPS is only accurate up to several meters. Urban canyons, multipath, and severe weather can contribute to these errors. RTK systems that can achieve centimeter level accuracy are expensive and require a ground station for operation. This motivates the need for redundant localization sensors that are relatively inexpensive. Other sensor modalities use the process of odometry to incrementally calculate the pose of the ego-vehicle across its time samples.

2.1.1 Wheel Odometry

In Wheel Odometry, a wheeled robot uses encoders to track wheel revolutions. The wheel revolutions are used to calculate the relative position from the initial starting

position of the robot to its current position [10]. Wheel Odometry has a few weaknesses. It is susceptible to position drift caused, in part, by lateral and longitudinal wheel slip. The error measurements over time are cumulative making it impractical for long range odometry as a stand-alone method. It also isn't suitable for applications where precise measurements are needed. For uneven or rough terrain, Wheel Odometry performs poorly. Any environmental disturbance that can cause slippage will have a significant impact on the trajectory calculation. One of its major strengths is that it's an inexpensive method for localization. It's often used to benchmark other odometry algorithms (e.g., Visual Odometry.) It also has a simple model and calculations. Overall, this method is well suited for vehicles of any size or budget and works well with other forms of odometry.

2.1.2 Inertial Odometry

Inertial Odometry (IO) utilizes an Inertial Navigation System (INS), namely inertial measurement unit (IMU) sensors to determine position, orientation, altitude, and linear velocity of a robot. Automotive grade IMU sensors are typically microelectromechanical systems (MEMS) devices with a 3-axis accelerometer and 3-axis gyroscope [10]. The accelerometer measures non-gravitational acceleration. The gyroscope measures orientation based on gravity and magnetism or angular rate of rotation [10,132]. Inertial Odometry is often used in combination with Visual Odometry [11-14]. IMUs are small and low power which makes Inertial Odometry feasible for vehicles of all sizes that operate at longer mission lengths. IMUs are relatively accurate

without external sensors or other references. However, they still experience drift errors that are inherent to odometry problems. Constant gyroscope and accelerometer error can accumulate in the integration process producing position and velocity errors. Consequently, over long periods of time this integration error can lead to poor performance.

2.1.3 Laser Odometry

In Laser Odometry point clouds are generally preprocessed and registered to recover the motion of an ego-vehicle [133]. These point clouds are often collected from LiDAR (Light Detection and Ranging) sensors. For this reason, Laser Odometry is often called LiDAR Odometry (LO). LiDAR has multiple packaging types (e.g., scanning, solid state) with varying size and field of view. LiDAR provides large amounts of detailed position data relative to the ego-vehicle. These properties make LiDAR Odometry a suitable choice for ego-vehicle localization. A classic approach to this problem is the Iterative Closest Point (ICP) algorithm [133,134]. In this approach the distance to keypoints from a transformed source point cloud to keypoints in a target point cloud are measured. Points that are sufficiently close together based on a dissimilarity metric are weighted more heavily in the cost function. The objective is to find the transformation that minimizes the sum of squared errors for the point cloud correspondences [133]. There are several methods that improve this approach [135-137]. In addition to point correspondences, LO pipelines can also utilize distribution correspondences [138] and network correspondences [139] to estimate the pose of an

ego-vehicle [133]. There are a few weaknesses to LO pipelines. Lidar Odometry is nonoptimal for highly occluded scenes and large planar surfaces. Lidar Odometry is more suitable for well-structured environments. Lidar Odometry is also computationally expensive and power hungry [10].

2.1.4 Radar Odometry

Radar Odometry estimates motion by analyzing radar scans. Radar sensors are unique in that they can measure velocity, range, and angle of arrival from the sensor itself. Radar sensors come in pulse and continuous wave formats [10]. Pulse wave radar emits short bursts of pulses and listens for echo returns. Frequency modulated-continuous wave (FMCW) radar transmits multiple streams of modulated continuous wave signals. Radar fused with IMU or RGB Cameras is a good option for odometry and overall vehicle position estimation [15,16]. Continuous Wave radar has relatively high-resolution images compared to Pulse Wave radar [10]. Continuous Wave radar also has low sampling rate, low power consumption and minimum target range. Radars in general are long range, immune to poor weather, and perform well in low textured environments. Radars are good standalone sensors for motion estimation. There are also 1D and 2D radar options available. For the Pulse Wave radar, there are blind spots. For some feature extraction methods (e.g., Hough Transform) radar data processing becomes computationally expensive [10]. Non-flat terrain and road infrastructure can lead to misleading returns from the radar (i.e., outliers are significant.)

2.1.5 Visual Odometry

Visual Odometry estimates a camera's pose from image sequence features. Visual Odometry is related to Structure from Motion (SfM) in that image features are often used to calculate depth (i.e., structure), but this is often done to aid in the calculation and refinement of the camera pose and scale [17]. In Structure from Motion these features are used to estimate the 3D structure of the scene as its primary objective. Visual Odometry works well with the fusion of IMU and LiDAR data as well [10-14]. In general, stereo visual odometry outperforms monocular visual odometry approaches due to the lack of scale ambiguity and ability to calculate depth more easily. For monocular VO the translation vector between image sequences has scale ambiguity so additional algorithms are needed to estimate scale. In general, VO struggles in scenarios with low texture, non-Lambertian surfaces, and overall poor lighting conditions [17]. However, due to the sensor's inexpensiveness, versatility, and ubiquity, VO remains an active area of research and is essential for automated driving.

Chapter 3. Visual Odometry Fundamentals

Visual Odometry (VO) is the process of estimating the ego motion of a vehicle, human, or other agent from the image frames of a camera or multiple cameras attached to the agent [17]. VO in the last few decades has been heavily researched [18-28]. The first relative pose estimation approach from images was proposed by Erwin Kruppa in 1913 [29,30]. In this work, Kruppa finds the relative pose of a 3D object between two calibrated images from five feature point correspondences [29,30]. The first VO system was credited in [31] to Hans Moravec for his PhD dissertation and work at the Stanford AI Lab in 1980 (Figure 1.) This research and following works were focused on applications for NASA's Mars Rover missions. Remote control from Earth to Mars is impractical due to the latency in RF signals. Likewise, on Earth relying solely on RF signals provided by technologies like GNSS are insufficient for robotics operation. Hence the need for more sophisticated, unmanned robots with adequate sensor redundancy. In his work, Moravec equipped a small electric cart with a TV camera. The motion estimation was done using a sliding stereo approach, where the camera took 9 images over a 52 cm track to recover the motion. Although this approach was slow, it marked the first known use of cameras to estimate motion for robotics applications.

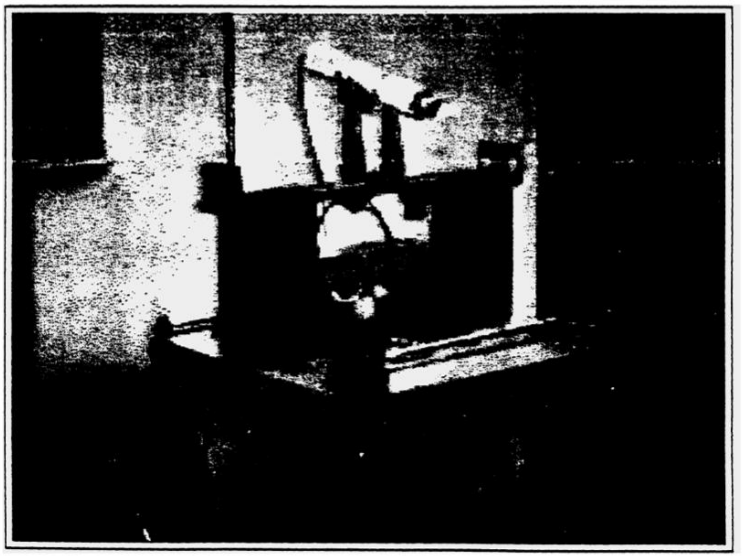


Figure 1. An image of the electric cart used by Hans Moravec to conduct his initial visual odometry experiments [31].

In 1989, researchers at Carnegie Mellon University published the first known use of camera pose networks for use in automated vehicles [2]. This work, called ALVINN (Autonomous Land Vehicle In a Neural Network) was published at NeurIPS, and utilized a camera and range finder as input to a backpropagation neural network to estimate the pose of the vehicle (Figure 2a.) The test vehicle was a large truck equipped with many computers. This work is credited for being one of the first visual odometry use cases in automated vehicles, in addition to being one of the first examples of the use of neural networks for the problem. The input images and range data were small (30x32 and 8x32 pixels, respectively) to accommodate the neural network. The network output consisted of 45 direction nodes (Figure 2b.)

(a) (b)

Figure 2. (a) An image of the NAVLAB test vehicle at Carnegie Mellon for their work on ALVINN and (b) the proposed neural network architecture for the pose estimation [2].

3.1 Advantages and Disadvantages

There are a few major advantages to utilizing VO pipelines over other localization methods in automated vehicles. The first major benefit is that cameras are ubiquitous and cheap on the market. This makes the sensors easily accessible due to low relative cost and ease of access. In contrast, although LiDAR odometry methods might generally outperform VO methods, their high cost makes them impractical to use at larger scales. Hence why many self-driving cars with LiDAR are limited to research settings and robotaxis (e.g., Waymo) whereas even vehicles with limited levels of autonomy still use cameras for ADAS applications [32]. Developing robust and accurate VO pipelines can help bring self-driving cars to market by providing a redundant source of localization information. It can complement GNSS data to help mitigate errors caused by RF interference (e.g., urban canyons, overcast skies.)

Another major advantage of using cameras for odometry is that the data can be shared across multiple tasks relatively efficiently. For example, many perception tasks in self-driving like object detection, object tracking, action recognition, image segmentation, and more utilize images in their perception stack. Although these tasks can be done with point clouds, it remains an active area of research [33]. Again, the use of cameras for vehicle localization can make self-driving more practical for consumers.

Cameras also offer rich information beyond the resolution of radar, GPS, INS, and other sensors. They are an attractive option due to their data diversity. Camera sensors take on the form of RGB, RGB-D, infrared, stereo, event, omnidirectional, and multi-view stereo camera systems (Figure 3a-g.) Each modality has its own advantages and disadvantages for VO.

RGB-D cameras offer rich color and depth content, usually through an additional active sensing method (e.g., structured light with lasers.) This gives RGB-D VO pipelines the advantage of depth information without the need for significant computational overhead. However, texture less and non-Lambertian surfaces remain an issue for this camera modality in the context of VO. Likewise, stereo VO pipelines can recover depth information more easily than monocular camera methods. Well calibrated stereo camera systems also don't have a scale ambiguity problem, because of the known stereo baseline between camera sensors. However, these methods require accurate calibration that can change in varying environments, and like RGB-D methods, have a limited range of depth. Multi-view stereo VO pipelines use multiple cameras (with two being the special case of stereo) to calculate odometry.

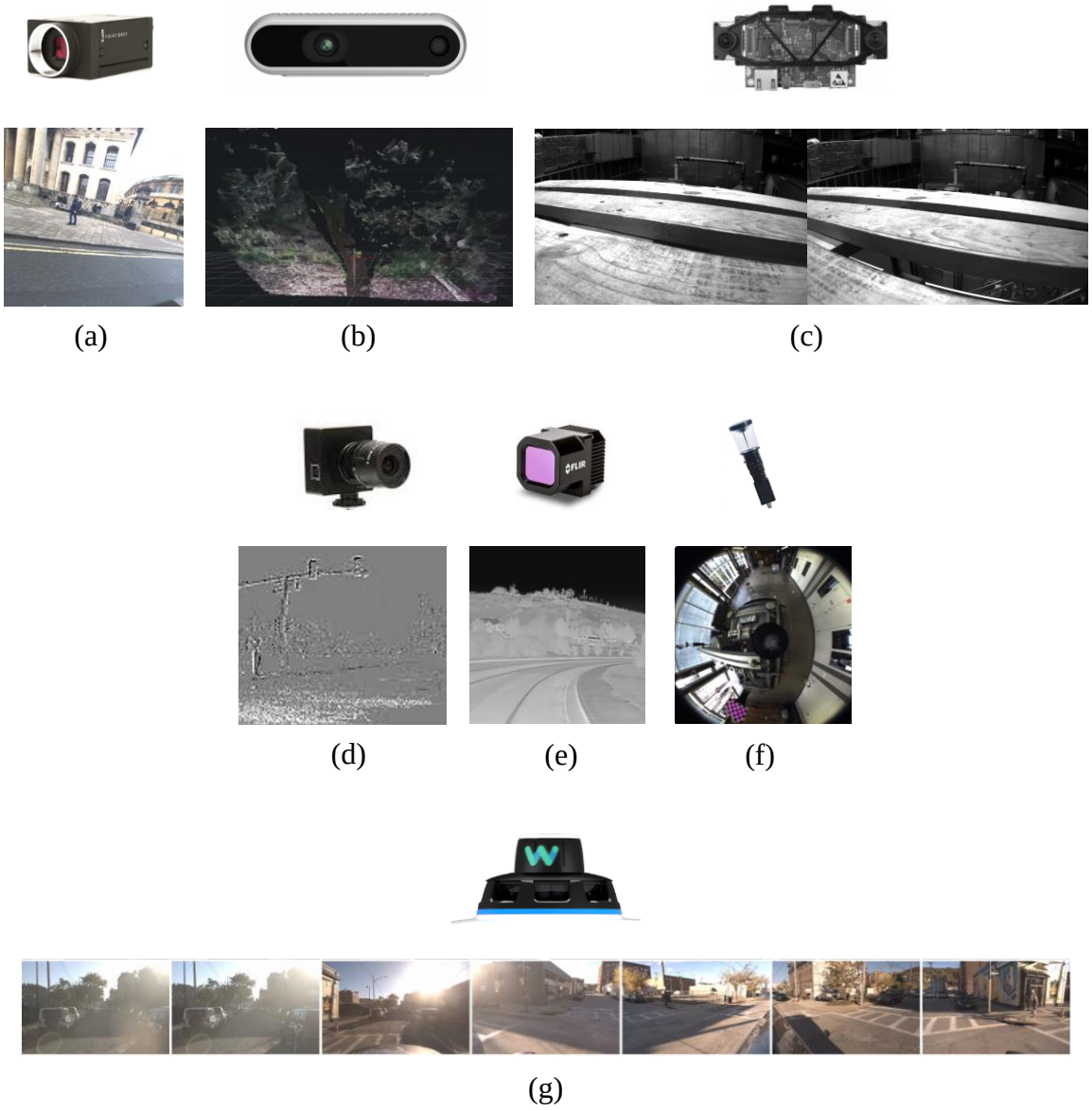


Figure 3. Different camera modalities for VO pipelines and examples of their respective data: (a) RGB, (b) RGB-D (c) Stereo, (d) Event, (e) Infrared, (f) Omnidirectional, and (g) Multi-View Stereo [34-40].

In general, these methods are the most accurate because the redundancy of several cameras can help constrain the geometry of the VO problem. However, the addition of multiple cameras is often a heavy computational load, and modelling can be complex (e.g., [41].) Event cameras in VO pipelines are a relatively new approach [42]. These cameras record the motion of a scene, don't experience motion blur, have high dynamic range, and have low latency. Despite these promising features and performance in relatively recent works [43,44], its major disadvantage is that they are less accessible, and for the moment are inaccessible for large scale robotics endeavors like self-driving cars. For omnidirectional camera approaches, the large field of view can lead to better retainment of features and information. These methods are also more robust to rotation-only motion, which is a challenge in most monocular VO pipelines with smaller fields of view. Again, these cameras are relatively expensive compared to their narrower field of view counterparts. In addition, another major disadvantage of omnidirectional VO pipelines is the more complex system modeling [45].

In contrast to the more specialized camera modalities, monocular RGB cameras are ubiquitous, cheap, and versatile. The major disadvantage of monocular VO pipelines is the added overhead of computing depth and the well-known scale ambiguity problem [17]. Monocular depth estimation and scale recovery have been an ongoing topic recently [46]. Infrared cameras are less utilized in the automated vehicles body of research for localization. Their major advantage is the ability to work in nighttime scenarios where RGB cameras tend to struggle, even in the presence of streetlights and head lights. However, they tend to have lower dynamic range and are only used when needed (e.g.,

localization for nighttime driving.) This thesis primarily focuses on the monocular RGB camera setting.

3.2 Visual Odometry, Structure from Motion, and SLAM

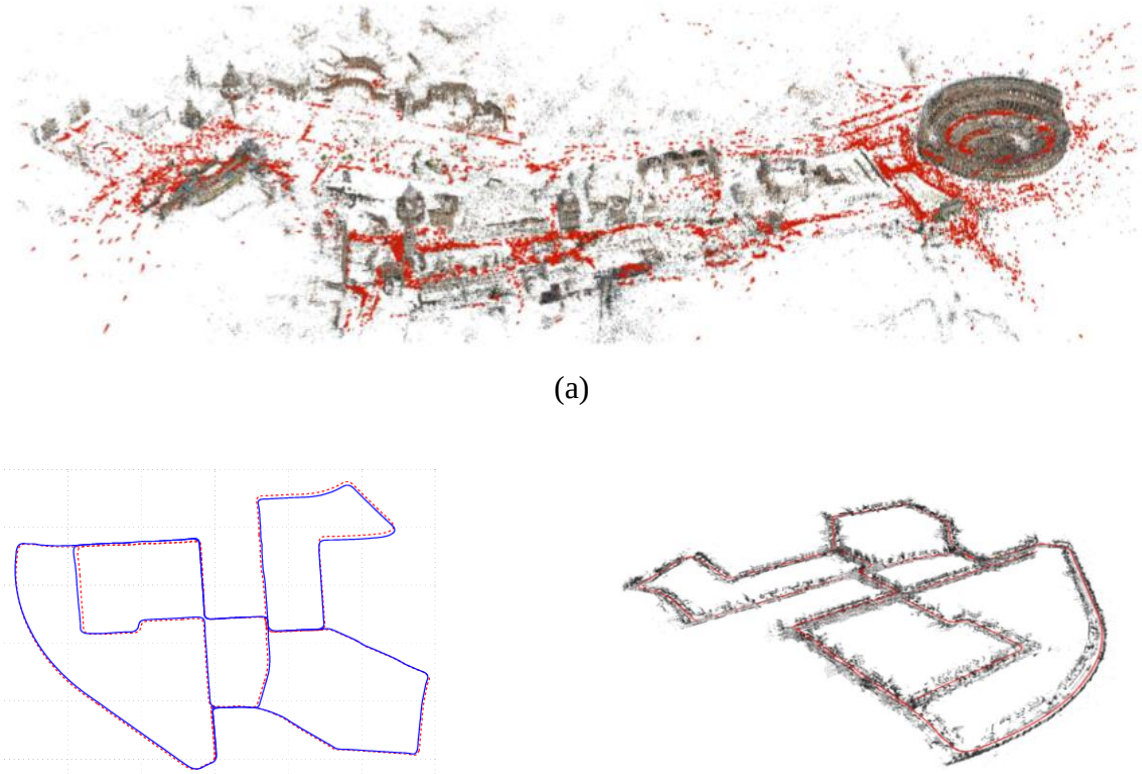
Visual Odometry is a major component of Simultaneous Localization and Mapping (SLAM) and Structure from Motion (SfM) (Figure 4.) Structure from Motion describes the calculation of both depth (structure) and relative camera poses (motion) in a scene. Classical SfM pipelines are often used to jointly reconstruct the 3D structure of the environment and poses of the cameras. This approach is usually adopted to reconstruct objects of interest, rather than moving dynamic scenes. The cameras involved in the reconstruction can also have different intrinsic properties. In this case, the well-known bundle adjustment problem can be employed to refine the estimated structure (depth), camera poses, and camera intrinsics [47]. Bundle adjustment is a back-end optimization scheme that minimizes the reprojection error of camera observations (pixels or 3D points.) The SfM problem is related closely to VO and SLAM in that the computation and refinement of depth and pose are similar in nature. However, the end goal of SfM is to reconstruct a presumably static environment with a generalized set of cameras. These cameras can be unordered, where VO and SLAM pipelines assume a sequential set of cameras. In VO and SLAM, it is also generally assumed that there is one camera with the same set of intrinsics, and the end goal is to localize an agent within its environment. SfM pipelines also tend to run offline, due to the incorporation of all the depth, poses, and intrinsics being formulated into one bundle adjustment problem. For VO and SLAM

pipelines, there are real time constraints. To reduce the optimization runtime, typically bundle adjustment is implemented in a windowed fashion.

SLAM algorithms are critical for highly automated vehicles (HAVs). HAVs typically have a high-definition map of its operation environment. This map provides the ego-vehicle with a globally consistent way to localize itself (e.g., [18-20].) Due to dynamic obstacles, season changes, and non-rigid bodies within a vehicle's environment, the map is subject to change. Consequently, a method for updating maps is needed to ensure the vehicle can safely operate in this environment. SLAM as the name implies, computes the vehicle's pose (localization), and generates a depth map to update the high-definition map of its environment (mapping.) These processes are done simultaneously. Additionally, many SLAM algorithms implement a variation of loop closure and map update scheme. Loop closure is the process of detecting previously encountered landmarks and refining the calculated trajectory to close loops caused by the inherent error drift of odometry algorithms [17]. This process is usually done in a back-end optimization step. VO algorithms typically do not include a loop closure step, because it requires the building and tracking of a map. However, the usage of other back-end optimization algorithms like bundle adjustment and pose graph optimization are used to help refine the calculated trajectory [18-28].

SLAM algorithms also differ from Visual Odometry pipelines in how depth information is processed and tracked. Both SLAM and VO pipelines tend to use depth estimation techniques to aid in the calculation of pose estimation through image and point warping. However, VO pipelines do not require a map to be built, processed, or

maintained. This thesis is primarily concerned with the localization of an ego vehicle, and not the overall system in self-driving vehicles. For this reason, this work primarily focuses on odometry when discussing pose estimation approaches. Some major breakthroughs in SLAM include MonoSLAM [48], PTAM [49], ORB-SLAM [18-20], LSD-SLAM [22] and many more. Overall, SfM is a 3D reconstruction approach from generalized, unordered sets of cameras. Whereas VO and SLAM pipelines can be considered local and global approaches for estimating an ego-vehicle's trajectory [17] (Figure 4.)



(a) (b) (c)

Figure 4. An example of the data output for the (a) Structure from Motion, (b) Visual Odometry, and (c) Simultaneous Localization and Mapping problems [50, 18, 28].

3.3 Classical Visual Odometry Methods

There are three major classes of VO algorithms: Classical, Direct, and Deep Learning. The first set of algorithms are considered classical or geometric VO methods. These VO methods start with a pipeline frontend that consists of feature extraction, feature matching, outlier rejection, motion estimation and optimization steps (Figure 5.) Many early methods use this schema. Classical methods tend to be accurate for large interframe motion and utilize sparse features to estimate ego-motion. However, they tend to lack robustness for significant photometric noise across image frames, motion blur, high and low frequency patterns. These methods rely on significant scene structure and static observations. These methods also tend to outright fail under these conditions and no odometry estimates are produced.

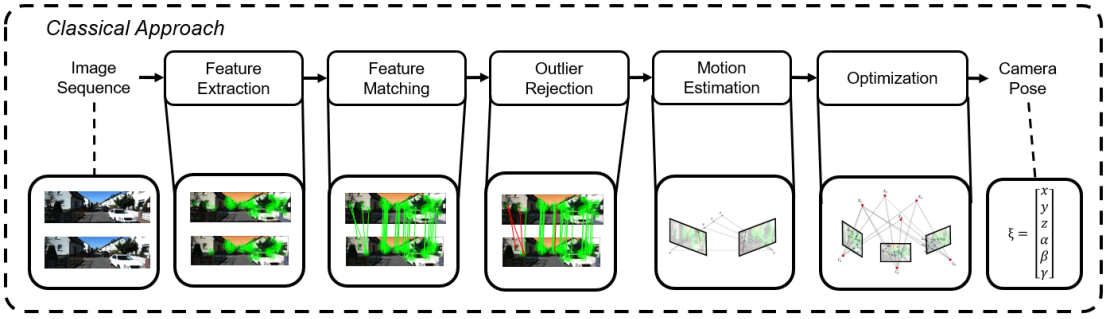


Figure 5. A high-level view of classical VO pipelines.

In the feature extraction step, interest points or corners are extracted from the image. There are several approaches to extracting corners from an image sequence.

Common methods include SURF [51], SIFT [52], ORB [53], Harris [54], and Shi-Tomasi [55] corner detection. These are all classical feature extraction approaches that use heuristics to identify feature points. In recent years, deep learning-based feature extraction methods have been proposed that are more robust to photometric inconsistencies (e.g., LoFTR [56]). These features are then matched between image frames with an algorithm like Brute Force Matching or FLANN [57]. These correspondences can be erroneous for many reasons. Consequently, an outlier rejection scheme is introduced to filter out correspondences that will negatively impact the pose estimation. Random Sample Consensus (RANSAC) is the standard in this area and has many proposed variants [58]. These features are triangulated by utilizing epipolar geometry. The final step includes a non-negotiable bundle adjustment formulation that optimizes the camera intrinsic and extrinsic parameters of the image sequence. The following sections will detail the inner workings of this classical approach.

3.3.1 Camera Model

Real camera systems have complex lens hardware that distorts incoming light rays for better camera functionality. For example, for the convenience of the photographer, cameras can have zoom capabilities so that objects within a scene can be placed within a proper depth of field [132]. This can help reduce blur for out of focus subjects in images. These lenses make camera modeling challenging, and in practice, cameras are assumed to follow a ‘thin lens model’ where these distortions are negligible [132]. As seen in Figure 6, a 3D point, P is projected into the camera after passing

through a lens, and is projected to 2D image point, p’. As the lens becomes thinner, point z’ and focal length f become approximately the same, and this assumption can hold in practice. The point, z’ is the distance from the lens to the projected point, p’. The focal length is the distance from the lens to the point where light rays converge on the camera’s principal axis. In this work, the thin lens model and the commonly adopted pinhole camera approximation will be used to model how 3D points are projected onto a camera’s image plane. This model is used in practice, because the size of the camera aperture and the depth of field are inversely proportional. For an infinitesimally small aperture, (i.e., pinhole) the model can have a large depth of field [132]. This simplifies the camera projection model. In this model all light rays converge to the center of the camera. The image plane to the camera center can be considered the focal length. And the principal point, p is the point where the camera’s principal axis intersects the image plane. This approach helps form simple relationships between the 3D observations and projected pixel points with similar triangles (Figure 7.)

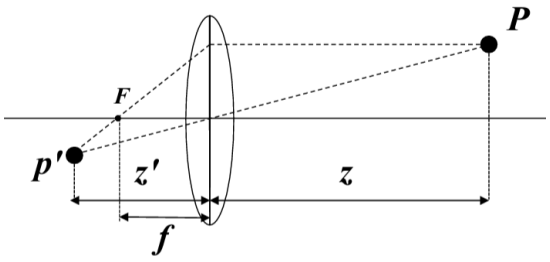


Figure 6. A basic model of a modern camera with focal length (f), 3D observation (P), depth (z), projected point (p’), and distance from lens to projection plane (z’). The light rays observed from 3D observations converge at point F on the principal axis [17].

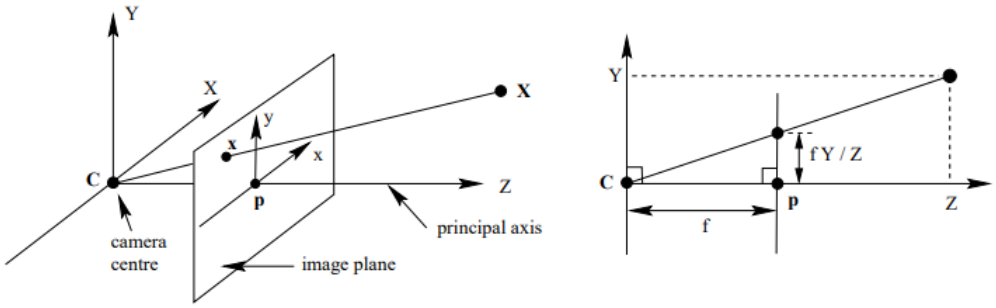


Figure 7. The Pinhole Camera Model used in many computer vision tasks. [47].

The 3D points observed in the camera frame can be projected onto the image plane using the focal length and principal point of a calibrated camera. This is done by constructing the intrinsic camera matrix, K (Eq. 1.2.) The projected points are then normalized by dividing the vector by its last element to put it into homogeneous form (last element is 1.) The homogeneous form is convenient for matrix operations.

$$p' = KP' \tag{1.1}$$

$$Intrinsic\ Camera\ Matrix,\ K = \begin{bmatrix} f_u & 0 & c_u & 0 \\ 0 & f_v & c_v & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{1.2}$$

$$projected\ point,\ p' = \begin{bmatrix} u \\ v \\ w \end{bmatrix} \rightarrow p' = \begin{bmatrix} u/w \\ v/w \\ 1 \end{bmatrix} \tag{1.3}$$

$$3D\ point,\ P' = \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{1.4}$$

For 3D points expressed in the world coordinate frame, these points can be projected into the current camera by applying a rigid body transformation with rotation matrix R and translation vector t. This transformation with both intrinsics, K and extrinsics R and t, form what is known as the Projection Matrix. (Eq. 2.1-2.2.)

$$x' = PX' \tag{2.1}$$

$$Projection\ Matrix, P = K[R|t] \tag{2.2}$$

The rotation matrix is an orthonormal matrix that describes the rotation of a point in n^{th} dimensions. This matrix belongs to the Special Orthogonal Group, SO(n) [60]. For this work 2D and 3D rotation matrices are used to describe rotations among 2D/3D Euclidean points and pixel locations. The rotation matrix can be decomposed into Euler angles: roll, pitch, and yaw. These angles describe the rotation about the x, y, and z axes respectively (Eq. 3.1-3.2) The translation vector, t is an n^{th} dimension change in Euclidean distance. The combination of the rotation matrix and translation vector form the transformation matrix (Eq. 3.4.) The transformation matrix provides a convenient form to concatenate rigid body transformations. This matrix belongs to the Special Euclidean Group, SE(n). The SE(2) and SE(3) manifold parameterizations are often used in the VO literature to represent relative camera poses [60].

$$Rotation\ Matrix, R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} = R_\gamma R_\beta R_\alpha \tag{3.1}$$

$$R_\alpha = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\alpha) & -\sin(\alpha) \\ 0 & \sin(\alpha) & \cos(\alpha) \end{bmatrix}, R_\beta = \begin{bmatrix} \cos(\beta) & 0 & \sin(\beta) \\ 0 & 1 & 0 \\ -\sin(\beta) & 0 & \cos(\beta) \end{bmatrix}, \quad (3.2)$$

$$R_\gamma = \begin{bmatrix} \cos(\gamma) & -\sin(\gamma) & 0 \\ sin & \cos(\gamma) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$roll: \alpha, \qquad pitch: \beta, \qquad yaw: \gamma$$

$$Translation\ vector, t = \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} \quad (3.3)$$

$$Transformation\ Matrix, T = \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix} \quad (3.4)$$

Quaternions can also be used in tandem with translation vectors to represent rigid body transformations. These rotations reside on the tangent space of the Special Unitary Group, SU(2) [60]. This representation is common in computer graphics for its compact form. However, the quaternion is less common for VO pipelines, and for this reason, the main rotation parameterization used is the rotation matrix and its corresponding Euler angles in this research.

3.3.2 Feature Detection and Matching

Feature points, salient points, keypoints, and corners are all used to describe points of interest in an image. It's a basic component of many computer vision algorithms. Likewise, feature detection is a key component in classical, geometric VO pipelines. In contrast to area-based or patch-wise comparisons in computer vision that

seek to acquire some global or high-level information about the image, the objective of feature detection is to extract local information in images. These features can be used for tasks including image registration, matching, warping, etc. Global information in an image can be sensitive to occlusions and photometric changes [61]. The benefit of having local features is that algorithms have fine grain information about the image. In general, after feature extraction, an area around the feature point is extracted and a descriptor is computed to match these keypoints.

To compare these features across images, there needs to be some notion of repeatability and uniqueness to reliably match features across images [59]. This often is the case in descriptor-based feature extraction where extracted features are assigned values that gauge the uniqueness of the feature. Feature detection methods should also be able to produce a large quantity of features from an image, to cover different objects, structures, etc. An insufficient number of detected features can lead to challenges downstream in a VO pipeline. In general, these features should also be relatively efficient so that performance in real time scenarios is possible. In VO and SLAM this is particularly important because pose estimates are often needed in excess of 10 Hz. Features that are computationally inefficient can make these approaches difficult to use in practice. Feature detectors also need to be rotation, translation, and scale invariant. In practice, cameras and object motion cause features to change location and size across image frames. Therefore, feature detectors need to detect features after they experience transformations from movement. Geometric and photometric variation are also an important consideration for the design of feature detectors. Images can experience

changes in lighting, noise, and blur across image frames [47]. And in some settings the presence of low and high frequency texture can further complicate the detection and matching process. Detectors that are robust to these inconsistencies are considered reliable for general computer vision tasks, and VO pipelines.

There are many notable feature detectors in the computer vision literature [52-56]. In general, these detectors can be classified by how they extract features as gradient, intensity, curvature, and learning based detectors. In gradient-based detectors, the image gradients are used to define a metric that reflects the 'cornerness' of a feature. Corners are desirable feature primitives to detect because matching features on lines leads to ambiguity. A pixel might be localized anywhere along the line in the next image frame. For corners of objects in an image, the pixels have locally high gradients which makes them easier to track across image frames. In intensity-based feature detectors the pixel's local intensity values are used to develop a notion of cornerness. In curvature-based detectors the objective is to extract information based on structure. In particular, edges, contours, and regions are used as features [62]. In learning-based methods machine learning and deep learning are used to extract features from images by learning from data. Deep learning feature detectors have become a recent trend across many computer vision tasks (including feature detection and matching) for their robustness and ability to extract higher level features from images (e.g., [56].) In contrast to hand crafted features and heuristics, these methods require less parameter tuning in practice (at inference time) and are more robust to changing environments, blur, brightness changes, and more (e.g., [56].) In indoor environments, line-based and blob detectors can be used to help track

low texture regions and groups of pixels with similar content [63,62]. This research focuses on outdoor environments in the context of automated driving, and therefore, this body of literature is not considered in this research.

A notable feature detector, called Features from Accelerated Segment Test (FAST) is commonly used in general computer vision tasks and the VO literature [64]. As the name implies, it is computationally efficient. It works by extracting a local, circular region of pixels and comparing the candidate feature to the pixels on the perimeter of the circle (Figure 8.) If most of the perimeter pixels are within a predefined intensity threshold, then the pixel is considered a reasonable feature to track.

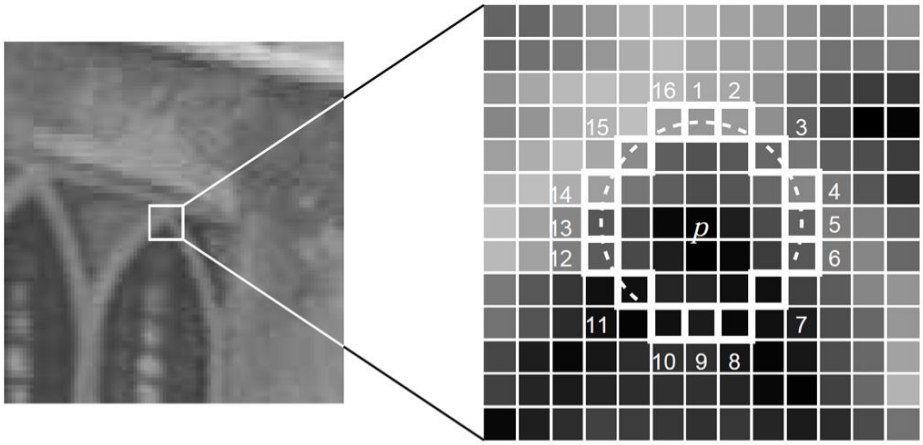


Figure 8. The local feature extraction approach of the well-known FAST feature detector [64].

Another notable feature detector, called the Harris detector is based on the correlation matrix of image gradients [54]. The intuition for this approach is that image gradients carry information about the cornerness of a feature. A score of zero suggests

that the gradients are near zero, which is common for low texture regions. Low texture regions are not good candidates for tracking, because matching is generally ambiguous since many pixels could be matched in the target frame. When one of the eigenvalues of the correlation matrix is much larger than the other, this suggests that there are strong gradients in one direction and not the other. This observation is common for edges, which are also ambiguous for matching since a pixel can be localized in many places along the edge in the target frame. Good features to track (i.e., corners) have two large eigenvalues in the correlation matrix, since image gradients tend to be larger in both image plane axes. This intuition is used for the Harris detector response function to extract features.

The Shi-Tomasi feature detector is an improvement on the Harris detector [55]. If the minimum eigenvalue is above a noise threshold, the feature is considered a good feature to track. The intuition is that the overall geometry of the feature primitive is not a concern, and if the eigenvalues are above the noise floor, then it's a reliable feature to track. Like the Harris detector, the Shi-Tomasi feature detector is rotation invariant, but not scale invariant.

The Scale Invariant Feature Transform (SIFT) detector and descriptor approach was considered state of the art for many years [52]. The approach uses Difference of Gaussians (DoG) to search an image for candidate feature points that are scale and rotation invariant. The keypoints are localized by determining their location and scale. Stable observations are selected. Local image gradients are used to determine orientation information. This approach makes the feature extraction method rotation, translation, and scale invariant. The descriptor uses the local orientation information based on the

gradients and is robust to deformation and illumination changes [52]. Later, Speeded Up Robust Features (SURF) was proposed as a fast approximation to SIFT [51]. This was done using Haar wavelets. It also outperformed its predecessors in robustness, and repeatability.

The Oriented FAST and Rotated BRIEF (ORB) feature detector was proposed as an efficient alternative to the SIFT and SURF features [53]. It is based on the binary descriptor BRIEF [65] and builds on the FAST keypoint detector. The implementation is robust to noise and two orders of magnitude faster than SIFT [53]. It is an essential component of the state-of-the-art geometric SLAM method, ORB-SLAM. Which is often used (without loop closure) for VO pipelines across multiple camera configurations [18-20].

Another notable feature extraction approach in the literature is the famous Kanade-Lucas-Tomasi (KLT) Tracker that uses the concept of optical flow [66]. In practice, the KLT Tracker is fast and a common approach for tracking features across multiple image frames in VO pipelines. The intuition behind sparse optical flow is that feature points experience small motion across image frames. Therefore, the KLT tracker seeks to minimize the image alignment error introduced by this incremental motion. As motions become larger this assumption of small movements begins to break down.

There are also recent works in deep learning for feature correspondences and matching. The Linear Invariant Feature Transform (LIFT) [67] models the detection, orientation estimation and descriptor pipeline altogether. The MagicPoint network was proposed in [68] and was an early use case for learning deep features for geometric

SLAM. Other methods include, D2Net [69], R2D2 [70], DISK [71], DRC-Net [72], LoFTR [56], and more. These approaches can also model multiple sections of the feature detection and matching pipeline. LoFTR is used in this research for its robustness to motion blur, and both high and low frequency texture.

There are several approaches to matching the previously mentioned descriptors. One such approach is brute force matching by seeking to exhaustively match a feature extracted in the source image to a feature in the target image for every target feature point. This method is simple but very inefficient. Another approach is to use the k-Nearest Neighbor (KNN) method [73] and match local candidates. Some methods use an area-based approach where pixel intensities are directly used for matching. This approach can be observed in basic stereo vision setups where matches are known to be observed row-wise, which makes the search less exhaustive. However, feature point matching across time (like in monocular VO) tends to avoid area-based matching. Feature matching can also be done with graphs by assigning features to nodes and correspondence distances to the edges of the graph [62]. Deep learning has also been used for matching correspondences [62].

3.3.3 Outlier Rejection

Point correspondences can contain many outliers throughout the feature detection and matching pipeline. This can be due to feature detectors, descriptors or matchers that lack robustness to geometric and photometric variation throughout the image. As a consequence, geometric VO pipelines require some notion of outlier rejection to

maximize the number of inlier point correspondences. For many years, the state-of-the-art outlier rejection scheme has been RANSAC. The algorithm starts by collecting the set of feature correspondences. A random sample of the feature correspondences are then taken, and a least squares model is fit to these points. The Euclidean distance to this model is calculated for all feature correspondences. The correspondences that are under a distance threshold are considered the inliers. This process is repeated until a predefined threshold is met. This threshold can be set to a max number of iterations based on probability of inlier success, outlier ratio, and number of points sampled [47]. The correspondence set with the most inliers (the most consensus) is used [58]. In the case of 1-point RANSAC only one iteration is needed and as the minimum number of points increases so does the number of iterations [74]. More recent methods relax the termination criterion to make the rejection process less conservative [78].

In general, the minimum number of correspondences required for relative camera motion is five-point correspondences, and the algorithm is thus sometimes referred to as 5-point RANSAC. For wheeled vehicles, by exploiting nonholonomic constraints, only one point correspondence is needed. In [74], Scaramuzza et. al. propose this method and call it 1-point RANSAC. Raguram et. al. propose Universal RANSAC (USAC), a general purpose framework for outlier rejection [75]. The N-adjacent Points Sample Consensus (NAPSAC) algorithm proposed in [76] relies on the intuition that inliers tend to be close together while outliers tend to be much farther away. The approach is effective at drawing inlier samples; however, it is prone to degeneracies [75]. In [77], the Progressive Sample Consensus (PROSAC) algorithm is proposed. In this algorithm, point

correspondences with a higher quality metric are used from the start instead of stochastically like RANSAC. The algorithm eventually progresses into the RANSAC scheme [77]. In [78], the Marginalizing Sample Consensus (MAGSAC) approach is proposed to improve on RANSAC. The model uses noise marginalization instead of thresholding. The model is considerably accurate, but rather slow. In [79], MAGSAC++ was proposed to speed up the algorithm. It uses several least squares fittings while marginalizing over the noise. The model is constructed using the Iteratively Reweighted Least Squares (IRLS) algorithm. Its quality metric improves on MAGSAC by using a lookup table for gamma functions instead of computing them at inference time [79].

3.3.4 Motion Estimation

Following outlier rejection in the classical, geometric VO pipeline, a motion estimation step is used to get a rough initial estimate of the camera's motion. There are different approaches to getting an initial estimate. However, the standard is to use epipolar geometry to form a constraint with the inlier set of pixel correspondences. The epipolar geometry connects pixels and 3D observations across multiple image frames. The epipolar plane can be constructed with the camera centers, and a 3D point observed in each frame. With a single image frame, the depth of a 3D point can't be localized from its projected pixel. Another view of this 3D point must be observed in another image frame in order to estimate the location of the 3D point. The epipolar line for a corresponding 3D point is a projection of the possible location of a 3D point in space, observed in another image frame. This can be observed by drawing a line from the

epipole to the pixel location of interest. An epipole is a point on the image plane where the line from camera center to camera center intersects. The relationship between the epipolar plane, epipolar lines, and epipoles can be observed in Figure 9.

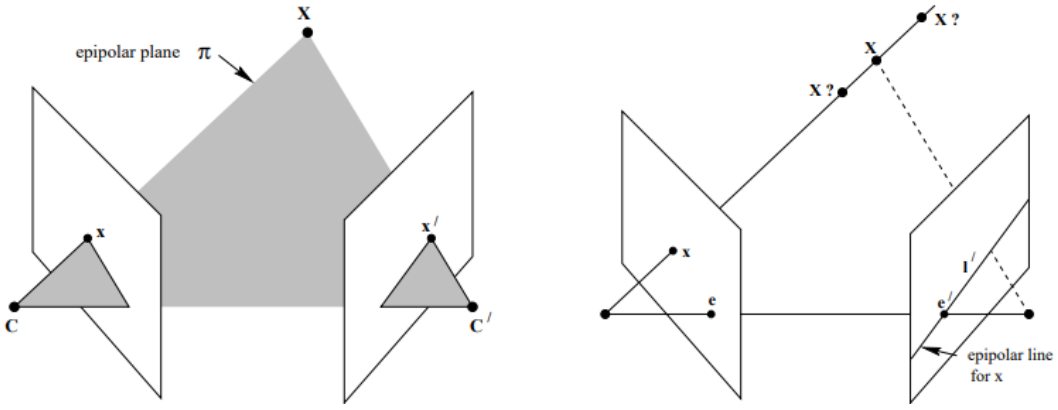


Figure 9. A basic depiction of the epipolar geometry used in classical VO pipelines [47].

The Fundamental Matrix constrains point correspondences between two image frames. The relationship in Eq. 4.1-2 essentially constrains the epipolar line to be orthogonal to any pixel observed in the image plane of the other image, where point correspondences x and x' are given.

$$epipolar\ line,\ l' = Fx \tag{4.1}$$

$$epipolar\ constraint,\ x'^{T}Fx = 0 \tag{4.2}$$

The Fundamental Matrix, F can be decomposed into a rigid body transformation and the camera intrinsics. The rigid body transformation is in the form of the Essential Matrix (Eq. 5.1-3.)

$$Fundamental\ Matrix,\ F = K'^{-T}EK^{-1} \tag{5.1}$$

$$Essential\ Matrix,\ E = [t]_x R \tag{5.2}$$

$$Skew\ Symmetrix\ Matrix,\ [t]_x = \begin{bmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{bmatrix} \tag{5.3}$$

The estimation of the Fundamental Matrix and Essential Matrix are commonly solved via the 8-point algorithm [47]. Given a set of N point correspondences (x', x), a Least Squares solution can provide the appropriate estimate of the parameters for each matrix. Once the Essential Matrix is computed, the rotation matrix and translation vector can be extracted using a singular value decomposition approach. In general, there are four possible solutions for each Essential Matrix decomposition (Eq. 6.1-2.)

$$T = [UWV^T|u_3],\ [UWV^T|-u_3],\ [UW^TV^T|u_3],\ or\ [UW^TV^T|-u_3] \tag{6.1}$$

$$W = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{6.2}$$

In Eq. 6.1, the U and V are the left and right matrices of the singular value decomposition. The third column of the left matrix, U is denoted as u_3. The estimated

translation vector is a unit vector, and a scale estimation approach must be implemented to get the true translation to scale. The Fundamental Matrix estimation approach for motion estimation is degenerate for planar motion, low parallax, and zero motion. Under the zero-motion case, the translation vector makes the skew symmetric matrix the zero matrix (Eq. 5.3.) Consequently, the epipolar constraint is satisfied without the need for computing rotation. This is an undesirable property for VO pipelines. In some VO pipelines, model selection is used to select a different motion model under these scenarios [18]. One such model is the Homography Matrix. A Homography is a perspective transformation, where the points observed are on a plane. Under this motion model, planar motion and zero-motion are not an issue. The Homography Matrix can be calculated in a similar fashion to the Fundamental and Essential Matrices. A classic approach is through the Direct Linear Transform [47], however, this approach is not robust to outliers. The Homography Matrix can be decomposed as seen in Eq. 7.2. Given the relative plane normal, n and the depth of the point(s) with respect to the camera the Planar Homography can be computed. This can be done similarly to the decomposition to the Essential Matrix by using the singular value decomposition [47]. There are also methods in the literature for Homography, Fundamental, and Essential Matrix estimation that formulate the problem as a point reprojection problem. Where the reprojection error of points is used to estimate the matrices [80,81]. There are also recent works that seek to estimate these matrices in a more robust fashion by leveraging deep learning [82,83].

$$x' = Hx \tag{7.1}$$

$$H = K'\left(R - \frac{tn^T}{d}\right)K^{-1} \tag{7.2}$$

3.3.5 Optimization

Rigid body transformations computed from the Fundamental or Homography Matrix are rough estimates of a camera's motion. Even with robust outlier rejection schemes, outliers can still contaminate the point correspondences used to calculate the pose estimates. This error in the pose estimation process can accumulate quickly over several image frame pairs and is known as error drift in odometry problems [17]. This error can be from systematic uncertainties in the VO pipeline and noise observed in the environment. An optional, but highly encouraged backend optimization scheme is often implemented in VO and SLAM pipelines. Like the SfM problem, bundle adjustment is often used to refine the pose and depth of the scene over several images. For VO pipelines, a windowed bundle adjustment is usually implemented to reduce computational load. Unlike stereo VO pipelines, monocular VO pipelines can't estimate depth from a single time instance. Multiple frames are needed to estimate and refine the depth. In bundle adjustment, the reprojection error of the point correspondences is often used as the objective for the optimization problem.

$$\min_{T,d} \sum_i^N \rho\left(x'_i - \pi(x_i, T, K, d)\right)^2 \tag{8.1}$$

$$\rho(\cdot) : robust\ loss\ function$$

$$projection\ function,\ \pi(x_i, T, K, d) = KTdK^{-1}x_i \tag{8.2}$$

The optimization problem can be formulated with a robust cost function, ρ. Often the robust cost function of choice is the Huber loss [84]. It behaves like the L2 loss within a radius, and like L1 loss beyond this region. This makes it robust to large outliers like the L1 loss, and more accurate within a region of confidence like the L2 loss. The projection function backprojects a pixel into the camera frame using intrinsics and depth. This 3D point is transformed given the estimated pose and then projected into the image plane of the other camera frame. The cost function is often minimized using the popular Gauss-Newton or Levenberg Marquardt methods [47]. This approach is the basis for many geometric optimization problems in VO. This approach can be modified to work for 3D-3D and 2D-3D correspondences as well.

3.3.6 Notable Geometric Visual Odometry and SLAM Methods

There are many monocular VO pipelines that have been proposed in the literature [18-28]. In monocular Visual SLAM (VSLAM), the first real-time case was in MonoSLAM [48]. Their proposed method was efficient, working at around 30Hz, and could produce smooth camera trajectories. Their method is a probabilistic SLAM approach that tracks the state translation, quaternion, velocity, and angular velocity vectors. Their states are approximated by a multivariate Gaussian probability distribution.

The authors use a general constant velocity and constant angular velocity model for state estimation. And they use an active feature measurement and map update scheme.

Another notable approach in the VO and VSLAM literature at this time was Parallel Tracking and Mapping [49]. This VSLAM approach was designed for Augmented Reality settings (i.e., indoor, and small desktop settings.) In this work, the authors parallelize the VSLAM pipeline on a dual-core machine to track camera motion and the map at the same time. Prior to this work, methods have operated sequentially, where map updates were done after the camera motion was determined. This parallelization made it possible to track motion and the map in real-time.

Many VO and VSLAM pipelines cite the work in LIBVISO2 [21] as a baseline in their research. Their approach is a stereo VO pipeline that has real-time scene flow estimation and dense stereo matching capability. The authors utilize a simple reprojection minimization problem and Kalman Filtering [85] for ego-motion estimation. This VO approach, although not a monocular method, is used to compare against many works in the monocular VO literature.

In addition to LIBVISO2, Mur-Artal et. al. in [18] proposed a state-of-the-art monocular VSLAM approach called ORB-SLAM. Their method utilizes ORB features across the tracking, mapping, relocalization, and loop closing portions of the pipeline. Their method works well for large scale environments due to their adoption of the Covisibility Graph. The Covisibility Graph is an undirected weighted graph that preserves the local structure of image frames [18]. They implement the model with a model selection scheme that uses classic epipolar geometry and the decomposition of the

Essential Matrix to recover pose with sufficient parallax motion. For limited motion and low parallax motion they use the Homography matrix for motion estimation instead. Their pipeline is parallelized over three threads to accommodate the demands of tracking, mapping, and loop closing. A high-level view of the architecture can be seen in Figure 10. Their work also extends to visual-inertial and stereo settings in the following works [19,20]. This method is the standard for geometric VSLAM pipelines and is often used as a baseline for monocular VO pipelines with loop closure disabled.

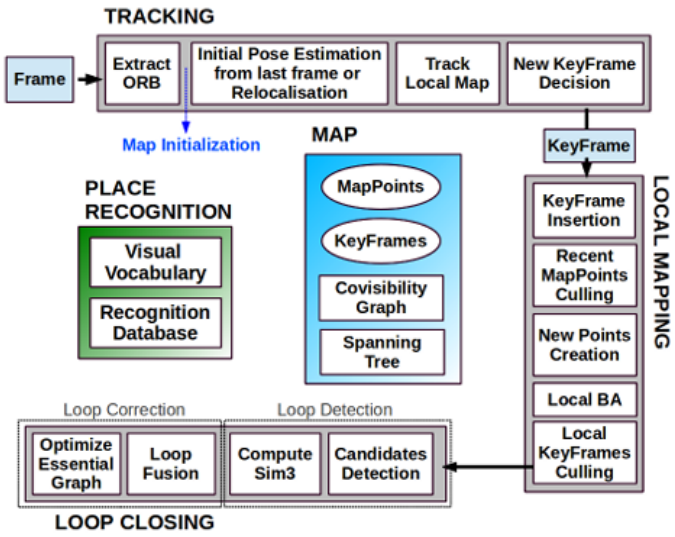


Figure 10. A high-level view of the ORB-SLAM architecture as seen in [18].

In recent years, other approaches have been proposed in lieu of the classic Fundamental and Essential Matrix decomposition for motion estimation. Kneip et. al. [86] propose an alternative to this problem by utilizing bearing vectors to help recover rotation separately from translation instead of epipolar lines (Figure 11.) The

Fundamental and Essential Matrix decompositions degenerate under planar motion and zero-motion scenarios. By decoupling the estimation of rotation and translation, frame-to-frame rotation and translation estimates can be accurate. In prior work, multiple frames were needed to get accurate pose estimates for monocular VO. Recently, the Probabilistic Normal Epipolar Constraint (PNEC) approach was proposed by Muhle et. al., where they reformulate the problem by adding uncertainty to the observed feature points [87].

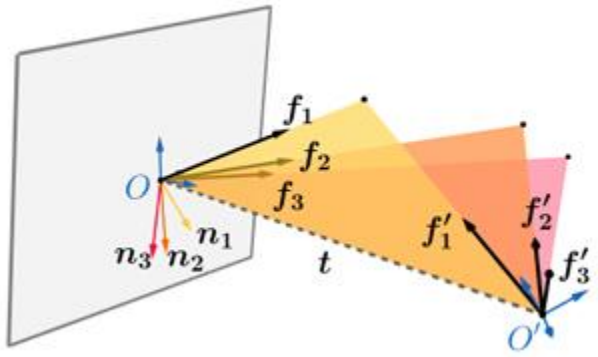


Figure 11. A depiction of the bearing vectors, f in the Normal Epipolar Constraint (NEC) and Probabilistic Normal Epipolar Constraint (PNEC) work. As seen in [87].

Overall, geometric VO and VSLAM pipelines have shown promising results for large motions and are relatively efficient. Under the constraints of real-time performance, the desire to use sparse features and efficient parallelization methods has shown to be effective in addressing the challenges in VO and VSLAM. Their use of sparse features incorporated in efficient front end and back-end optimization frameworks has also been shown to make these methods fairly accurate. However, these methods are still delicate

and tend to breakdown under the presence of photometric errors like inconsistent illumination from shadows, non-Lambertian surfaces, and dynamically moving objects [17,47]. Performance in low texture regions has also shown poor performance.

3.4 Direct Visual Odometry Methods

Geometric, Monocular VO and VSLAM have shown promising results with real-time performance and robustness against large camera motions. However, many of these methods lack robustness to motion blur, low texture, and changes in illumination. These methods also sacrifice the quality of information extracted from images by using sparse feature points. For these reasons, Direct VO pipelines have been proposed to leverage more information from the images. Rather than extracting sparse features throughout an image frame, the images are aligned through direct photometric optimization, by iterating over the SE(3) manifold. These Direct methods have shown to be highly accurate. However, their region of convergence tends to be smaller than the optimization schemes in geometric VO pipelines and are often prone to getting stuck in local minima. These methods also require careful attention to camera modeling for this reason. Many of these methods seek to calculate affine brightness parameters to address changing levels illumination across image frames [23]. And in some cases, camera parameters like gamma correction and exposure time are addressed to help enforce interframe photometric consistency [23]. Implementation of these algorithms often requires the use of low-level programming (e.g., SIMD Intrinsics) to achieve real time performance.

One early method, Dense Tracking and Mapping (DTAM), uses whole image alignment to track pose across image frames, and calculates a dense depth map [88]. This model was designed for indoor environments like PTAM. Although the approach considers illumination changes in their tracking scheme, it is not robust to global illumination changes. An example of the typical cost function for tracking in direct VO can be seen in Eq. 9.1. The intensities of the source image are warped using the estimated depth and pose. A robust cost function, ρ is chosen depending on the problem. Common choices are L1 loss [88], and more recently the Huber loss [23]. The projection function, π is the same as the projection function used in geometric bundle adjustment problems.

$$C(u,d) = \frac{1}{N}\sum_u \rho\left(I_{target}(u) - I_{source}\pi\left(KT\pi^{-1}(u,d)\right)\right) \tag{9.1}$$

Engel et. al. proposed Large Scale Direct SLAM (LSD-SLAM) that uses a direct VSLAM framework and achieves accurate performance over large-scale environments [22]. For their pose tracking they down weight their residuals with the variance of the residual error to help with robustness to outliers. Their final cost function also tracks depth in a similar fashion along with Sim(3) manifold optimization for tracking the pose and scale. They achieve real-time performance by leveraging SIMD Intrinsics for efficient low-level programming, and only tracking regions of the image with high gradients. A more detailed view of the high-level architecture can be seen in Figure 12.

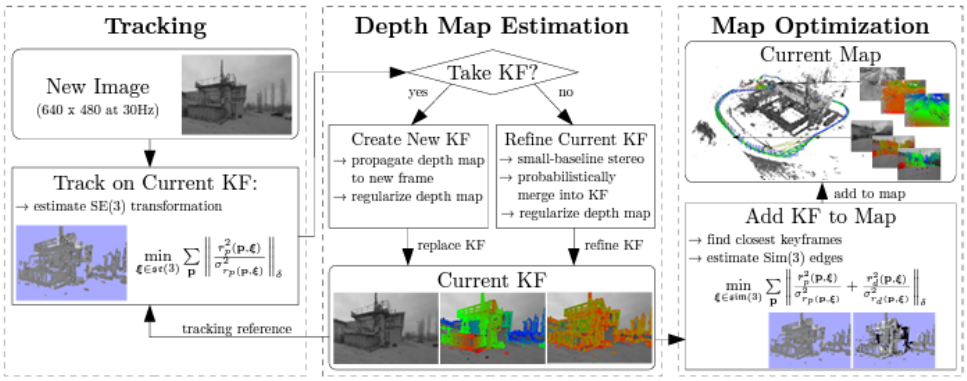


Figure 12. The high-level VSLAM architecture for LSD-SLAM as seen in [22].

In [23], Direct Sparse Odometry (DSO) is proposed as a pure direct VO pipeline. This approach differs from most direct VO pipelines in that it uses a full photometric calibration to effectively model photometric inconsistencies, which includes modeling the inverse response function and lens attenuation. It is sparse and direct because the optimization scheme operates directly on pixel intensities but does not use a geometric prior that leads to correlations in the Hessian matrix [23]. This approach is considered state of the art for direct VO. There are also hybrid approaches that leverage geometric and direct techniques. In particular, Semi-direct Visual Odometry (SVO) [24], proposed by Scaramuzza et. al. extracts FAST corners to do an indirect optimization on the points. And then this pose estimate is refined by doing a direct optimization, locally around the extracted features. This approach is fast and robust to high frequency textures, commonly found on ground images [24].

3.5 Deep Learning-based Visual Odometry Methods

In recent years, forward-facing, monocular VO and SLAM pipelines have started to incorporate deep learning into their pipelines. In these methods, a large corpus of data is prepared for training. In contrast to online learning methods, where statistical inference is done in a real-time fashion, deep learning methods learn through a training process where data is randomly fed into a neural network architecture over many epochs. These architectures often contain a large set of parameters. The learning process takes place by updating a loss function through backpropagation.

An early work has been proposed using a downward-facing camera and a range finder [2]. In this work, a multilayer perceptron network is fed this data, and the network predicts 45 directional outputs to predict the change in pose. In more recent years, the progress of Convolutional Neural Network (CNN) architectures has prompted the rebirth of deep learning for the VO and SLAM problem. In [89], an approach called PoseNet, uses a pose encoder to address the problem of camera relocalization in a supervised manner. This eventually led to works like DeepVO [25] that use pose encoders and Long Short-Term Memory units to create a VO pipeline in a supervised end-to-end manner. Supervised learning methods have shown evidence of model overfitting, and therefore lack robustness against out of distribution datasets. There have been many works since then that seek to solve the VO problem with unsupervised learning. The most effective approaches exploit the image warping relationship utilized in geometric and photometric bundle adjustment to learn both depth and pose during the training process. This process

does not require pose labels, and this joint learning approach has shown better robustness against unseen data distributions [26-28]. There are also some hybrid deep learning approaches that utilize deep camera pose networks as a robust front end to the VO problem, and bundle adjustment for the VO backend. In particular, D3VO is considered state of the art for deep camera pose networks. Despite being a monocular VO model, it is competitive against stereo and LiDAR approaches that often outperform monocular VO pipelines. It achieves this by learning aleatoric uncertainty throughout the training process to down weight residual errors caused by non-Lambertian surfaces and moving objects [28]. They incorporated this learned uncertainty map into their back end pose graph optimization framework to down weight large residuals in an online manner. This approach currently remains the best monocular VO method on the KITTI benchmark [90].

Overall, camera pose networks have shown great robustness towards many of the challenges in traditional geometric and direct VO pipelines. They also require less camera and environmentally dependent fine tuning that is common in these traditional methods, at the expense of large training times. General research trends continue to favor camera pose networks. Approaches like D3VO that leverage concepts from direct VO and deep learning VO pipelines have shown to be effective. And across all classes of VO pipelines uncertainty estimation has been shown to be effective at reducing errors.

3.6 Downward-Facing Camera Methods

Forward-facing cameras have dominated the general VO and VSLAM literature for both general scenarios and in automated driving applications. A major reason for this is because many feature extraction methods are not robust to some of the challenges that looking at the ground may introduce. In many cases for automated vehicles, ground plane images introduce severe motion blur, low and high frequency texture, large displacements, and changes in illumination. However, downward-facing approaches can also be appealing because they lack large moving obstacles and occlusions. They also typically have simpler models because motion is parallel to the image plane. And in some scenarios the lateral and longitudinal motion of the ego-vehicle are the only axes that are of interest. For Unmanned Aerial Vehicles (UAVs), Autonomous Underwater Vehicles (AUVs), and other Unmanned Ground Vehicles (UGVs), downward-facing cameras are often used because their environments of operation require it [92-94]. For example, UAVs at high altitudes need to localize themselves using buildings, vegetation, roads, etc., because their line of sight does not contain many features (e.g., clouds, skyline). Drones like Skydio use a downward facing camera (amongst others) to help constrain its motion [95]. Likewise, AUVs need downward-facing cameras because in deep bodies of water there might be few features to track in the open water, and tracking the seabed in some scenarios might be the only option. For UGVs in space like the Mars rover, downward facing cameras are also appealing, because many of the available features are present on the ground, and not the rover's immediate line of sight.

In HAVs, this problem has also been addressed but with little success [101,103]. There appears to be multiple reasons for these challenges. In contrast to these other

robotics platforms, HAVs typically operate at higher speeds, while the camera is fixed at the same height. As a result, the observable features are often small, severely blurred, and present with both low and high frequency textures. At heights less than 2 meters, the observable motion also leads to large motions which can be challenging for techniques used in more recent VO pipelines. This work seeks to overcome these challenges by implementing a more robust VO front end for the problem.

There are several approaches for addressing the VO problem from downward-facing cameras in an automated vehicle setting. Some of these methods utilize correlation, optical flow, dense point correspondence, ground plane estimation and hybrid methods. As mentioned earlier, in [2] ALVINN was proposed. This is considered the first example of using deep learning for camera pose estimation in an automated vehicle setting. The vehicle was equipped with a camera tilted downward to observe the road and a range finder. This data was given to a multi-layer perceptron network and yielded 45 directional outputs to estimate the direction of the vehicle. Later, in [105] the AURORA land departure system was proposed. This method used a downward-facing camera on the side of a vehicle to detect when a vehicle was leaving its current lane. This was done by using correlation and template matching to detect lane lines. Nourani-Vatani and Borges [96] proposed a correlation-based visual odometry method for an Ackerman-steering model. The method converts changes in pixels from the features to distances and transforms these distances to the vehicle reference frame. Then the robot pose is incremented. The changes in pixel locations are calculated with correlation-based shift detection and templates are determined by the template quality measure (TQM). TQM considers auto-

correlation, variance, median absolute difference, and entropy to evaluate the effectiveness of the template. The template with the highest TQM is used. This method works best for surfaces with high variability rather than smooth surfaces. However, as discussed in [102], these methods are susceptible to error from shadows and other lighting disturbances.

Aqel et. al. also uses a correlation-based method for testing an optimal configuration for a downward-looking camera [101]. Their work also includes a designed light source used at different times of day. They evaluate the quality of correlation between these different configurations. Yoshida et. al. [97] approach the Structure from Motion problem with an integrated approach that combines the correspondence and 3D recovery stages of SfM. At the correspondence stage, rather than using image feature correspondences, optical flow is considered according to a Gestaltian measure [97]. In general, optical flow works well when there are less image features available. This integrated approach gives a more reliable solution, but if certain constraints are not met, the entire reconstruction fails and provides no solution. Zucchelli et. al. proposed a constrained minimization problem for SfM from optical flow to better capture the constraints of man-made and naturally occurring structures [98]. Their solution is stable and efficient. Fakih and Zelek use a hybrid method where feature correspondences and optical flow are used [99]. The method works well when feature correspondences are low or noisy. This is particularly applicable to downward-facing cameras because the ground features are typically fine and difficult to extract consistently across image sequences. In general, feature

correspondences are more accurate than optical flow, however, optical flow works when feature correspondences are sparse.

Swank proposed a monocular, downward-looking camera that used feature detection and optical flow [94]. The optical flow analysis was done with a Pyramidal implementation of the Lucas-Kanade optical flow technique. To avoid large variations in light intensity across frames a light source was added with the camera. This method doesn't ensure consistent trajectory calculations because image sequences might lack the features for reliable detection. Zienkiewicz and Davison proposed a downward-facing camera on a small robot for VO. They assume that the ground is planar. They used an iterative dense alignment approach which utilizes all the data from an image to provide correspondences [100,104]. They also provide details on an auto-calibration process. Problems arise when objects on the ground don't appear flat. Other methods like Hong et. al. use shadow removal to address issues with light variation that the other authors have discussed [102]. They achieved better results for their downward-facing method than their forward-facing method. In [105], Lee et. al. use a downward-facing and oscillating camera to recover the velocity of the moving vehicle. Their method uses the pattern created by the modulation in the frequency domain to recover the velocity magnitude and direction. This method is fast and robust to changes in climate like rain and snow.

Following the major trend in recent years for forward-facing methods, some approaches have used deep learning for camera pose estimation. In [106], Gilles and Ibrahimpasic propose a camera pose network in an unsupervised learning manner for a downward-facing camera on a differential drive mobile robot. They propose networks

that use both short and longer sequences of images for pose estimation. They leverage the spatial transformer and differentiable bilinear sampling to warp images [107]. In [108], the authors reformulate the common bundle adjustment problem for a downward-facing stereo camera on a mobile platform by using planar motion constraints. And in [109], the authors seek to also estimate the camera's tilt angle to later use for creating a rectified map in a SLAM pipeline. In other works, with forward-facing cameras, the ground plane is used for scale estimation [110,111].

Unfortunately, the current state of the literature on downward-facing cameras in VO is sparse. Many of the existing methods operate in low-speed scenarios, and over short trajectories due to challenges with motion blur. Many of these methods also use special lighting or cameras to make the problem more optimal under varying illumination [94,96-106,108-111]. This research seeks to overcome these challenges while maintaining a simple model.

Chapter 4. Feature Extraction Experiments

In this chapter, a series of experiments for different feature extraction approaches are benchmarked to get a baseline understanding of the challenges that current methods face in contrast to forward-facing VO scenarios. In Experiment 1a, different feature detection and matching methods commonly used in geometric, forward-facing VO pipelines are benchmarked to gauge the robustness of these features on downward-facing camera images. In Experiment 1b, sparse and dense optical flow are tested on downward-facing camera images to demonstrate the effectiveness of different optical flow methods that work well in tracking for forward-facing VO pipelines. In Experiment 1c, a downward-facing VO method for drones is adapted to the HAV setting to gauge its feasibility in estimating planar motion. In Experiment 1d, an end-to-end camera pose network that replaces the classical VO pipeline is proposed to compare how feasible the problem is in comparison to recent forward-facing methods. In Experiment 1e rather than choose feature extractors that are robust to motion blur, motion blur is leveraged for motion estimation. Each section begins with the problem formulation followed by the results of the experiment.

4.1 Problem Setup

These experiments were on data collected from the Carla Simulator [91], Oxford Robotcar Dataset [35], and KITTI Dataset [90]. There are not any popular datasets specifically used for downward-facing cameras in an automated vehicle setting. In the Carla Simulator cameras are fixed in a downward rotation to simulate downward-facing cameras. In the Oxford Robotcar Dataset and KITTI the images are warped from a downward-facing camera to produce a downward-facing view. Every experiment consists of an ego-vehicle with a camera fixed at a height h.

4.1.1 Carla Simulator Dataset

Carla is a simulator designed to emulate autonomous driving scenarios. It was built with Unreal Engine and has support for ROS plugins. In this research Carla is used to generate images of a simulated town and provide ground truth for both a forward-facing and downward-facing camera. The data was collected on prebuilt towns called Town 01 through Town 05. The vehicle is a model of the Tesla Model 3. The front-facing and downward-facing camera are placed at a height of 1.4 meters. At this height the downward-facing camera needs to be moved forward to avoid capturing the hood of the ego-vehicle. For this reason, the downward-facing camera was placed 4 meters ahead. The forward-facing camera was also placed 4 meters ahead to make the transformation between the two cameras a pure rotation. The downward-facing camera's orientation was set to -90 degrees down (Figure 13.)

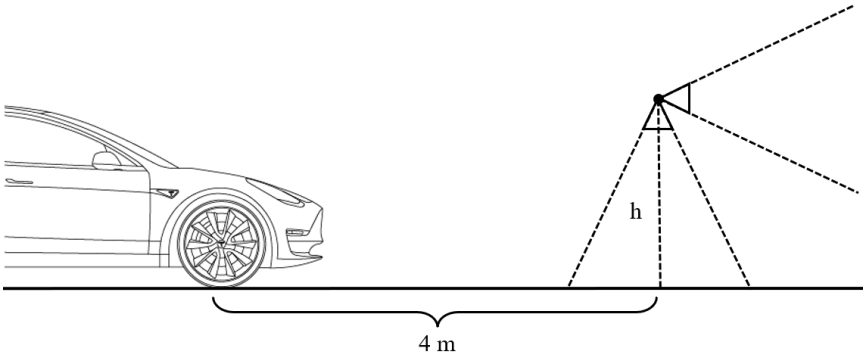


Figure 13. The vehicle setup in the Carla Simulator. [112].

4.1.2 Oxford Robotcar Dataset

The Oxford Robotcar Dataset is a large-scale dataset for autonomous driving [35]. The data was collected over a year, traversing similar routes. The dataset's purpose is to aid in the research and development of long-term localization solutions. It is useful for change detection across seasons, and changing environments (e.g., construction.) The dataset covers over 1000 kilometers of driving across different seasons, weather, and time of day. They also provide a variety of data sources including GPS, RTK, INS, LiDAR, radar, and cameras retrofitted onto a Nissan LEAF ego-vehicle. The Grasshopper2 on the rear of the vehicle is used for experiments in this research (Figure 14.) The camera has a 1:1 aspect ratio at a size of 1024x1024 pixels, making it a good option for observing ground features since the height is sufficiently large in the main axes of motion. The images from the rear camera are warped downward and translated 3.2 meters to view the ground plane. The camera height is 1.44 meters. Unlike the other datasets, the real

vehicle extrinsics show that the camera is pitched down approximately 15 degrees. This is considered warping the camera downward.

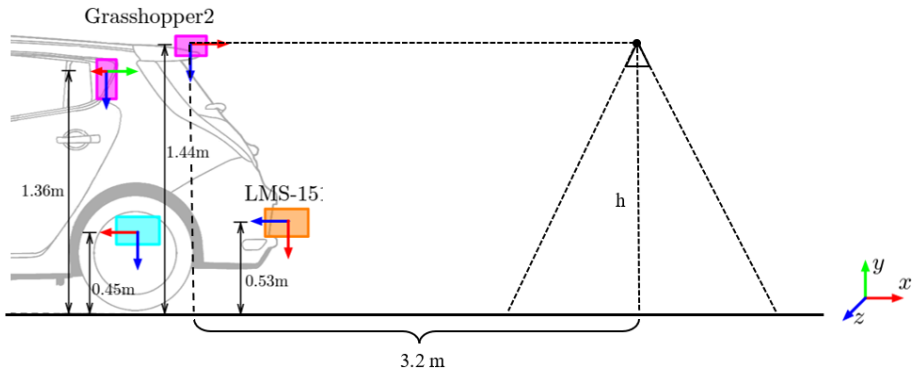


Figure 14. A diagram of the vehicle sensor setup in the Oxford Robotcar Dataset that is relevant for this research [35].

4.1.3 KITTI Dataset

The KITTI Dataset is the most well-known dataset in the VO and SLAM literature [90] as well as other areas in computer vision for automated vehicles (e.g., segmentation, tracking, depth completion.) The dataset has become the standard for the comparison of algorithms. It provides 11 image sequences (00-10) with labeled ground truth. The KITTI benchmark uses the remaining sequences (11-21) to fairly evaluate algorithms on unseen ground truth data. In this research, the KITTI Dataset is used to help assess the quality of the trajectories produced by the proposed methods. The left color stereo camera is used for forward-facing and downward-facing VO methods. The

downward-facing camera is warped virtually downward by translating it 6 meters forward and rotating the camera 90 degrees downward. The height of the forward-facing camera is 1.65 meters, and the downward-facing camera is raised another meter (2.65 meters above ground plane) since the height of the original image is small. This increases the area of the ground plane that is visible in the downward-facing camera (Figure 15.)

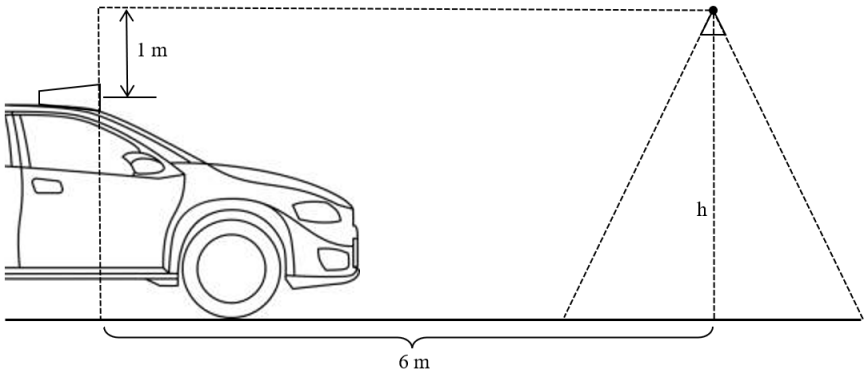


Figure 15. A diagram of the sensor setup for the forward-facing camera in the KITTI Dataset and the virtually warped downward-facing camera [90,113].

4.2 Phase 1 Experiments

In this section there are five experiments that showcase the challenges of extracting features from ground plane images taken from downward-facing cameras. Each experiment is outlined in its problem formulation section where the relevant theory and motivation is discussed. Immediately afterwards the results of each experiment are discussed.

4.2.1 Experiment 1a: Feature Detection and Matching

4.2.1.1 Problem Formulation

In geometric VO pipelines from forward-facing cameras, the classic approach is to extract salient features from the image, and then match these features to form constraints from the epipolar geometry. There are many challenges that can arise from taking this approach. In real world scenarios camera observations can contain motion blur, changing illumination from non-Lambertian surfaces, and moving obstacles. All of these issues can violate the photometric consistency assumption that is used in computer vision. Motion blur distorts images across image frames, making the overall shape of its features inconsistent across frames. Shadows, intense sun rays, and non-Lambertian surfaces can make large portions of an image suddenly dark, saturated, or otherwise inconsistent across image frames. Moving obstacles are not guaranteed to move at the same speed of the ego-vehicle, and can therefore contaminate the motion estimation step, worsening the quality of the pose estimate. These phenomena are well known in the VO literature. To address these issues, different generalized approaches have been proposed to create both unique descriptors and robust outlier rejection to yield the best matches.

In the downward-facing VO setting, motion blur is more severe since the camera's motion and image plane are parallel. At higher speeds, the ego-vehicle experiences large and quick motions within the exposure time of the camera, causing the light that's collected to be blurred. Additionally, the ground plane for most streets and

roads contains low texture, making it difficult for many gradient-based approaches to detect features that can be tracked. Ground plane images can also have images with very fine grain texture that is highly repetitive. High frequency texture is also a challenge in traditional VO settings, because it is difficult to produce descriptors that can distinguish between highly repetitive features that are often observed in man-made environments. The combination of large motions, severe motion blur, and varying degrees of texture frequency make detecting features from downward-facing cameras challenging.

Following this information, several classical feature extraction methods are tested to gauge how severe these issues are. FAST [64], SIFT [52], and ORB [53] feature detectors are the classical feature detection methods chosen to test. The SURF feature detector is currently not available in the commonly used OpenCV library due to its patent, and since its major benefit is speed over SIFT, we exclude it in this experiment. The FAST detector uses ORB descriptors since it is a pure feature detection method. The SIFT and ORB feature detection methods have their own descriptors. For fairness, all approaches use a brute force matcher that matches each point against points in the other image. Since the purpose of this experiment is to gauge the robustness of the feature detection method, faster matching methods are not considered. For outlier rejection, the traditional RANSAC scheme is also used for all methods.

4.2.1.2 Experiment Results

In this experiment a few classic feature detection and matching pipelines and a modern approach are used for feature extraction on the Carla Dataset. The feature

detection and matching pipelines are tested on 15,000 images each. The goal was to gauge how robust each pipeline was to limited road textures. Feature detection, description and matching are foundational to geometric VO pipelines and without robust feature extraction a VO pipeline can't reliably recover the motion of the vehicle. The mean feature count was tracked to quantify how many features were detected. Low feature counts after outlier rejection suggest the feature detectors aren't reliably extracting repeatable features. The standard deviation of the feature count is tracked to quantify the spread of the number of features for the image pairs. A low standard deviation relative to its mean suggests that the detector can consistently extract features from image pairs. The minimum and maximum feature counts are tracked to get a sense of the best- and worst-case scenarios for feature extraction. The failure count is important for understanding how often features can't be extracted from an image pair. During a failure zero features are extracted (or matched) which leads to odometry failure and no motion estimation. The results are shown in Table 1. For fairness and speed, the FAST feature detector is coupled with ORB descriptors. All methods except for the LoFTR matcher use brute force matching. All methods used the classic RANSAC algorithm for outlier rejection.

The FAST feature detector pipeline had the most features extracted at 7,562.7 features extracted on average with an inlier count of 1,046.5 on average. Following this observation, are LoFTR, SIFT, and ORB in that order. The number of features varies largely for most methods relative to their feature count. ORB performed the worst over all by failing to extract features for 361 image pairs. It also failed to produce matches for

1,835 image pairs. Likewise, SIFT had a high failure rate. FAST had few failures for both feature extraction and matching. However, LoFTR was the most consistent, and only had failures for degenerate Carla images as seen in Figure 16, where the Carla Simulator lagged, and the camera simulation incorrectly returned an image from inside the vehicle. Overall, FAST extracts a large quantity of features and is relatively robust to low texture images. However, LoFTR is the most consistent with no failures on non-degenerate images. This suggests that LoFTR is the most suitable for motion estimation over longer sequences where methods like SIFT and ORB don't work well for feature extraction on ground plane images.

Carla Dataset Feature Extraction Results	Feature Count | Inlier Count				
Detector+Descriptor+Matcher	Mean	Std. Dev.	Min	Max	Fails
FAST+ORB+BF	7562.7 | 1046.5	4725 | 881.2	0 | 0	22843 | 5329	4 | 9
SIFT+SIFT+BF	484.3 | 141.9	783.5 | 288.9	0 | 0	12276 | 4676	731 | 1246
ORB+ORB+BF	296.4 | 9.1	126.6 | 9.3	0 | 0	500 | 122	361 | 1835
LoFTR	3211.4 | 3211.4	627.3 | 627.3	3 | 0	4008 | 4008	0 | 2

Table 1. The results for different feature extraction, description, and matching pipelines from classic and a more modern method. The left-hand side has the statistics for the unfiltered features and the right-hand side has the statistics for the inliers. BF: Brute Force Matcher.

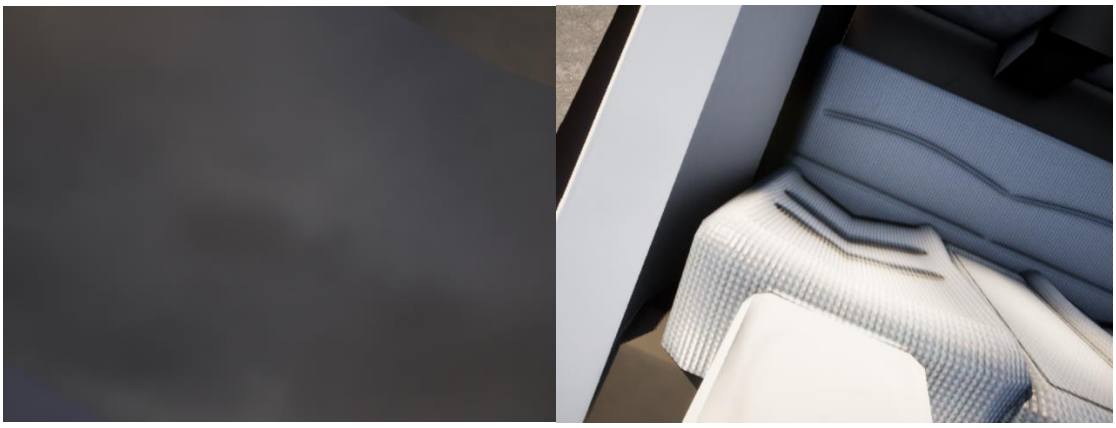


Figure 16. An example failure mode for feature extraction in the Carla Dataset.

4.2.2 Experiment 1b: Sparse and Dense Optical Flow

4.2.2.1 Problem Formulation

In this experiment, optical flow was proposed as a basic approach to recovering camera motion. In a downward-facing camera with a fixed camera height, the motion is parallel to the image plane. The intuition is that if optical flow can accurately recover the change in pixels, then that can be backprojected to the camera's reference frame to recover the camera's pose. In the first experiment, Lucas-Kanade optical flow with Shi-Tomasi features is used to estimate the camera motion. Lucas-Kanade optical flow works on the assumption that little motion occurs across image frames (i.e., photometric consistency.) This can be expressed by the partial differential equation in Eq. 10.1.

$$\frac{\partial I}{\partial x}\frac{\partial x}{\partial t} + \frac{\partial I}{\partial y}\frac{\partial y}{\partial t} + \frac{\partial I}{\partial t} = 0 \tag{10.1}$$

For N point correspondences this can be expressed as an over-constrained matrix equation and can be solved using least squares (Eq. 11.1.)

$$\begin{bmatrix} \frac{\partial I}{\partial x_1} & \frac{\partial I}{\partial y_1} \\ \vdots & \vdots \\ \frac{\partial I}{\partial x_N} & \frac{\partial I}{\partial x_N} \end{bmatrix} \begin{bmatrix} \frac{\partial x}{\partial t} \\ \frac{\partial y}{\partial t} \end{bmatrix} = - \begin{bmatrix} \frac{\partial I}{\partial t_1} \\ \vdots \\ \frac{\partial I}{\partial t_N} \end{bmatrix} \tag{11.1}$$

Eq. 11.1 can be rewritten in a simplified form:

$$\begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} \sum I_x I_t \\ \sum I_y I_t \end{bmatrix} \tag{12.1}$$

$$I_x = \frac{\partial I}{\partial x}, I_y = \frac{\partial I}{\partial y}, I_t = \frac{\partial I}{\partial t}, u = \frac{\partial x}{\partial t}, v = \frac{\partial y}{\partial t} \tag{12.2}$$

If the leading matrix in Eq. 12.1 is invertible and well-conditioned, then the optical flow can be computed. To get the camera pose from the optical flow, the flow is treated as pure pixel translations.

$$P_i = DK^{-1} p_i \tag{13.1}$$

$$P_i = \begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix}, \; p_i = \begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} \tag{13.2}$$

$$P_2 - P_1 = \; DK^{-1}(p_2 - p_1) \tag{13.3}$$

$$\begin{bmatrix} x_2 - x_1 \\ y_2 - y_1 \\ z_2 - z_1 \end{bmatrix} = DK^{-1} \begin{bmatrix} u_2 - u_1 \\ v_2 - v_1 \\ 1 - 1 \end{bmatrix} = h \begin{bmatrix} f_u & 0 & c_u \\ 0 & f_v & c_v \\ 0 & 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} u_2 - u_1 \\ v_2 - v_1 \\ 0 \end{bmatrix} = h \begin{bmatrix} (u_2 - u_1)/f_u \\ (v_2 - v_1)/f_v \\ 0 \end{bmatrix} \tag{13.4}$$

$$\begin{bmatrix} dx \\ dy \\ dz \end{bmatrix} = h \begin{bmatrix} u/f_u \\ v/f_v \\ 0 \end{bmatrix} \tag{13.5}$$

As seen in Eq. 13.5, the translation of the ego-vehicle can be estimated with optical flow given the camera height and focal length. For dense optical flow, rather than extracting sparse feature points to track, the pixels are used to directly calculate optical flow. In [114], Farneback, dense optical flow works by assuming that pixels undergo a locally quadratic motion model (Eq. 14.1-2).

$$f_i(x) = x^T A_i x + b_i^T x + c_i \tag{14.1}$$

$$f_{i+1}(x) = f_i(x - d) = (x - d)^T A_{i+1}(x - d) + b_{i+1}^T (x - d) + c_{i+1} \tag{14.2}$$

A displacement field, d introduced to this model creates a linear constraint as seen in Eq. 15.1.

$$A(x)d(x) = \Delta b(x) \tag{15.1}$$

This problem can be solved using weighted least squares. Under an affine motion model this problem becomes:

$$d(x) = Sp = \begin{bmatrix} 1 & x & y & 0 & 0 & 0 & x^2 & xy \\ 0 & 0 & 0 & 1 & x & y & xy & y^2 \end{bmatrix} [a_1 \quad a_2 \quad \dots \quad a_7 \quad a_8]^T \tag{16.1}$$

The new objective for the weighted least squares optimization becomes:

$$\sum w \| A_i d(x) - \Delta b_i \|^2 \tag{17.1}$$

The solution for this objective:

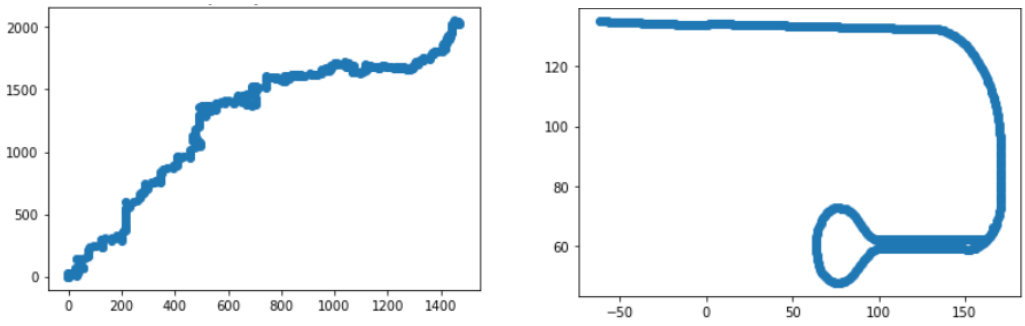
$$p = (\sum w_i S_i^T A_i^T A_i S_i)^{-1} \sum w_i S_i^T A_i^T \Delta b_i \tag{18.1}$$

The dense optical flow approach from Farneback [114] is used for the initial camera motion estimate in pixels. A histogram of pixels for magnitude and angle is created and the bin with the maximum magnitude and angle count is used for the optical flow estimate. Using backprojection as seen in Eq. 13.5, the camera's pose in the camera's reference frame can be calculated.

4.2.2.2 Experiment Results

For this experiment, the Lucas-Kanade Optical Flow method was used for feature extraction. The sparse optical flow method implemented in OpenCV [126] was used for experimentation. A maximum of 100 corners and a minimum distance of 0.1 was used. A partial trajectory of Town 03 in the Carla Dataset was used to test the proposed method. In Figure 17, the estimated trajectory is seen on the left, and the ground truth is seen on the right. The camera pose estimate is qualitatively poor. This is likely because the Carla

Dataset has large translations between image frames. Consequently, the motion estimates are poor for vanilla optical flow. In Experiment 1a, for the Carla Dataset, feature extraction was prone to failures for some methods. This experiment motivates the need to also have a robust motion estimation scheme when feature detection and matching is possible.



(a) (b)

Figure 17. The Lucas-Kanade Optical Flow method with Shi-Tomasi Corners: (a) Estimated Trajectory (b) Ground Truth on Carla Town 03.

Next, Farneback Dense Optical Flow was used to estimate the camera motion. For many of the images in the Carla Dataset, there are both high and low frequency textures. Despite the absence of motion blur, this alone made it difficult to extract features in previous experiments. A sample image pair can be seen in Figure 18. The vehicle is moving forward as can be observed from the movement of the rough patches on the ground. In Figure 18, the dense optical flow is shown with images for its magnitude and angle. Despite the clear transition, the magnitude of the optical flow is low. In smoother

regions, the optical flow is estimated to be approximately zero. The angle measurements are highly varying and appear to be random. Both observations can be seen from the histograms in Figure 19. The magnitude is highly contaminated by zero-valued flow pixels, making it hard to extract the correct magnitude, even when camera motion can be visually observed within the image (Figure 20a.) For the angle histogram, there are two distinct peaks that don't correspond to the correct angle of flow (Figure 20b.) This behavior was shown across many image runs. For this reason, dense optical flow is considered not to be a feasible approach for estimating a camera's pose from downward-facing cameras.

Figure 18. A pair of images for the dense optical flow input.

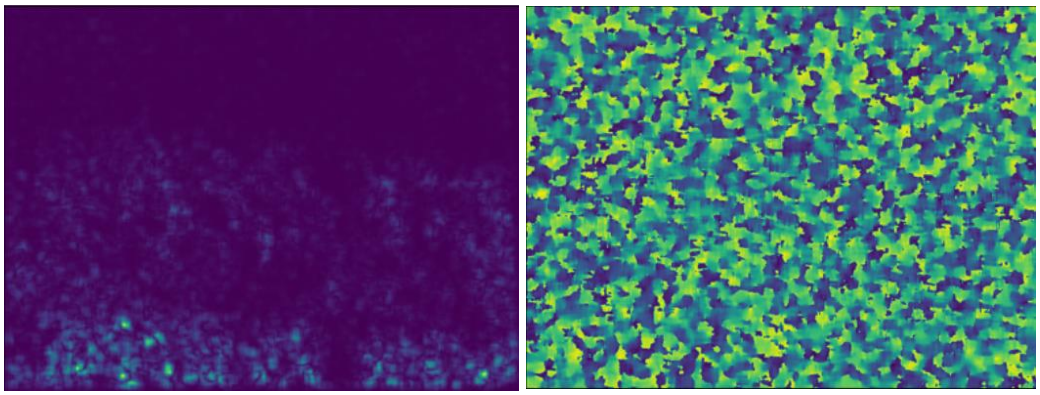
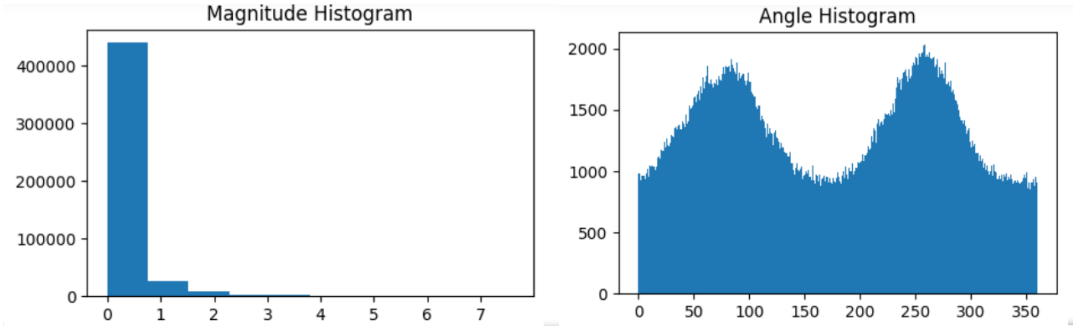


Figure 19. The magnitude of the optical flow (left) and the angle (right) for the image. Dark blue represents small magnitude or angle. Bright green represents large magnitude or angle.

.



(a) (b)

Figure 20. The histogram for the magnitude (a) and angle (b) of the optical flow.

4.2.3 Experiment 1c: Power Spectrum Analysis

4.2.3.1 Problem Formulation

In Experiments 1a and 1b, sparse and dense features were used for feature extraction for the downward-facing VO problem. This approach attempts to build on the work in [92], that uses power spectrum analysis for the localization of a drone. Their method also uses a downward-facing, monocular camera (Figure 21a.) The major innovation for this approach is the use of the cross-power spectrum to estimate the motion of the camera and drone. Given an image pair that is related by a planar motion,

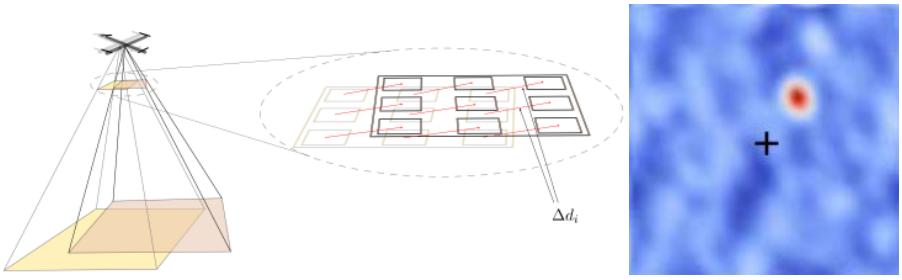
their Fourier Transforms are calculated. A translation (i.e., shift) in the spatial domain is equivalent to modulating the frequency domain by a complex exponential. These Fourier Transform images are related in this way in two dimensions, where u and v are the change in pixels across the horizontal and vertical axes of the image plane (Eq. 19.1.)

$$I_A(\omega_x, \omega_y) = e^{-j(u\omega_x + v\omega_y)} I_B(\omega_x, \omega_y) \tag{19.1}$$

The displacement is obtained by calculating the cross-power spectrum of images I_A and I_B (Eq. 20.1.) The inverse Fourier Transform is the Dirac impulse (u, v) pixels away from the origin (Figure 16b.) The displacement is obtained by finding the maximum of the inverse Fourier Transform (Eq. 20.2.) This approach is called Phase Correlation Matching [92]. The image is divided into patches, and their image centers and the locations of the Dirac impulses are used as points to calculate a Homography. The Homography is decomposed to get the pose estimate [92].

$$\frac{I_A I_B^*}{|I_A||I_B|} = e^{-j(u\omega_x + v\omega_y)} \tag{20.1}$$

$$F^{-1}\left[e^{-j(u\omega_x + v\omega_y)}\right] = \delta(x - u, y - v) \tag{20.2}$$



(a) (b)

Figure 21. (a) A diagram of the drone and camera setup. (b) An image of the cross-power spectrum for an image patch pair [92].

4.2.3.2 Experiment Results

In this experiment, the Carla Dataset was used to generate downward-facing camera images and ground truth trajectories. These images are free of motion blur, with data containing both low and high frequency texture. Figure 22 shows an example of an image pair that has road markings and shadows, and subtle texture that should have sufficient features to track motion. From the lane marking it is clear to see that the vehicle is moving.

Algorithms 1-3 in [92] were directly applied both patch-wise and for entire image pairs. In Figure 23, example images of the power spectrum magnitude are shown. Both power spectrums are concentrated in the middle of the image with the first image appearing to be slightly rotated. These power spectral images are common for many of the images in the Carla Dataset. An inverse FFT sample is shown in Figure 24. In Figure 24a, the expected output is shown. Following the results of [92], there should be a small signal resembling the 2D Dirac impulse. However, many of the images tested appear like

Figure 24b., where no Dirac impulse is present. As a result, no motion was observed for any images, despite the presence of motion in the images. This is likely because these images are relatively smooth and lack significant structures. Even for images with lane markings, it is difficult to gauge how far the camera has moved. In a drone setting, buildings, trees, cars, and other objects might be clear indicators of motion within the image. However, it appears that for the automated vehicle setting, this approach does not work.

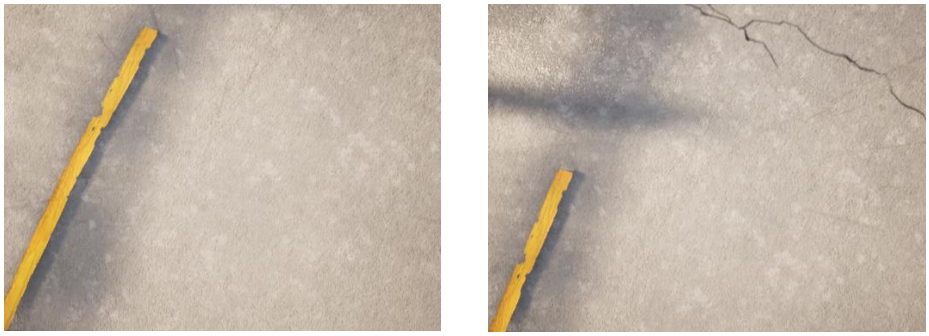


Figure 22. A sample image pair used from the Carla Dataset.

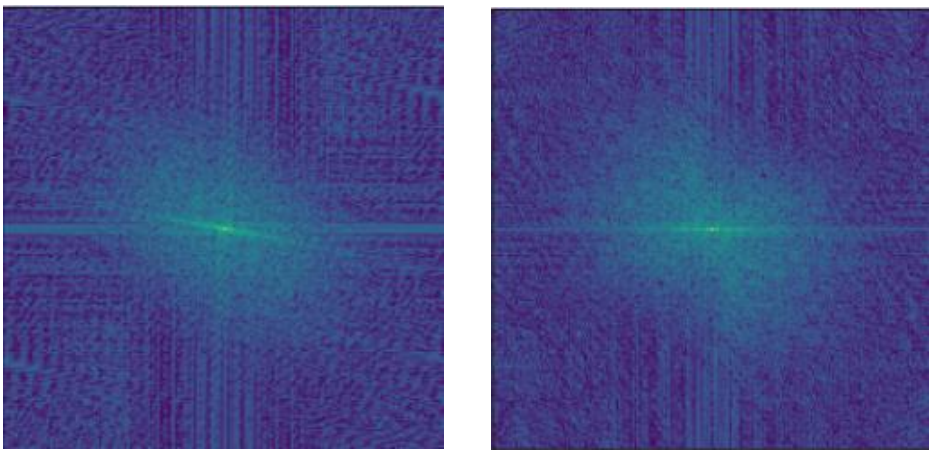
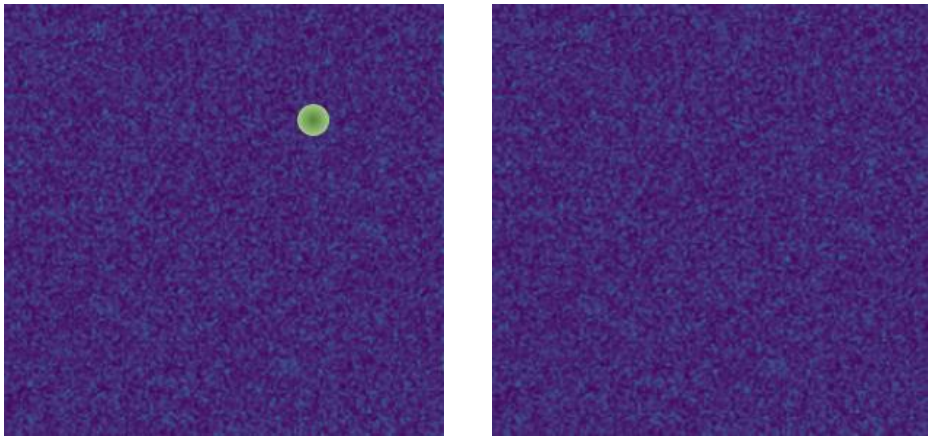


Figure 23. A sample image pair with power spectrum magnitude displayed.



(a) (b)

Figure 24. (a) The expected inverse FFT of the power spectrum. (b) The output for many of the inverse FFTs.

4.2.4 Experiment 1d: End-to-End Deep Learning

4.2.4.1 Problem Formulation

Camera pose networks have been increasingly more common in the VO literature. Particularly for their robustness against photometric inconsistencies caused by non-Lambertian surfaces, moving obstacles, and more. They're also easier to deploy because they require much less fine tuning of parameters after training compared to geometric and direct VO pipelines that often require fine tuning of parameters for their environments of operation. Unsupervised camera pose networks have shown to be the most robust to photometric inconsistencies. Many of these networks jointly learn camera pose and depth through variations of the image warping loss (e.g., [28].) In downward facing VO in the automated vehicle setting, the height is fixed, and the camera is observing the ground

plane. The camera's depth is approximately constant from frame to frame and equal to the camera's height. For this reason, it is not useful to learn depth for downward facing VO. In [106], an unsupervised learning approach is used to estimate the odometry in SE(2). Their method uses the spatial transformer [107] to learn the yaw angle. They propose two networks that use a single image pair and another that uses five images to estimate the pose (Figure 25.) Likewise, in this work, methods that learn camera pose from CNN encoders were considered. For the proposed network the approach in DeepVO was considered [25]. DeepVO is an early case in the VO literature of a supervised learning approach for camera pose networks. It consists of a CNN encoder, LSTM modules, and MLP output layer. CNN encoders are common neural network architectures that are effective at learning filters for local dependencies in images and have shown much success across classification, segmentation, monocular depth estimation [46], camera pose networks [28], and more. LSTM modules retain memory of their hidden states over long sequences, and in DeepVO they are used to learn longer term dependencies for the CNN features [25]. The MLP network is used to transform the LSTM output to the lie algebra of the SE(3) manifold. For this experiment, the DeepVO model architecture is used to train the camera pose network (Figure 26.) Rather than learning in a supervised manner, the common image warping loss [28] is used (Eq. 21.1.) that calculates the L1 loss and Structural Similarity Index Metric (SSIM) [115] between the target and warped source images. This experiment is used as a baseline for deep learning under the downward-facing VO scenario.

$$L = (1-\alpha)\left\|I_2(p) - I_1\big(w(K,d,p)\big)\right\|_1 + \alpha SSIM\left(I_2, I_1\big(w(K,d,p)\big)\right) \quad (21.1)$$

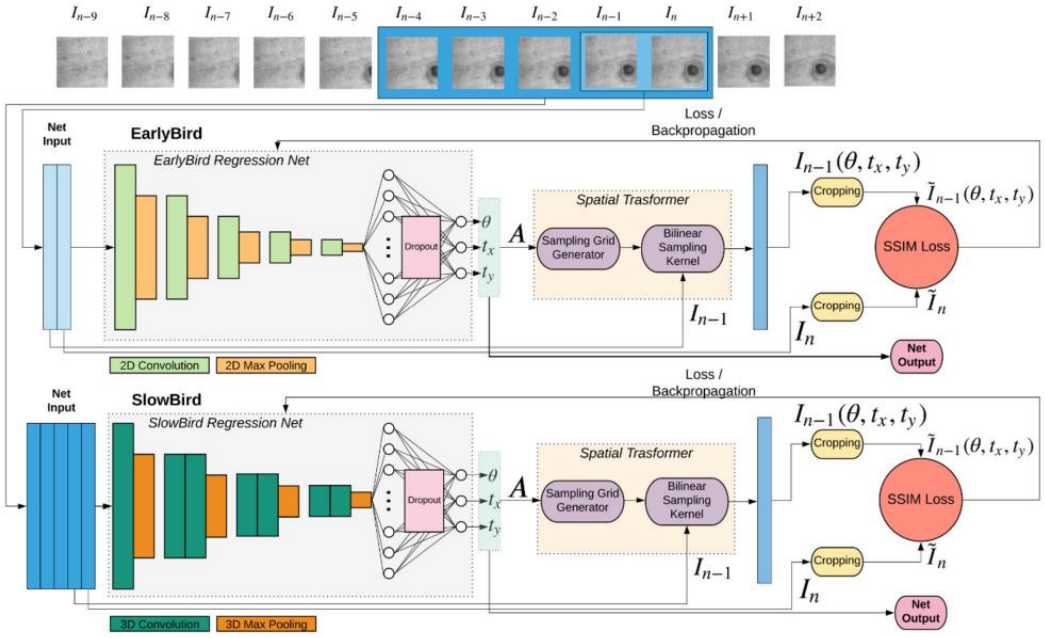


Figure 25. A diagram of the architecture in [106] for their unsupervised camera pose network from a downward-facing camera. The EarlyBird network uses two image pairs to regress camera pose, and the SlowBird network uses five image pairs to regress camera pose.

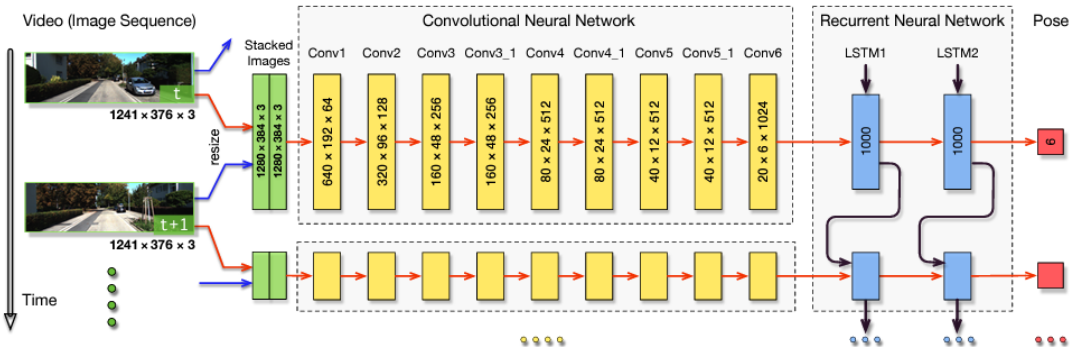


Figure 26. A diagram of DeepVO as outlined in [25].

4.2.4.2 Experiment Results

In this experiment, an unsupervised camera pose network is proposed following the work of the supervised learning method, DeepVO [25]. DeepVO was chosen, because unlike many unsupervised camera pose networks, DeepVO doesn't require a monocular depth estimation network. For downward-facing VO the depth is approximately constant. The loss function in [106] is used to learn the camera pose in an unsupervised manner. Instead of learning the lie algebra of SE(3), the lie algebra of SE(2) is learned. Following the work of [127], instead of attempting to train the heading directly, the cosine and sine of the heading were learned to mimic the elements of the 2D rotation matrix. This yields an output for $\cos(\theta)$, $\sin(\theta)$, t_x, and t_y. The model can be seen in Table 2.

Layer Name	Input Features	Output Features	Kernel Size	Stride
Conv3d	1	16	(2,3,3)	(1,1,1)
MaxPool3d	-	-	(1,2,2)	(1,1,1)
Conv3d	16	32	(2,3,3)	(1,1,1)
MaxPool3d	32	64	(1,3,3)	(1,1,1)
Conv3d	64	128	(1,3,3)	(1,1,1)
MaxPool3d	-	-	(1,2,2)	(1,1,1)
Conv3d	128	256	(1,3,3)	(1,1,1)
Conv3d	-	-	(1,2,2)	(1,1,1)
LSTM cell	181	1024	-	-
LSTM cell	181	1024	-	-
Linear	1024	4	-	
Linear	67072	1	-	-

Table 2. The final proposed network architecture.

The model is trained on downward-facing camera images from the Carla Dataset. This dataset was chosen because the images are relatively simple in structure and do not contain motion blur. The dataset contained 30,000 images from Town 01-04. Town 05 was left for testing after training. This dataset is similar in size to the training split for many camera pose networks trained on KITTI. The images were resized to 150x200 to be sufficiently small for the network. Larger images can have redundant information that

isn’t useful for learning at the price of more computation. The network is fed a pair of three images. The model is trained with the Adam optimizer [128] with a learning rate of 1e-4 for 10 epochs. The qualitative results can be seen in Figure 30.

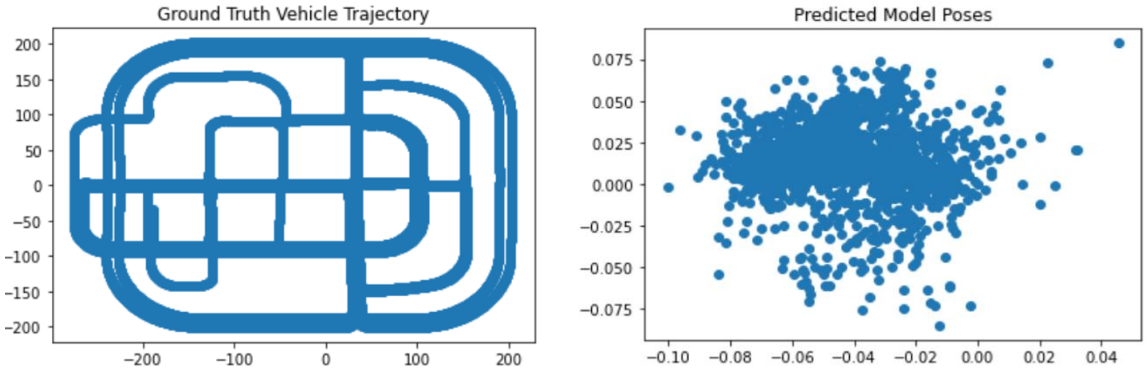


Figure 27. Qualitative results for the proposed camera pose network. On the left is the Carla Ground Truth, and on the right is the predicted model trajectory.

In Figure 27, it is clear that the trained model didn’t learn the camera pose well. This is likely because the model was not trained on as many trajectories as [106]. Due to the high number of output features in the LSTM cells, this could have also led to overfitting, that was not a clear concern during training. For future work, the LSTM model could be replaced with the spatial transformer as originally proposed in [106]. Another issue was the availability of the data and length of the trajectories. It is likely that although forward-facing methods can learn pose with only 30k images, for downward-facing methods it might require much more data as suggested by [106]. In the future, more trajectories can be used for training in addition to more epochs.

4.2.5 Experiment 1e: Motion from Blur

4.2.5.1 Problem Formulation

In previous experiments, there were major challenges with attempting to extract features from downward-facing images. A major source of these challenges seemed to be the presence of severe motion blur. The final experiment in Phase 1 seeks to leverage motion blur to estimate the pose of the ego-vehicle, rather than seek methods that are robust to motion blur. There are a few works that use motion blur to estimate optical flow or camera pose [116-118]. In general, computer vision tasks seek to deblur images to improve the quality of images for their respective problems. Methods like DeblurGAN [119] and DeblurGAN v2 [120] achieve this through adversarial learning. Where models like SelfDeblur [121] jointly learn a deblurring module and optical flow network to deblur images in an automated vehicle setting. These methods were tested and failed to deblur ground plane images in the Oxford Robotcar Dataset. This is likely because the images were out of distribution for the training samples of the network. Many of the images used have significant structures (e.g., people, buildings) that might also make it easier to reconstruct the original image. For downward-facing cameras a lot of the blur is of fine-grained particles, possibly making it harder to deblur.

There are at least two VO pipelines that leverage motion blur to estimate camera pose. The first is Motion Blur Aware Visual Odometry (MBA-VO) [122], that reblurs images instead of deblurring them. By reblurring the pose can be estimated through direct photometric optimization of the blurry and reblurred image. However, this method is

designed for indoor use where motion blur is inconsistent, and often from sudden jerks of the camera. For this reason, there are blur free images present to help reblur the image. In the automated vehicle setting there is consistent, severe blur over long trajectories, which makes this approach impractical for this research. In World from Blur [117] an ensemble of networks for deblurring, depth estimation and camera pose estimation are used to reconstruct a world from blurry images. Again, the use of deblurring modules doesn't seem to be well suited for this setting, and this approach is left for future work.

In [118], a simple method for extracting optical flow from a blurred image is proposed. In this method the authors observe that an image blurred in a planar motion creates a rotated 2D sinc function in the frequency domain. This is because, for simple particles that are blurred, they take on the form of rectangles. The uniform rectangular function is a sinc in the frequency domain. Following this observation, they use steerable gaussian filters (Table 3) to extract the orientation of the blur (Eq. 22.1-4), and then use the cepstrum to calculate the magnitude of the blur, producing the optical flow of the blurred image. This method was chosen as a baseline for its ability to recover motion from blurred images with simple particles.

The optical flow estimation starts by applying a gaussian blur to the image to help eliminate artefacts at the edge of the image in the frequency domain. Then, the power spectrum is calculated (Eq. 22.5), and a steerable gaussian filter is exhaustively applied to the image for angles 0 through 180 degrees. The angle with the maximum response is used for the orientation. Then the cepstrum is calculated of the 1D slice of the image set at the extracted orientation. The minimum of the cepstrum is the magnitude of the motion

blur. Similar to Experiment 1b, the optical flow is backprojected to get a motion estimate (Eq. 13.5.) To speed up rotation estimation instead of exhaustively searching for theta, the search is done with Gauss-Newton Optimization and restricted to [0,2π] (Eq. 23.1-5.) The overall system is shown in Figure 28.

$G_{2a} = 0.921(2x^2 - 1)e^{\mu}$	$k_a(\theta) = \cos^2(\theta)$
$G_{2b} = 1.843xye^{\mu}$	$k_b(\theta) = -2\cos(\theta)\sin(\theta)$
$G_{2c} = 0.921(2y^2 - 1)e^{\mu}$	$k_c(\theta) = \sin^2(\theta)$
$\mu = -(x^2 + y^2)$	

Table 3. The set of Gaussian basis filters, G_{2x} and steerable kernels, k_x [118].

$$RG_{2a} = \sum_{x,y} G_{2a}(x,y)FImg(x,y) \quad (22.1)$$

$$RG_{2b} = \sum_{x,y} G_{2b}(x,y)FImg(x,y) \quad (22.2)$$

$$RG_{2c} = \sum_{x,y} G_{2c}(x,y)FImg(x,y) \quad (22.3)$$

$$RG_2^{\theta} = k_a(\theta)RG_{2a} + k_b(\theta)RG_{2b} + k_c(\theta)RG_{2c} \quad (22.4)$$

$$\text{Power Spectrum, P} = \log\left(FT\{I(x,y)\} * FT\{I(x,y)\}^*\right) \quad (22.5)$$

$$\theta^{k+1} = \theta^k + \delta\theta, \quad (23.1)$$

$$\delta\theta = -(J^T J)^{-1} J^T r(\theta^k), \quad (23.2)$$

$$r(\theta^k) = -\left(RG_2^{\theta}\right)^2,\ J = \frac{\partial r(\theta^k)}{\partial \theta^k} \quad (23.3)$$

$$\delta\theta = \frac{-r(\theta^k)}{\nabla r(\theta^k)} = -\frac{1}{2}\frac{RG_2^\theta}{\nabla RG_2^\theta}, \tag{23.4}$$

$$\theta^{k+1} = \theta^k - \frac{1}{2}\mu\frac{RG_2^\theta}{\nabla RG_2^\theta} \tag{23.5}$$

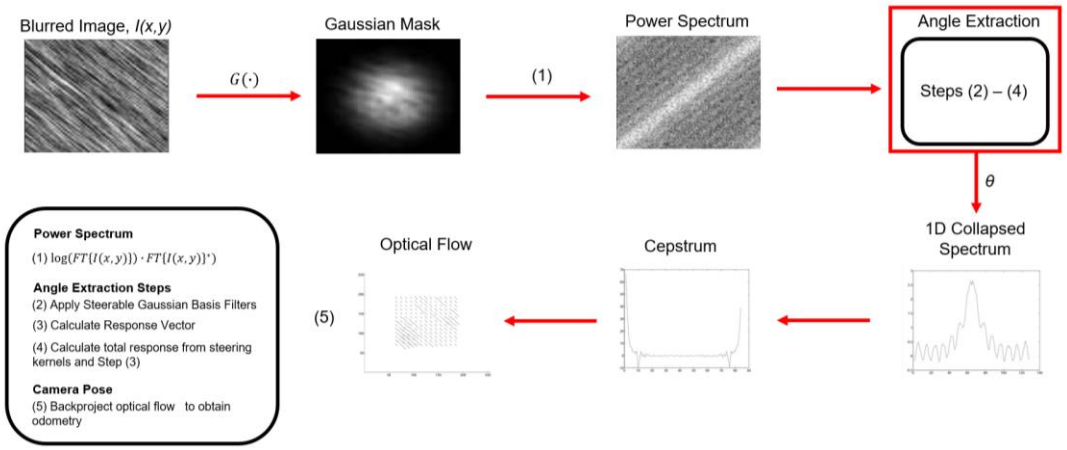


Figure 28. A diagram of the motion from blur baseline illustrated in [118].

4.2.5.2 Experiment Results

Previous experiments show that the presence of varying texture and motion blur make it difficult to recover motion. Several of the classic approaches to solving this problem don't produce accurate features, flow, or camera pose. In this experiment, rather than finding robust features to reduce the impact of motion blur, it is used to estimate camera motion instead. A baseline experiment for the approach in [118] is reproduced. The image presented in the work is a motion blurred image of noise (Figure 29.) This is similar to downward-facing camera images of a road because many of the blurred

particles are fine grain in nature. The algorithm is run on patches of the sample image and the entire image.

The original image has size 256x256. For the patch-wise experiment 64x64 image patches are used to estimate the flow of the image. As seen in Figure 30, the optical flow results for the patch-wise approach are spread from 44-47 degrees. This is expected based on the reported error in [118]. For the full image 45 degrees exactly is estimated. This illustrates that for simple motion blurred images with planar motion, this method can produce reasonable estimates of the orientation of the optical flow.

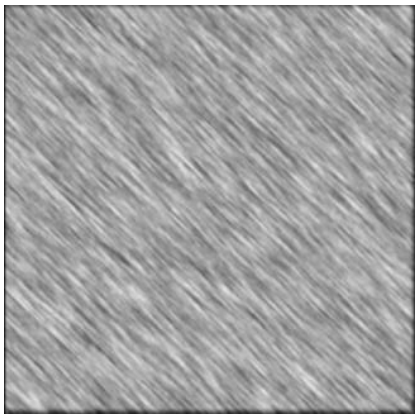


Figure 29. The sample image presented in [118] that is used for the baseline experiment.

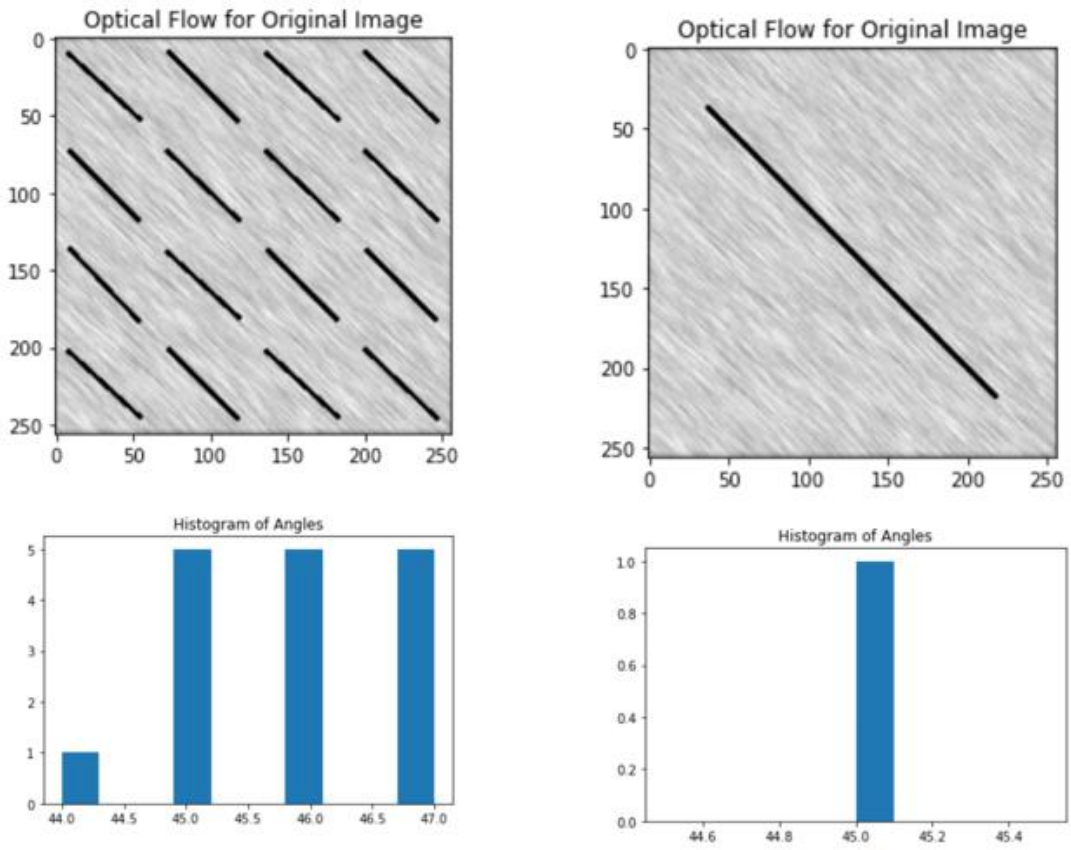


Figure 30. Baseline experiment results for patch-wise and whole image optical flow estimation. Results are consistent with the results in [118].

This approach was also repeated with data from the Oxford Robotcar Dataset. Rear-facing camera images were taken from sequence 2015-10-30-13-52-14 and warped downward to produce downward-facing camera images. The images have size 1024x1024. For the patch-wise experiment 256x256 kernels are used. The image in Figure 31 shows significant motion blur, which is common for this dataset. The vehicle is moving forward on an urban road in Oxford.

Figure 31. A sample image from the Oxford Robotcar Dataset warped downward.

For the Oxford Robotcar Dataset, similar results can be observed in this baseline experiment (Figure 32.) The vehicle is moving forward, and the algorithm estimates that the vehicle is moving with 0 or 180 degrees of rotation. Throughout these experiments, this ambiguity between 0 and 180 degrees can be observed. This is because at 0 and 180 degrees the max response is the same. For the whole image experiment, only 180 degrees is estimated.

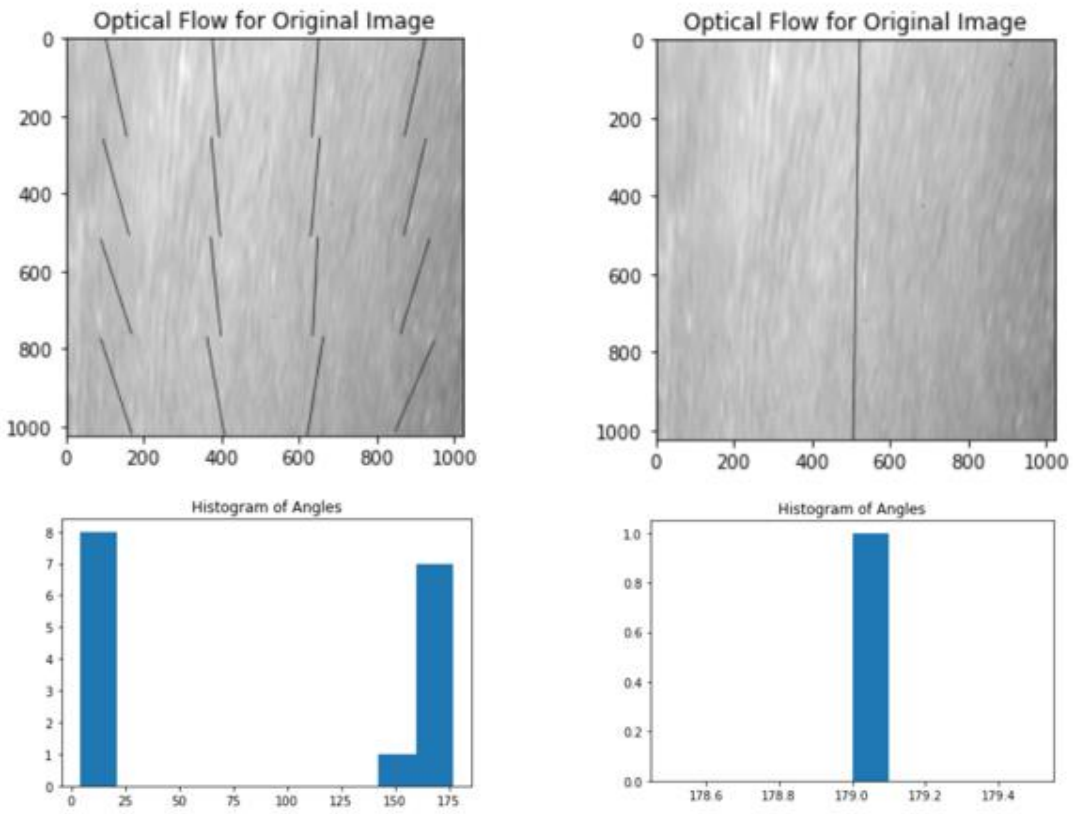


Figure 32. Baseline experiment results for the Oxford Robotcar Dataset.

In another experiment, the magnitude estimation is evaluated. To estimate the magnitude of the flow, the cepstrum of the image was calculated. In this experiment, Carla images were artificially blurred by averaging and warping intermediate images along the desired change in pose. This way the exact flow magnitude is known. As can be seen in Figure 33, the magnitude of the flow is the first peak after the initial signal attenuates.

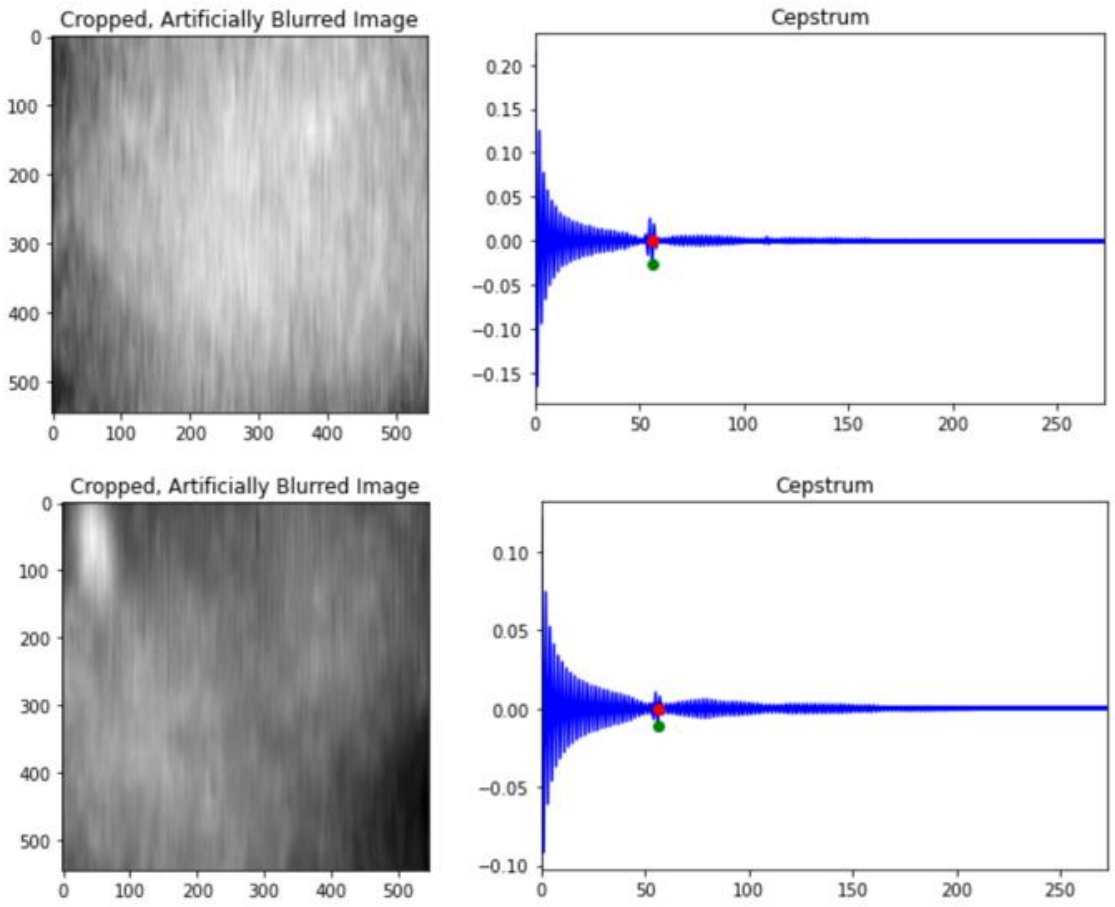
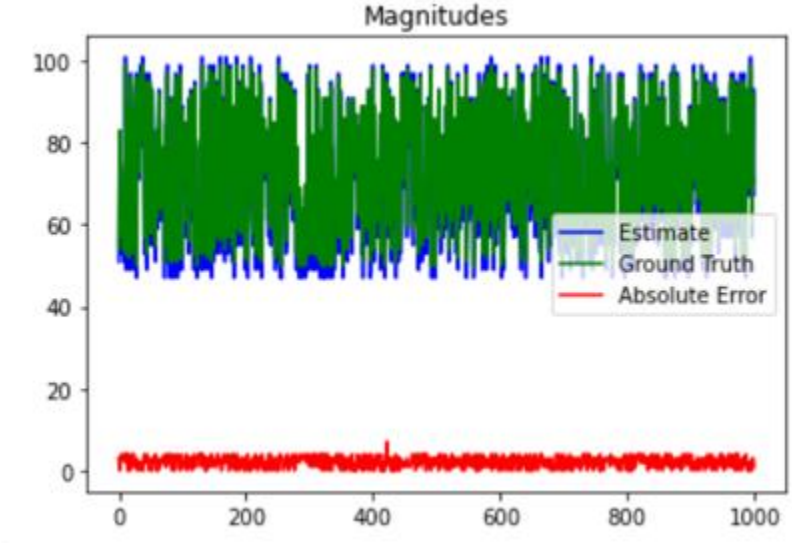


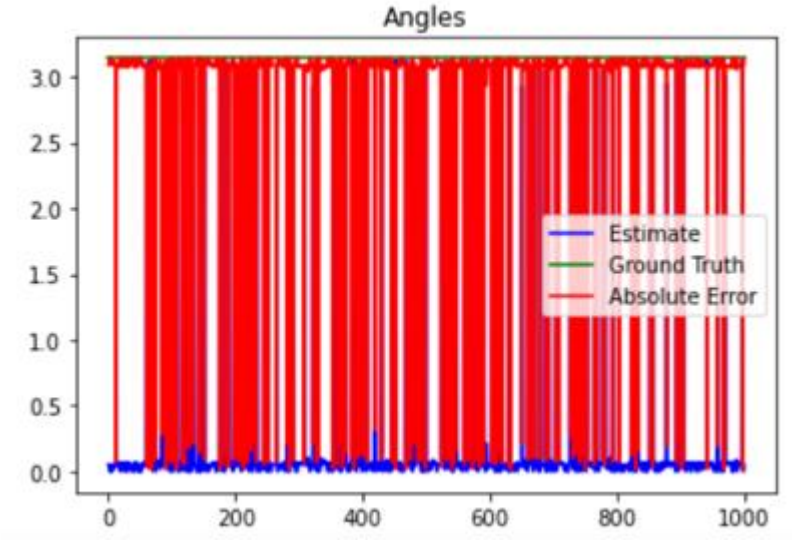
Figure 33. Artificially blurred Carla Images with the minimum of their cepstrum shown.

This process was done over 1,000 synthetically blurred images. The translation was done in the y axis of the image plane, varying by 50-100 pixels with 180 degrees of rotation. The mean error for the magnitude was 2.096 pixels which is equivalent to 0.72 cm in the camera's reference frame. The standard deviation was 1.293 pixels (0.46 cm). The mean error for the orientation was 154.8 degrees with a standard deviation of 58.3

degrees. The angle orientation is high, because of the ambiguity between 0 and 180 degrees. The results can be seen in Figure 34.



(a)



(b)

Figure 34. The results of synthetically blurring 1000 Carla images from a downward-facing camera and measuring the frame's optical flow from motion blur. (a) Magnitude Estimation and (b) Angle Estimation Results.

The magnitude estimation works well for a vehicle that is moving in a straight line, however, when the vehicle moves in both the x and y axes (e.g., a turn) the approach begins to breakdown. This method was run on the Oxford Robotcar Dataset for a subset of the 2015-10-30-13-52-14 sequence. The flow was backprojected as in Eq. 13.5. And a constant velocity ego-vehicle model was used to estimate the odometry from optical flow in a single image. As seen in Figure 35, the rotation of the trajectory is not recovered. This might be because the real images have shadows, and other varying illumination changes and sources of blur. This can introduce more low frequency content in the image, making the response function the largest at 0 or 180 degrees consistently. For this reason, this method is also not a good choice for estimating odometry from downward-facing cameras.

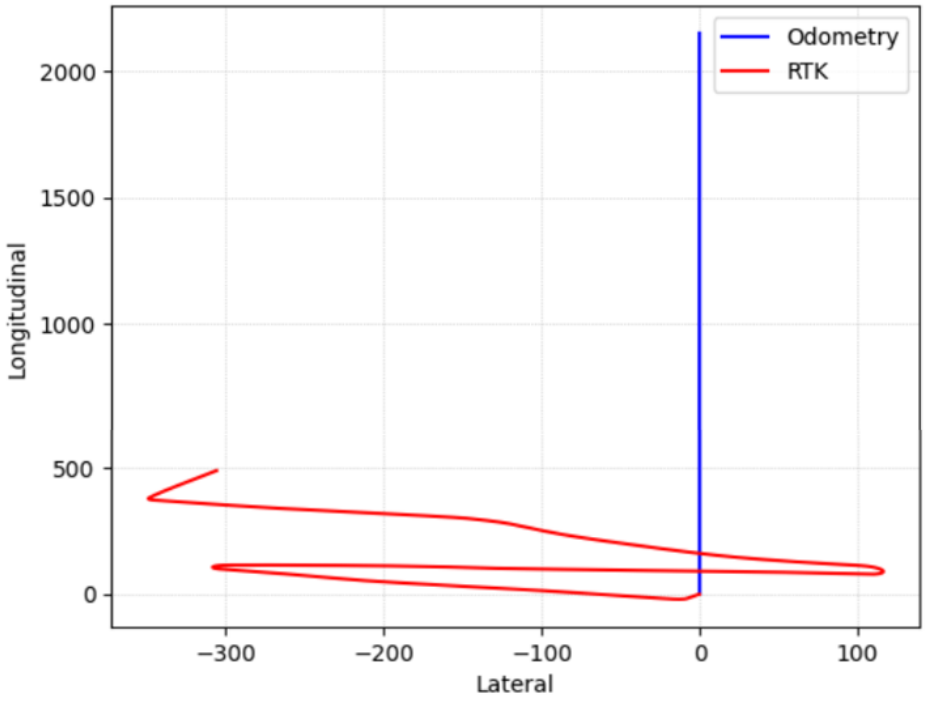


Figure 35. A sample subsequence ran on the Oxford Robotcar Dataset, Sequence: 2015-10-30-13-52-14.

4.3 Summary

Overall, these experiments show that extracting features from ground plane images is a challenging problem. In Experiment 1a classical feature detection, description and matching pipelines that used ORB and SIFT failed to produce reliable features for simulated Carla images. However, FAST with ORB descriptors was robust throughout the experiment with few failures and a high inlier match count on average. The deep learning matcher LoFTR was the most consistent, producing feature matches with few outliers and failure cases only during simulation failures. In Experiment 1b sparse optical flow failed to produce accurate trajectories. This is because for downward-facing camera images there

are large motions from frame to frame. In the forward-facing case the axes of the image plane experience little motion making it adequate for feature tracking. For dense optical-flow there were many zero-motion flow estimates in smooth regions and random phase estimates that made it difficult to extract the correct flow estimate. In Experiment 1c the power spectrum analysis approach failed to produce Dirac impulses to recover the motion estimate. Unlike drone images with significant structures from buildings, cars, vegetation and other obstacles, ground plane images have little structure and varying degrees of texture making it difficult to recover motion with this method. In Experiment 1d an end-to-end camera pose network was proposed like DeepVO but trained in an unsupervised manner. This approach also failed to produce an adequate trajectory. Lastly, in Experiment 1e rather than finding features robust to motion blur, motion blur was leveraged to recover motion. For simple images proposed by the original work and synthetically blurred images showed promising results. However, for real world images this approach failed to produce accurate orientation estimates. Following the observations of Experiments 1a-e in the next chapter, the deep learning matcher LoFTR is used for a geometric VO pipeline from a downward-facing camera.

Chapter 5. Deep Learning based Feature Detection and Matching

In this chapter, the observations from experiments detailed in Chapter 4 will be used to create a geometric VO pipeline from a downward-facing camera. Many of these experiments failed to produce features or trajectories that were sufficient for a VO pipeline. However, in Experiment 1a the deep learning matcher LoFTR was consistent and robust compared to other feature detection, description and matching pipelines. For this reason, it is used to produce robust feature correspondences for the pipeline. These features are used in optimization scheme to recover motion estimates for the odometry. The following sections detail the problem formulation and results for the experiment.

5.1 Problem Formulation

In Phase 1, several experiments were conducted to showcase the challenges associated with VO from downward-facing cameras. Feature extraction methods from both geometric and direct approaches under varying paradigms struggled with producing useful information for this research problem. In this experiment, a geometric approach for VO from downward-facing cameras is proposed. The method utilizes more recent advances in deep feature detection, description, and matching. The detector-free matching approach called Local Feature Transformer (LoFTR) is used for robust feature

extraction and matching in the VO pipeline [56]. The LoFTR module uses self-attention and cross-attention layers popularized by the Transformer architecture [123]. A vanilla attention layer utilizes query, key, and value vectors as input to an attention model. The query vector retrieves information from the value vector based on the dot product of the query and key vectors. Elements of the value vector are weighted by the similarity of the query and key vectors. This is generally how attention works (Figure 36.) LoFTR extracts features with its multi-level, coarse-to-fine Local Feature CNN (Figure 37.) The coarse layer is flattened, and positional encodings are applied. Positional encodings give positions a unique sinusoidal format. This is important for LoFTR to successfully match in regions that aren't distinctive [56]. These features are passed into the LoFTR Module, and that output is passed to the differentiable matching layer [56]. The Coarse-to-Fine Module takes the fine features from the Local Feature CNN and passes them to a LoFTR Module. These output features are correlated (center vector of first set of features to all other feature vectors) and softmax is applied. This produces a heat map that can be used to localize matches in the matching module. This approach can achieve subpixel correspondence matches.

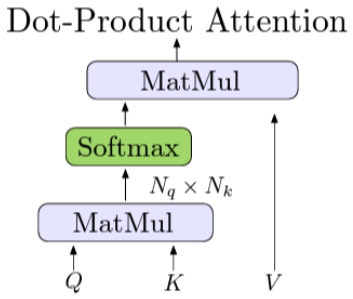


Figure 36. Basic attention layer for transformers [56,123].

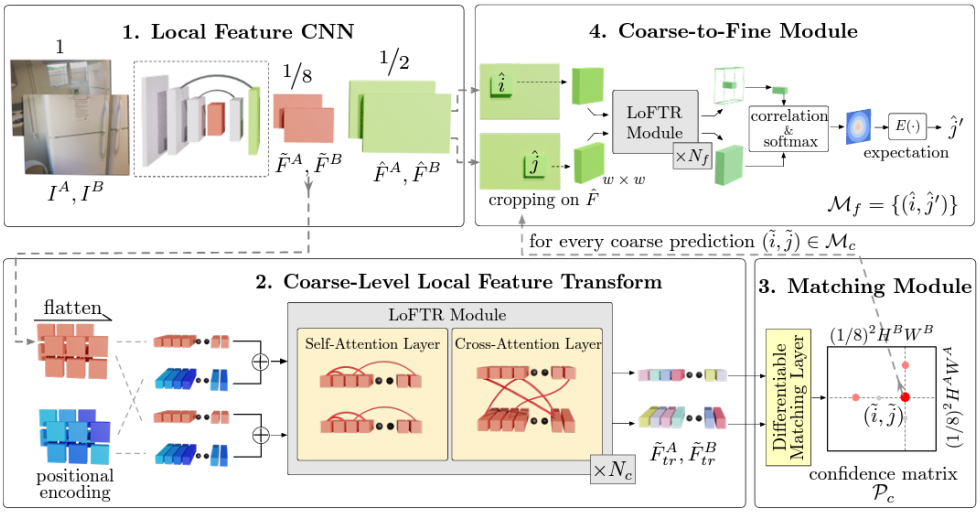


Figure 37. Overall architecture of the LoFTR framework proposed in [56]. This approach is used to match features in the geometric VO front end of this thesis research.

The LoFTR matcher is robust to low texture regions, repetitive features, and motion blur unlike many classic feature detection, description, and matching pipelines. This makes it a great candidate for feature extraction and matching in a geometric VO pipeline from downward-facing cameras. The model is also open source and available through the Kornia computer vision library [124] for Pytorch [125]. The outlier rejection used for this pipeline is MAGSAC++.

As discussed earlier, traditional Essential Matrix decomposition is degenerate for planar motion and pure rotation scenarios. For this reason, estimating the Fundamental Matrix from LoFTR matches isn't feasible for the downward-facing camera setting. The Homography Matrix is the motion estimation model chosen for this research, because of these reasons. The depth of the camera is fixed at the camera height, and the normal vector of the observed plane is assumed to be $n = [0, 0, -1]^T$. Instead of using the Direct

Linear Transform [47] and then decomposing the Homography into multiple pose solution [47] like traditional approaches, the problem is reformulated as a weighted least squares problem (Eq. 24.1-3.) Due to the robustness of LoFTR, outliers don't tend to dominate the matching process, but are still present and can be observed by outliers in the correspondence distances. For this reason, the correspondences are weighted based on how close they are to the mean (i.e., residual error). This technique is used in direct VO pipelines to reject outlier poses [28]. It is similar to down weighting the correspondences by their inverse covariance. Rather than calculate pose and scale separately like some methods [111], the pose and scale are directly calculated through the optimization problem (Eq. 24.1.)

It's common in VO pipelines to use a strategy for keyframe selection [17]. Keyframe selection is the process of choosing target frames that vary significantly from the source frame. Without keyframe selection, image pairs with little relative motion can lead to odometry that is computationally inefficient and odometry with significantly more error drift. In this experiment the keyframes for the odometry are calculated when the weighted average of the point correspondence distance exceeds a threshold. In the experiments this is generally set to 5 pixels since this tends to translate to an average velocity of greater than 1 mph and suggests significant motion.

$$C = \sum_i^p \omega_i \left(p_{2,i} - H(T,K,n,h)p_{1,i}\right)^2 \tag{24.1}$$

$$H(T,K,n,h) = K\left(R - \frac{tn^T}{h}\right)K^{-1} \tag{24.2}$$

$$\omega_i = \frac{1}{1+\|\Sigma_{res}\|_2^2} \tag{24.3}$$

5.2 Experiment Results

In the Phase 1 experiments of this research, there were many challenges to estimating odometry from ground plane images. Feature point extraction, optical flow, power spectral analysis, and a basic deep learning approach were not feasible solutions for estimating camera pose. However, the LoFTR feature matching approach in Experiment 1a showed robustness across 15,000 images in the Carla Dataset, with only failures from severe glitches in the simulator. For this reason, this matching approach is used again to formulate a geometric VO pipeline for downward-facing cameras.

The LoFTR detector-free matching approach leverages transformers to match deep features. The matcher is robust to motion blur and varying texture frequency. Similar to geometric bundle adjustment, inlier feature point matches are used in a weighted optimization problem to estimate the odometry directly from a reprojection error formulated for Planar Homographies, similar to [129]. The Oxford Robotcar Dataset is used for estimating odometry. The height is fixed at 1.44m and the plane normal vectors are all set to $n = [0\ ,0,\ -1]^T$. The cost function in Eq. 24.1 is minimized using the Levenberg-Marquardt algorithm [47] which is common in Homography estimation problems, in contrast to Gauss-Newton optimization for typical optimization problems that handle pose estimation. In early experiments, Gauss-Newton failed to minimize the error and remained in its initial local minima. The optimization was

implemented using the Theseus optimization library for Pytorch [130]. The max number of iterations was set to 150 with a learning rate of 0.05. Pose regularization was applied to reduce divergence with a damping factor of 0.005.

The trajectory results can be seen in Figure 38. The trajectory is measured over 200 samples, similar in length to [108] and longer than [106]. The overall trajectory is smooth and lacks large jumps like [106]. Unfortunately, it is difficult to compare downward-facing VO methods in detail. Many methods do not release their datasets, and don't use full trajectories like KITTI which are the standard. This is because most downward-facing VO methods drift quickly, and don't produce accurate trajectories over several hundred or thousands of meters. In [106], their differential drive robot only traverses trajectories less than 4 meters. And in [108] their trajectories can produce odometry at similar lengths to this work but are more unstable. Despite the smoothness of the trajectories for this approach it still has trouble recovering accurate rotation. In future work, a more robust initialization can be applied from a forward-facing camera. This might leverage the benefit of more accurate rotation from forward-facing methods, and the benefit of relatively accurate translation estimation from downward-facing methods. This all can be done from a single camera since this method virtually projects each camera downward.

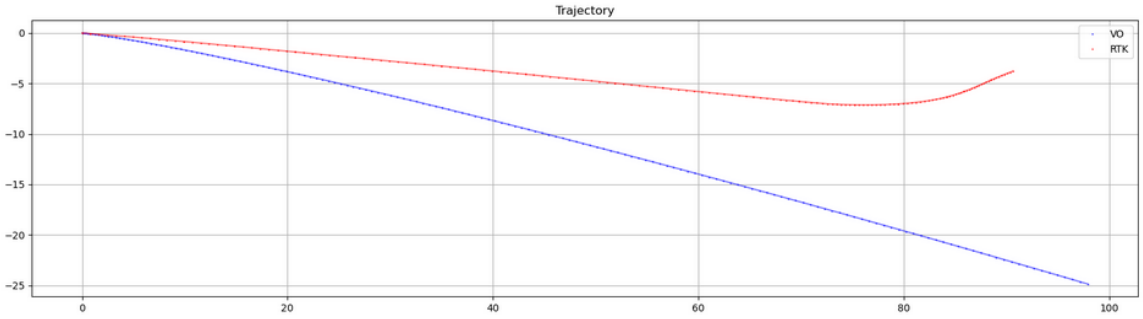


Figure 38. An estimated trajectory for the Oxford Robotcar Dataset, Sequence: 2015-10-30-13-52-14.

Chapter 6. Conclusion and Future Work

6.1 Conclusion

Visual Odometry has been a popular research topic for the last 40 years. Odometry alone has shown robustness across different sensor modalities including LiDAR and vision. In the automated vehicle setting, VO can provide an alternative to sensors like GPS that are prone to large errors in the presence of urban canyons and poor weather. Recent approaches focus on deep learning and the estimation of uncertainty from forward-facing cameras. In this research, VO from downward-facing cameras is revisited in an attempt to produce a simple yet robust alternative. VO from downward-facing cameras has applications across many robotic platforms including UAVs, AUVs, and UGVs due to the relative location of prominent features in their respective environments. However, in the automated vehicle setting, this is particularly challenging due to sever motion blur and varying texture.

In this thesis, several approaches for estimating a camera's pose from a downward-facing camera were explored. In Experiment 1a, many classical approaches failed to produce reliable features for simulated images in Carla. A deep-learning approach to feature point matching called LoFTR made it possible to extract features from downward-facing camera images for both simulated and real images. In Experiment 1b, sparse and dense optical flow were used to estimate camera motion. Both methods

proposed for the frontend of a VO pipeline produced inaccurate flow estimates. For the sparse method, large motions made it difficult to accurately produce the correct flow. In the dense flow experiment many zero-motion estimates made it difficult to estimate motion in general. In Experiment 1c, a power spectral analysis method for drones failed to produce any pose estimates. Likely due to the lack of structure in ground plane images. In Experiment 1d, a deep learning approach like DeepVO, but trained under an unsupervised learning scheme is proposed. This method also failed to produce accurate camera pose estimates. This was likely due to insufficient data and overfitting. In Experiment 1e, a motion from blur front end was proposed, but failed to produce rotation estimates on real images.

In Phase 2, a geometric VO pipeline was produced based on the observations from Phase 1 experiments. This includes the selection of LoFTR matching and an optimization scheme for Planar Homography estimation. This method didn't fail to extract features from images unlike methods in Phase 1. It produced smooth trajectories in contrast to other downward-facing cameras. However, rotation estimates were still not accurate and significant drift was observed over time.

Overall, the VO problem from downward-facing cameras is still ill-posed and challenging. However, the use of cross-view information from forward-facing methods and downward-facing methods in an automated vehicle setting may be an interesting topic for future research.

6.2 Future Work

As previously discussed, an interesting direction might be using both forward-facing methods and downward-facing methods for VO from a single camera. The emergence of robust deep learning matching schemes like LoFTR make this possible as shown in this research for the downward-facing view. The combination of these views might lead to more accurate estimation, similar to other multi-view odometry methods.

Another interesting direction might be the use of the downward-facing approach in Experiment 2 to create a GPS anomaly detection pipeline. Since the odometry is still locally accurate up to centimeter level, it could still be a sufficient reference in the presence of large GPS errors that might exceed several meters.

It also might be interesting to incorporate this Homography estimation approach in a loss function for a deep camera pose network to enforce consistency across a forward-facing camera and downward-facing camera. This intuition could also work for pose graphs where the downward-facing camera is an additional factor.

Bibliography

[1] E. Roberts, “Robotics: A Brief History,” Stanford University Department of Computer Science. [Online]. Available: https://cs.stanford.edu/people/eroberts/courses/soco/projects/1998-99/robotics/history.html

[2] D. Pomerleau, “ALVINN: An Autonomous Land Vehicle in a Neural Network,” Carnegie Mellon University, NeurIPS, 1989. [Online]. Available: https://papers.nips.cc/paper_files/paper/1988/hash/812b4ba287f5ee0bc9d43bbf5bbe87fb-Abstract.html

[3] Defense Advanced Research Projects Agency, “The Grand Challenge.” DARPA. Available: https://www.darpa.mil/about-us/timeline/-grand-challenge-for-autonomous-vehicles

[4] S. A. Bagloee, M. Tavanna, M. Asadi, T. Oliver, “Autonomous vehicles: challenges, opportunities, and future implications for transportation policies,” in *the Journal of Modern Transportation*, 2016, pp. 284-303. [Online]. Available: https://link.springer.com/article/10.1007/s40534-016-0117-3

[5] M. Callegari, S. Brillarelli, C. Scoccia, “Archimedes, Vitruvius and Leonardo: The Odometer Connection,” in the Advances in Historical Studies, 2020, pp. 330-343. [Online]. Available: https://www.researchgate.net/publication/347407325_Archimedes_Vitruvius_and_Leonardo_The_Odometer_Connection

[6] National Park Service, “Mormon Odometer,” National Park Service. [Online]. Available: https://www.nps.gov/articles/000/mormon-odometer.htm

[7] M. Parker, “Chapter 20: Automotive Radar,” in *Digital Signal Processing 101*, 2nd ed. Elsevier, 2017, pp.253-276. Accessed: Apr, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780128114537000202

[8] D. Tazartes, “An Historical Perspective on Inertial Navigation Systems,” in International Symposium on Inertial Sensors and Systems (ISISS), 2014. [Online]. Available: https://ieeexplore.ieee.org/document/6782505

[9] National Aeronautics and Space Administration, “Global Positioning System History,” NASA, 2012. [Online]. Available: https://www.nasa.gov/directorates/heo/scan/communications/policy/GPS_History.html

[10] S. A. S. Mohamed, M. Haghbayan, T. Westerlund, J. Heikkonen, H. Tenhunen, J. Plosila, “A Survey on Odometry for Autonomous Navigation Systems,” in IEEE Access, Aug. 2019. [Online]. Available: https://ieeexplore.ieee.org/document/8764393

[11] Tong Qin, Jie Pan, Shaozu Cao, and Shaojie Shen, “A general optimization-based framework for local odometry estimation with multiple sensors,” arXiv preprint arXiv:1901.03638, 2019.

[12] Stefan Leutenegger, Simon Lynen, Michael Bosse, Roland Siegwart, and Paul Furgale, “Keyframe-based visual-inertial odometry using nonlinear optimization,” The International Journal of Robotics Research, 34(3):314–334, 2015.

[13] Michael Bloesch, Sammy Omari, Marco Hutter, and Roland Siegwart, “Robust visual inertial odometry using a direct EKF-based approach,” In 2015 IEEE/RSJ international conference on intelligent robots and systems (IROS), pages 298304. IEEE, 2015.

[14] Anastasios I Mourikis and Stergios I Roumeliotis, “A multistate constraint Kalman filter for vision-aided inertial navigation,” In Proceedings 2007 IEEE International Conference on Robotics and Automation, pages 3565–3572. IEEE, 2007.

[15] E. B. Quist and R. W. Beard, ‘‘Radar odometry on fixed-wing small unmanned aircraft,’’ IEEE Transactions on Aerospace and Electronic Systems, vol. 52, no. 1, pp. 396–410, Feb. 2016.

[16] C. Doer, G. F. Trommer, “Radar Visual Inertial Odometry and Radar Thermal Inertial Odometry: Robust Navigation even in Challenging Visual Conditions,” in 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Oct. 2021. Prague, Czech Republic.

[17] D. Scaramuzza, F. Fraundorfer, “Visual Odometry Part I: The First 30 Years and Fundamentals,” in the IEEE Robotics & Automation Magazine, Dec. 2011.

[18] Raul Mur-Artal, Jose Maria Martinez Montiel, and Juan D Tardos, “ORB-SLAM: a versatile and accurate monocular SLAMsystem,” in the IEEE Transactions on Robotics, 31(5):11471163, 2015.

[19] Raul Mur-Artal and Juan D Tardos, “ORB-SLAM2: An open-source slam system for monocular, stereo, and rgb-d cameras,” in the IEEE Transactions on Robotics, 33(5):1255–1262, 2017.

[20] Carlos Campos, Richard Elvira, Juan J. Gómez Rodríguez, José M. M. Montiel and Juan D. Tardós, “ORB-SLAM3: An Accurate Open-Source Library for Visual, Visual-Inertial and Multi-Map SLAM,” in the *IEEE Transactions on Robotics 37(6):1874-1890, 2021*.

[21] A. Geiger, J. Ziegler, C. Stiller, “Stereoscan: Dense 3D reconstruction in real-time,” in the Intelligent Vehicles Symposium (IV), 2011, pp. 963–968.

[22] Jakob Engel, Thomas Schops, and Daniel Cremers, “LSDSLAM: Large-scale direct monocular SLAM,” In the European Conference on Computer Vision (ECCV), pages 834–849. Springer, 2014.

[23] Jakob Engel, Vladlen Koltun, and Daniel Cremers, “Direct sparse odometry,” in the IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI), 2017.

[24] Christian Forster, Matia Pizzoli, and Davide Scaramuzza, “SVO: Fast semi-direct monocular visual odometry,” In the 2014 IEEE International Conference on Robotics and Automation (ICRA), pages 15–22, 2014.

[25] S. Wang, R. Clark, H. Wen, N. Trigoni, “DeepVO: Towards End-to-End Visual Odometry with Deep Recurrent Convolutional Neural Networks,” in the IEEE International Conference on Robotics and Automation (ICRA), 2017.

[26] R. Li, S. Wang, Z. Long, D. Gu, "UnDeepVO: Monocular Visual Odometry through Unsupervised Deep Learning," in the IEEE International Conference on Robotics and Automation (ICRA), 2018.

[27] T. Zhou, M. Brown, N. Snavely, D. G. Lowe, "Unsupervised Learning of Depth and Ego-Motion from Video," in the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017.

[28] N. Yang, L. von Stumberg, R. Wang, D. Cremers, "D3VO: Deep Depth, Deep Pose and Deep Uncertainty for Monocular Visual Odometry," in the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2020.

[29] Erwin Kruppa, "Zur Ermittlung eines Objektes aus zwei Perspektiven mit innerer Orientierung", Sitzungsberichte der Mathematisch-Naturwissenschaftlichen Kaiserlichen Akademie der Wissenschaften, Vol. 122 (1913), pp. 1939–1948.

[30] G. Gallego, E. Mueggler, P. Sturm, "Translation of 'Zur Ermittlung eines Objektes aus zwei Perspektiven mit innerer Orientierung' by Erwin Kruppa (1913)," arXiV, 2017. [Online]. Available: https://arxiv.org/abs/1801.01454

[31] H. Moravec, "Obstacle avoidance and navigation in the real world by a seeing robot rover," Ph.D. dissertation, Stanford Univ., Stanford, CA, 1980.

[32] United States Department of Transportation, "Driver Assistance Technologies," National Highway Traffic Safety Administration. Accessed: Apr, 2023. [Online]. Available: https://www.nhtsa.gov/equipment/driver-assistance-technologies?utm_source=paid_search&utm_medium=cpc&utm_campaign=adas_2023&utm_content=rearautomaticbraking_search&gclid=CjwKCAjwitShBhA6

EiwAq3RqA38KrdviyhxBb-vaM9VqED8STBxhXl66XEbc_TGkL7P1PlqafYyoyRoCUQwQAvD_BwE

[33] J. Janai, F. Guney, A. Behl, A. Geiger, "Computer Vision for Autonomous Vehicles: Problems, Datasets and State of the Art," arXiV, 2021. [Online]. Available: https://arxiv.org/abs/1704.05519

[34] M. Burri, J. Nikolic, P. Gohl, T. Schneider, J. Rehder, S. Omari, M. W. Achtelik, and R. Siegwart, "The EuRoC micro aerial vehicle datasets," The International Journal of Robotics Research, vol. 35, no. 10, pp. 1157–1163, 2016.

[35] W. Maddern, G. Pascoe, C. Linegar and P. Newman, "1 Year, 1000km: The Oxford RobotCar Dataset", *The International Journal of Robotics Research (IJRR)*, 2016.

[36] Intel RealSense, "Intel RealSense Depth Camera D435f," Intel. Accessed: Apr. 2023. [Online]. Available: https://www.intelrealsense.com/depth-camera-d435f/

[37] R. B. Benosman, "Event Computer Vision 10 years Assessment: Where We Came From, Where We Are and Where We Are Heading To," in the IEEE Conference on Computer Vision and Pattern Recognition Workshop (CVPRW), 2021. [Online]. Available: https://tub-rip.github.io/eventvision2021/

[38] Teledyne FLIR, "FLIR Launches Second Generation Thermal Camera for Self-Driving Cars," Jan. 2019. [Online]. Available: https://www.flir.com/news-center/press-releases/flir-launches-second-generation-thermal-camera-for-self-driving-cars/

[39] M. Schonbein, T. Strauss, A. Geiger, “Calibrating and Centering Quasi-Central Catadioptric Cameras,” in the IEEE International Conference on Robotics and Automation (ICRA), 2014.

[40] Waymo, “Meet Waymo One,” Accessed: Apr. 2023. [Online]. Available: https://waymo.com/

[41] A. J. Yang, C. Cui, I. A. Barsan, R. Urtasun, S. Wang, “Asynchronous Multi-View SLAM,” arXiV, 2021. Accessed: Apr. 2023. [Online]. Available: https://arxiv.org/abs/2101.06562

[42] Y. Zhou, “Event-based Visual Odometry: A Short Tutorial,” in the IEEE Conference on Computer Vision and Pattern Recognition Workshop (CVPRW), 2021. [Online]. Available: https://tub-rip.github.io/eventvision2021/

[43] H. Rebecq, T. Horstschaefer, G. Gallego, D. Scaramuzza, “EVO: A Geometric Approach to Event-Based 6-DOF Parallel Tracking and Mapping in Real-time,” in the IEEE Robotics and Automation Letters (RAL), 2016.

[44] A. R. Vidal, H. Rebecq, T. Horstschaefer, D. Scaramuzza, “Ultimate SLAM? Combining Events, Images, and IMU for Robust Visual SLAM in HDR and High Speed Scenarios,” arXiV, 2017. Accessed: Apr 2023. [Online]. Available: https://arxiv.org/abs/1709.06310

[45] M. O. A. Aqel, M. H. Marhaban, M. I. Saripan, N. B. Ismail, “Review of visual odometry: types approaches, challenges and applications,” Springerplus, 2016. Accessed: Apr. 2023. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5084145/

[46] C. Godard, O. M. Aodha, M. Firman, G. Brostow, “Digging Into Self-Supervised Monocular Depth Estimation,” in the IEEE International Conference on Computer Vision (ICCV), 2019.

[47] R. Hartley, A. Zisserman, “Multiple View Geometry in Computer Vision,” 2nd ed. Cambridge University Press, 2003.

[48] A. J. Davison, I. D. Reid, N. D. Molton, O. Stasse, “MonoSLAM: Real-Time Single Camera SLAM,” in the IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI), vol. 29, no. 6, 2007.

[49] G. Klein, D. Murray, “Parallel Tracking and Mapping for Small AR Workspaces,” in the 6th IEEE and ACM International Symposium on Mixed and Augmented Reality, 2007.

[50] J. L. Schonberger, J. Frahm, “Structure-from-Motion Revisited,” in the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.

[51] H. Bay, T. Tuytelaars, L. Van Gool, “SURF: Speeded Up Robust Features,” in the European Conference on Computer Vision (ECCV), 2006. Springer. [Online]. Available: https://link.springer.com/chapter/10.1007/11744023_32

[52] D. G. Lowe, “Object recognition from local scale-invariant features,” in the International Conference on Computer Vision (ICCV), 1999. [Online]. Available: https://ieeexplore.ieee.org/document/790410

[53] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: An efficient alternative to SIFT or SURF," in the International Conference on Computer Vision (ICCV), 2011, pp. 2564-2571.

[54] C. Harris, M. Stephens, "A combined corner and edge detector," in the Proceedings of the 4th Alvey Vision Conference, 1988. pp. 147-151.

[55] Jianbo Shi and Tomasi, "Good features to track," *1994 Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, Seattle, WA, USA, 1994, pp. 593-600, doi: 10.1109/CVPR.1994.323794.

[56] J. Sun, Z. Shen, Y. Wang, H. Bao and X. Zhou, "LoFTR: Detector-Free Local Feature Matching with Transformers," *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Nashville, TN, USA, 2021, pp. 8918-8927, doi: 10.1109/CVPR46437.2021.00881.

[57] X. Zhang and A. L. P. Tay, "Fast Learning Artificial Neural Network (FLANN) Based Color Image Segmentation in R-G-B-S-V Cluster Space," *2007 International Joint Conference on Neural Networks*, Orlando, FL, USA, 2007, pp. 563-568, doi: 10.1109/IJCNN.2007.4371018.

[58] M. A. Fischler, R. C. Bolles, “Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography,” Communications of the ACM, vol. 24, no. 6, June, 1981, pp. 381-395. https://doi.org/10.1145/358669.358692

[59] J. Davis (2021). “Interest Points,” Computer Vision for HCI (CSE 5524) [Powerpoint slides].

[60] J. Sola, J. Deray, D. Atchuthan, “A micro Lie theory for state estimation in robotics,” arXiV, 2018. [Online]. Available: https://arxiv.org/abs/1812.01537

[61] J. Davis (2021). “Template Matching,” Computer Vision for HCI (CSE 5524) [Powerpoint slides].

[62] J. Ma, X. Jiang, A. Fan, J. Jiang, J. Yan, “Image Matching from Handcrafted to Deep Features: A Survey,” in the International Journal of Computer Vision, 2021, pp. 23-79. Springer. [Online]. Available: https://link.springer.com/article/10.1007/s11263-020-01359-2

[63] L. Zhou, G. Huang, Y. Mao, S. Wang and M. Kaess, "EDPLVO: Efficient Direct Point-Line Visual Odometry," 2022 International Conference on Robotics and Automation (ICRA), Philadelphia, PA, USA, 2022, pp. 7559-7565, doi: 10.1109/ICRA46639.2022.9812133.

[64] E. Rosten, T. Drummond, “Machine Learning for High-Speed Corner Detection,” in the European Conference on Computer Vision (ECCV), 2006. [Online]. Available: https://link.springer.com/chapter/10.1007/11744023_34

[65] M. Calonder, V. Lepetit, C. Strecha, P. Fua, “BRIEF: Binary Robust Independent Elementary Features,” in the European Conference on Computer Vision (ECCV), 2010. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-642-15561-1_56

[66] B. D. Lucas, T. Kanade, “An Iterative Image Registration Technique with an Application to Stereo Vision,” in the Proceedings of the 7th International Joint Conference on Artificial Intelligence (IJCAI), 1981. [Online]. Available: https://www.researchgate.net/publication/215458777_An_Iterative_Image_Registration_Technique_with_an_Application_to_Stereo_Vision_IJCAI

[67] K. M. Yi, E. Trulls, V. Lepetit, P. Fua, “LIFT: Learned invariant feature transform,” in the European Conference on Computer Vision (ECCV), 2016.

[68] D. DeTone, T. Malisiewicz, A. Rabinovich, “Toward geometric deep slam,” arXiV, 2017. [Online]. Available: https://arxiv.org/abs/1707.07410

[69] M. Dusmanu, I. Rocco, T. Pajdla, M. Pollefeys, J. Sivic, A. Torii, T. Sattler, “D2-Net: A trainable cnn for joint detection and description of local features,” in the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2019.

[70] J. Revaud, P. Weinzaepfel, C. De Souza, N. Pion, G. Csurka, Y. Cabon, M. Humenberger, “R2D2: repeatable and reliable detector and descriptor,” in the Proceedings of the 33rd International Conference on Neural Information Processing Systems (NeurIPS), 2019.

[71] M. Tyszkiewicz, P. Fua, E. Trulls, “DISK: Learning local features with policy gradient,” in the Proceedings of the 34th International Conference on Neural Information Processing Systems (NeurIPS), 2020.

[72] X. Li, K. Han, S. Li, V. Prisacariu, “Dual-resolution correspondence networks,” in the Proceedings of the 34th International Conference on Neural Information Processing Systems (NeurIPS), 2020.

[73] T. Li, B. Jiang, Z. Tu, B. Luo, J. Tang, “Image Matching Using Mutual k-Nearest Neighbor Graph,” in the International Conference of Young Computer Scientists, Engineers and Educators (ICYCSEE), 2015, pp. 276-283. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-662-46248-5_34

[74] D. Scaramuzza, F. Fraundorfer, R. Siegwart, “Real-Time Monocular Visual Odometry for On-Road Vehicles with 1-Point RANSAC,” in the IEEE International Conference on Robotics and Automation, 2009. [Online]. Available: https://ieeexplore.ieee.org/document/5152255

[75] R. Raguram, O. Chum, M. Pollefeys, J. Matas, “USAC: A Universal Framework for Random Sample Consensus,” in the IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI), 2013. [Online]. Available: https://www.researchgate.net/publication/240308787_USAC_A_Universal_Framework_for_Random_Sample_Consensus

[76] D. R. Myatt, P. Torr, S. Nasuto, J. Bishop, R. Craddock, “NAPSAC: High noise, high dimensional robust estimation – its in the bag,” in the Proceedings of the British Machine Vision Conference, 2002. [Online]. Available: https://www.researchgate.net/publication/221259336_NAPSAC_High_Noise_High_Dimensional_Robust_Estimation_-_it's_in_the_Bag

[77] O. Chum, J. Matas, “Matching with PROSAC – progressive sample consensus,” in the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), 2005. [Online]. Available: https://ieeexplore.ieee.org/document/1467271

[78] D. Barath, J. Matas, J. Noskova, “MAGSAC: Marginalizing Sample Consensus,” in the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2019. [Online]. Available: https://ieeexplore.ieee.org/document/8953287

[79] D. Barath, J. Noskova, M. Ivashechkin, J. Matas, “MAGSAC++, a Fast, Reliable and Accurate Robust Estimator,” in the IEEE Conference on Computer Vision and Pattern Recogntion (CVPR), 2020. [Online]. Available: https://ieeexplore.ieee.org/document/9156822

[80] L. Nogueira, E. C. de Paiva, G. Silvera, “Towards a Unified Approach to Homogrpahy Estimation Using Image Features and Pixel Intensities,” arXiV, 2022. [Online]. Available: https://arxiv.org/abs/2202.09716

[81] J. Zhao, “An Efficient Solution to Non-Minimal Case Essential Matrix Estimation,” in the IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 44, no.4, pp.1777-1792, 2022. [Online]. Available: https://ieeexplore.ieee.org/document/9220804

[82] H. Le, F. Liu, S. Zhang, A. Agarwala, “Deep Homography Estimation for Dynamic Scenes,” In the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2020, pp. 7649-7658. [Online]. Available: https://ieeexplore.ieee.org/document/9157755

[83] R. Ranftl, V. Koltun, “Deep Fundamental Matrix Estimation,” in the European Conference on Computer Vision (ECCV), 2018. Springer. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-030-01246-5_18

[84] C. Stachniss (2014). “Robust Least Squares for SLAM,” University of Bonn, Photogrammetry & Robotics Lab. [Lecture]. Available: https://www.ipb.uni-bonn.de/html/teaching/msr2-2020/sse2-08-robust-slam.pdf

[85] Q. Li, R. Li, K. Ji, W. Dai, "Kalman Filter and Its Application," 2015 8th International Conference on Intelligent Networks and Intelligent Systems (ICINIS), Tianjin, China, 2015, pp. 74-77, doi: 10.1109/ICINIS.2015.35.

[86] L. Kneip, R. Siegwart, and M. Pollefeys, “Finding the exact rotation between two images independently of the translation,” In the European Conference on Computer Vision, 2012. Springer. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-642-33783-3_50

[87] D. Muhle, L. Koestler, N. Demmel, F. Bernard, D. Cremers, "The Probabilistic Normal Epipolar Constraint for Frame- To-Frame Rotation Optimization under Uncertain Feature Positions," 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), New Orleans, LA, USA, 2022, pp. 1809-1818, doi: 10.1109/CVPR52688.2022.00186.

[88] R. A. Newcombe, S. J. Lovegrove, A. J. Davison, “DTAM: Dense tracking and mapping in real-time,” in the Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2011. [Online]. Available: https://www.researchgate.net/publication/221111724_DTAM_Dense_tracking_and_mapping_in_real-time

[89] A. Kendall, M. Grimes, R. Cipolla, "PoseNet: A Convolutional Network for Real-Time 6-DOF Camera Relocalization," 2015 IEEE International Conference on Computer Vision (ICCV), Santiago, Chile, 2015, pp. 2938-2946, doi: 10.1109/ICCV.2015.336.

[90] A. Geiger, P. Lenz, R. Urtasun, “Are we ready for autonomous driving? the KITTI vision benchmark suite,” In the Conference on Computer Vision and Pattern Recognition (CVPR), 2012.

[91] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, V. Koltun, “CARLA: An Open Urban Driving Simulator,” in the Proceedings of the 1st Annual Conference on Robot Learning, 2017, pp. 1-16.

[92] G. Araguas, C. Paz, G. P. Paina, L. Canali, “Visual Homography-based Pose Estimation of a Quadrotor using Spectral Features,” in the Latin America Congress on Computational Intelligence (LA-CCI), 2015. [Online]. Available: https://ieeexplore.ieee.org/document/7435939

[93] J. Zhang, V. Ila and L. Kneip, "Robust Visual Odometry in Underwater Environment," 2018 OCEANS - MTS/IEEE Kobe Techno-Oceans (OTO), Kobe, Japan, 2018, pp. 1-9, doi: 10.1109/OCEANSKOBE.2018.8559452.

[94] A. J. Swank. “Localization Using Visual Odometry and a Single Downward-Pointing Camera,” NASA Glenn Research Center, 2012. Cleveland, Ohio. [Online]. Available: https://ntrs.nasa.gov/citations/20120016473

[95] F. Dellaert. (2020). “Perception in Aerial, Marine & Space Robotics: a Biased Outlook.” In the IEEE International Conference on Intelligent Robots and Systems (IROS) Keynote. [Online]. Available: https://www.youtube.com/watch?v=F_MpBIeO0c0

[96] N. Nourani-Vatani, P. V. K. Borges, “Correlation-based Visual Odometry for Ground Vehicles,” in the Journal of Field Robotics, 2011. [Online]. Available: https://www.researchgate.net/publication/220648273_Correlation-based_visual_odometry_for_ground_vehicles

[97] Y. Yoshida, I. Horiba, S. Yamamoto, N. Sugie, “Determining 3D Informations together with Correspondence from a Sequence of Orthographically Projected Optical Flows,” in IEEE Conference on Systems, Man, and Cybernetics, 1999. [Online]. Available: https://ieeexplore-ieee-org.proxy.lib.ohio-state.edu/document/823251

[98] M. Zucchelli, J. Santos-Victor, H. I. Christensen, “Constrained Structure and Motion Estimation from Optical Flow,” in the International Conference on Pattern Recognition, 2002. IEEE. [Online]. Available: https://ieeexplore.ieee.org/document/1044712

[99] A. Fakih, J. Zelek, “Structure from Motion: Combining Features Correspondences and Optical Flow,” in the 19th International Conference on Pattern Recognition, 2008. IEEE. [Online]. Available: https://ieeexplore.ieee.org/document/4761007

[100] J. Zienkiewicz, A. Davison, “Extrinsics Autocalibration for Dense Planar Visual Odometry,” in the Journal of Field Robotics, 2014. [Online]. Available: https://www.researchgate.net/publication/268528977_Extrinsics_Autocalibration_for_Dense_Planar_Visual_Odometry

[101] M. O. A. Aqel, M. H. Marhaban, M. I. Saripan, N. B. Ismail, A. Khmag, “Optimal Configuration of a Downward-Facing Monocular Camera for Visual Odometry,”

Indian Journal of Science and Technology. May 2016. [Online]. doi: 10.17485/ijst/2015/v8i32/92101.

[102] N. Nourani-Vatani, J. Roberts, M. V. Srinivasan, “Practical Visual Odometry for Car-like Vehicles,” in the IEEE International Conference on Robotics and Automation, 2009. May, 2009. [Online]. Available: https://ieeexplore.ieee.org/document/5152403

[103] S. Hong, J. Song, J. Baek, J. Ryu, “Visual Odometry for Outdoor Environment using a Downward-Tilting Camera and Self-Shadow Removal Algorithm,” in the 12th International Conference on Control, Automation and Systems, 2012. Oct., 2012. [Online]. Available: https://ieeexplore.ieee.org/document/6393363

[104] J. Zienkiewicz, R. Lukierski, A. Davison, “Dense, Auto-Calibrating Visual Odometry from a Downward-Looking Camera,” in the British Machine Vision Conference, 2013. [Online]. Available: https://www.doc.ic.ac.uk/~ajd/Publications/zienkiewicz_etal_bmvc2013.pdf

[105] M. Lee, K. Kim, S. Kim, “Measuring Vehicle Velocity in Real Time Using Modulated Motion Blur of Camera Image Data,” in the IEEE Transactions on Vehicular Technology, vol. 66, no. 5, 2017.

[106] M. Gilles, S. Ibrahimpasic, “Unsupervised deep learning based ego motion estimation with a downward facing camera,” The Visual Computer, 2021. [Online]. Available: https://link.springer.com/article/10.1007/s00371-021-02345-6

[107] M. Jaderberg, K. Simonyan, A. Zisserman, K. Kavukcuoglu, “Spatial Transformer Networks,” in the Proceedings of Advanced Neural Information Processing Systems, 2015.

[108] M. V. Ornhag, Marten Wadenback, “Planar Motion Bundle Adjustment,” In Proceedings of the 8th International Conference on Pattern Recognition Applications and Methods (ICPRAM), 2019. pp. 24-31.

[109] M. Wadenback, A. Heyden, “Planar Motion and Hand-Eye Calibration Using Inter-Image Homographies from a Planar Scene,” In International Conference on Computer Vision Theory and Applications (VISAPP), 2013. Barcelona, Spain. pp 164–168.

[110] B. Lee, K. Daniilidis, D. D. Lee, “Online Self-Supervised Monocular Visual Odometry for Ground Vehicles,” in the IEEE International Conference on Robotics and Automation (ICRA), 2015. Seattle, WA. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/7139928

[111] B. Kitt, J. Rehder, A. Chambers, “Monocular Visual Odometry using a Planar Road Model to Solve Scale Ambiguity,” in the Proceedings of the European Conference on Mobile Robots, 2011. [Online]. Available: https://www.researchgate.net/publication/236021144_Monocular_Visual_Odometry_using_a_Planar_Road_Model_to_Solve_Scale_Ambiguity

[112] CAD Block Libraries, “Tesla Motors Model 3 Tesla Inc electric car side Autocad block,” Accessed: Apr. 2023. [Online]. Available: https://ceco.net/autocad-

blocks/vehicles/cars/side-view/autocad-drawing-tesla-inc-model-3-tesla-motors-electric-car-side-dwg-dxf-436

[113] iStock, “Car Drawing stock illustrations,” Accessed: Apr. 2023. [Online]. Available: https://www.istockphoto.com/illustrations/car-drawing

[114] G. Farneback, “Two-Frame Motion Estimation Based on Polynomial Expansion,” in Lecture Notes in Computer Science, 2003. [Online]. Available: https://www.researchgate.net/publication/225138825_Two-Frame_Motion_Estimation_Based_on_Polynomial_Expansion

[115] Z. Wang, A. C. Bovik, H. R. Sheikh, E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," in the IEEE Transactions on Image Processing, vol. 13, no. 4, pp. 600-612, April 2004, doi: 10.1109/TIP.2003.819861.

[116] Y. Zheng, S. Nobuhara and Y. Sheikh, "Structure from motion blur in low light," in the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2011, Colorado Springs, CO, USA, 2011, pp. 2569-2576, doi: 10.1109/CVPR.2011.5995594.

[117] J. Qiu, X. Wang, S. J. Maybank, D. Tao, "World From Blur," 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 2019, pp. 8485-8496, doi: 10.1109/CVPR.2019.00869.

[118] I. Rekleitis, “Visual Motion Estimation based Motion Blur Interpretation,” M.S. thesis, Department of Computer Science, McGill University, Montreal, Canada, 1995. [Online]. Available:

https://www.researchgate.net/publication/2687203_Visual_Motion_Estimation_based_on_Motion_Blur_Interpretation

[119] O. Kupyn, V. Budzan, M. Mykhailych, D. Mishkin, J. Matas, "DeblurGAN: Blind Motion Deblurring Using Conditional Adversarial Networks," 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 2018, pp. 8183-8192, doi: 10.1109/CVPR.2018.00854.

[120] O. Kupyn, T. Martyniuk, J. Wu, Z. Wang, "DeblurGAN-v2: Deblurring (Orders-of-Magnitude) Faster and Better," 2019 IEEE/CVF International Conference on Computer Vision (ICCV), Seoul, Korea (South), 2019, pp. 8877-8886, doi: 10.1109/ICCV.2019.00897.

[121] P. Liu, J. Janai, M. Pollefeys, T. Sattler, A. Geiger, "Self-Supervised Linear Motion Deblurring," in IEEE Robotics and Automation Letters, vol. 5, no. 2, pp. 2475-2482, April 2020, doi: 10.1109/LRA.2020.2972873.

[122] P. Liu, X. Zuo, V. Larsson, M. Pollefeys, "MBA-VO: Motion Blur Aware Visual Odometry," 2021 IEEE/CVF International Conference on Computer Vision (ICCV), Montreal, QC, Canada, 2021, pp. 5530-5539, doi: 10.1109/ICCV48922.2021.00550.

[123] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, I. Polosukhin, “Attention is all you need,” in the Proceedings of the Neural Information Processing Systems (NeurIPS), 2017.

[124] E. Riba, D. Mishkin, D. Ponsa, E. Rublee and G. Bradski, "Kornia: an Open Source Differentiable Computer Vision Library for PyTorch," 2020 IEEE Winter Conference on Applications of Computer Vision (WACV), Snowmass, CO, USA, 2020, pp. 3663-3672, doi: 10.1109/WACV45572.2020.9093363.

[125] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, A. Lerer, “Automatic differentiation in PyTorch,” in the Proceedings of the Neural Information Processing Systems, Autodiff Workshop, 2017.

[126] G. Bradski, “The OpenCV Library,” in the Dr. Dobb’s Journal of Software Tools, 2000. [Online]. Available: https://opencv.org/

[127] R Y. Zhou, C. Barnes, J. Lu, J. Yang and H. Li, "On the Continuity of Rotation Representations in Neural Networks," 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 2019, pp. 5738-5746, doi: 10.1109/CVPR.2019.00589.

[128] D. P. Kingma, Jimmy Ba, “Adam: A method for stochastic optimization,” arXiv preprint arXiv:1412.6980, 2014.

[129] P. Georgel, S. Benhimane, N. Navab, “A Unified Approach Combining Photometric and Geometric Information for Pose Estimation,” in the British Machine Vision Conference, 2008.

[130] L. Pineda, T. Fan, M. Monge, S. Venkataraman, P. Sodhi, R. T. Q. Chen, J. Ortiz, D. DeTone, A. Wang, S. Anderson, J. Dong, B. Amos, M. Mukadam, “Theseus:

A Library for Differentiable Nonlinear Optimization," in the Advances in Neural Information Processing Systems, 2022.

[131] J. Davis (2021). "Camera Models and Calibration," Computer Vision for HCI (CSE 5524) [Powerpoint slides].

[132] M. Kok, J. D. Hol, T. B. Schon, "Using Inertial Sensors for Position and Orientation Estimation," arXiV, 2018. [Online]. Available: https://arxiv.org/abs/1704.06053

[133] N. Jonnavithula, Y. Lyu, Z. Zhang, "LiDAR Odometry Methodologies for Automated Driving: A Survey," arXiV, 2021. [Online]. Available: https://arxiv.org/pdf/2109.06120

[134] Z. Zhang, "Iterative point matching for registration of free-form curves," in the International Journal of Computer Vision, pp. 119-152, 1994. [Online]. Available: https://link.springer.com/article/10.1007/BF01427149

[135] D. Chetverikov, D. Svirko, D. Stepanov, P. Krsek, "The Trimmed Iterative Closest Point algorithm," in the International Conference on Pattern Recognition, 2002, pp. 545-548. IEEE. doi: 10.1109/ICPR.2002.1047997.

[136] A. Segal, D. Haehnel, S. Thrun, "Generalized-ICP." in Robotics: Science and Systems, 2009. [Online]. Available: https://www.researchgate.net/publication/221344436_Generalized-ICP

[137] S. Bouaziz, A. Tagliasacchi, M. Pauly, "Sparse iterative closest point," in Computer Graphics Forum, vol. 32, no. 5. Wiley Online Library, 2013, pp. 113-123. [Online]. Available: https://onlinelibrary.wiley.com/doi/10.1111/cgf.12178

[138] J. Zhang, S. Singh, “Loam: Lidar Odometry and Mapping in real-time,” in Robotics: Science and Systems, 2014. [Online]. Available: https://www.researchgate.net/publication/311570125_LOAM_Lidar_Odometry_and_Mapping_in_real-time

[139] C. R. Qi, H. Su, K. Mo, L. J. Guibas, “Pointnet: Deep Learning on point sets for 3d classification and segmentation,” in Proceedings of the IEEE conference on Computer Vision and Pattern Recognition, 2017, pp. 652-660.